Abstract

Web 2.0 applications have been celebrated for creating opportunities for user generated content and have also prompted concerns about the collection and storage of user information. This paper discusses the infrastructures driving dynamic Web applications — focusing particularly on Ajax (Asynchronous JavaScript and XML) — and argues that these infrastructures encourage a particular style of user participation that aids the data collection and analysis activities of major Web 2.0 platforms. While these applications provide simple interfaces for many to many content production, in the process they have often inserted themselves as intermediaries. By focusing on the role of Ajax, this paper illustrates ways in which taken-for-granted infrastructures can shape the sorts of communication patterns and relationships that are built upon the Web.

Introduction

Early discourses of Web 2.0 described a better, more dynamic and more participatory evolution of Web 1.0. Optimistic commentators remarked that Web 2.0 created new opportunities for individuals to create and distribute content with equal (or at least comparable) power as large corporations (Grossman, 2006; Shirky, 2010). But that excitement is increasingly displaced by concerns about the collection and storage of user information on the platforms that facilitate participation (Albrechtslund, 2008; Fuchs, *et al.*, 2012; Roosendaal, 2012, 2011). Many of the platforms to which users entrusted their data have grown into or been acquired by monolithic corporations whose business models revolve around combining and analyzing 'big data.' If Web 2.0 allowed amateur media creators to "[beat] the pros at their own game" (Grossman, 2006), it also allowed an emerging class of data-focused companies such as Google to win at a different game entirely. McChesney commented, "It is supremely ironic that the Internet, the much-ballyhooed champion of increased consumer power and cutthroat competition, has become one of the greatest generators of monopoly in economic history" [1].

In this paper I describe how the technical infrastructures driving dynamic Web applications — a core feature of Web 2.0 — facilitate collection of users' information and constrain the ability of users to make decisions about how and where their data is accessed. Web applications have created simple interfaces for decentralizing content production and many to many communications, but in the process have inserted themselves as intermediaries. This has concentrated the capacity to distribute online content as well as to capture and analyze users' information among a small number of corporate platforms.

In this paper I refer to both *Web applications* and *Web platforms*. *Web application* refers to Web sites that have been developed to provide similar functionality as desktop (and smartphone) software. These applications run in a Web browser, but unlike early Web pages are not limited to largely static content. Instead, Web applications allow fluid interactivity such as re-arranging and editing data, manipulating graphics, or playing games. The Web sites for Google Maps and Facebook are both examples of Web applications that allow software-like functionality in a Web browser. This paper is concerned with Web applications that form parts of or otherwise integrate with Web platforms. The term *platform* has become widely adopted to refer to "content intermediaries" that host user-generated content, such as social media sites [2]. This term performs powerful discursive work and has been used to simultaneously convey appealing connotations to users and marketers, as well as to favourably frame the legal and political responsibilities of platform owners (Gillespie, 2010). Gillespie (2010) identifies several dimensions implied in the term:

> Computational, something to build upon and innovate from; political, a place from which to speak and be heard; figurative, in that the opportunity is an abstract promise as much as a practical one; and architectural, in that YouTube is designed as an open-armed, egalitarian facilitation of expression, not an elitist gatekeeper with normative and technical restrictions.[3]

The Web sites for Facebook and Google Maps are both Web applications that are integrated with their parent platforms, upon which millions of users have built networks of maps status updates, like buttons, and other content embedded throughout the Web. Helmond (2015) asserts that a decentralization of Web features has occurred in tandem with a recentralization of "platform ready" data, as platforms simultaneously extend throughout the Web and collect standardized data in a central repository.

Critical scholarship has often focused on how people interpret and use Web 2.0. These approaches have been fruitful, and effectively represent how Web 2.0 was defined largely by changes to user activities (especially participation). However, these changes to users' behaviours occurred in relation to the technical infrastructures of the Web, and accounting for this infrastructure promises to contribute to a greater understanding of the Web's effects. I draw upon the growing field of infrastructure studies, which examines infrastructures in relation to their histories and social effects (Jackson, *et al.*, 2007; Sandvig, 2013; Star, 1999). The story I share in this article parallels Sandvig's (2015) account of how the Internet changed from an 'anti-television' point-to-point system into one designed to favour mass-distribution of commercial video. Where Sandvig illustrates that "the availability and quality of video on the Internet are significant new political and economic battlegrounds where culture is controlled" [4], I argue that the technologies behind dynamic Web applications are similar battlegrounds.

Specifically, I describe the role of Ajax (Asynchronous JavaScript and XML). Ajax is used to transfer information from the server without reloading the page, allowing "almost instant" interactivity (Garrett, 2005). This is a key technique for dynamic Web applications, such as Web 2.0 exemplars Facebook and Google Maps. Ajax is used to transfer information from the server without reloading the page, allowing "almost instant" interactivity (Garrett, 2005). These and other Web 2.0 applications have been rightfully lauded for making it easier to produce, access, and share content. To do so, they draw upon complex infrastructures for storing, arranging, sharing and displaying users' content. This means that while Web 2.0 has decentralized content production, the storage and arrangement of user content is concentrated in a small number of platforms. To develop an understanding of the benefits and costs of this state of affairs, it is necessary to investigate the infrastructures supporting Web applications. Web 2.0's seemingly frictionless pathways are facilitated by tremendous amounts of labour, resources, and debate. Each technical capability or limitation is the result of negotiations among sophisticated systems of hardware, software, people, and standards, and these negotiations have both winners and losers. Thus by investigating *how* things work, one can also question *why* they work, and who gains or loses from this arrangement.

I argue that the technical infrastructures of Web applications support an arrangement where content production has been decentralized, but the storage, arrangement, and distribution of Web content is concentrated among a small number of platforms. By maintaining a constant data exchange between server and browser, Ajax applications create similar traffic demands as video (Schneider, *et al.*, 2008), so corporate Ajax applications benefit from the distribution infrastructures described by Sandvig (2015). Moreover, popular Web 2.0 platforms use Ajax to constantly reconfigure the content of data on the server and the structure of its display in a Web browser. To aid this, user content is heavily structured, which simultaneously aids back-end 'big data' analysis. Moreover, persistent change makes users' content transient — encouraging frequent content submission — and provides users with minimal control about how and where their content will be displayed. By examining the role of Ajax, this paper illustrates how the infrastructures behind dynamic Web applications encourage a particular style of user participation that aids and obfuscates the data collection and analysis activities of major Web 2.0 platforms.

The first part of this paper describes the methodologies of infrastructure studies, discussing online video distribution to illustrate how this field can illuminate social and cultural effects of the Internet. I then discuss the meanings of Web 2.0, which differed between several groups. Particular focus is placed on the technical and business suggestions made by O'Reilly, which offer insights into the design of Web 2.0 infrastructures. Following this, I describe how Web applications function, especially through using Ajax. And finally, I discuss social and cultural effects of these application infrastructures.

Infrastructural perspectives

Infrastructure studies is "a call to study boring things" [5]. The infrastructure of the Internet is made up of seemingly dry and technical mechanisms for facilitating information exchanges in a large network. These involve many layers, from protocols for distributing data between network nodes to standards for writing HTML, JavaScript and other code that can be interpreted by Web browsers. When seeking to understand social and cultural impacts of the Internet, studying the nuances of coding standards can indeed seem like tedious path. However, identifying when and how changes to these standards occurred and addressing the factors that spurred these changes

can reveal a great deal about the ethics and values taken for granted in these structures. Star (1999) compared the infrastructures of information systems to those of cities:

> Study a city and neglect its sewers and power supplies (as many have), and you miss essential aspects of distributional justice and planning power (Latour and Hermant, 1998). Study an information system and neglect its standards, wires, and settings, and you miss equally essential aspects of aesthetics, justice, and change. [6]

Infrastructure studies can serve as a bridge between social and technical perspectives. One of the strengths of this approach is that it encourages perspectives of technology where "the emphasis is not on novelty but continuity and consistency with the past" (Jackson,*et al.*, 2007). This is particularly relevant to a study of Web 2.0, which was imagined as a wholly new version of the Web (it was called '2.0' after all) but which actually describes a set of practices and perspectives of the Web with lineages deep into Web 1.0 and the early Internet.

The infrastructure of Web distribution

In his account of Internet video distribution, Sandvig (2015) illustrated the infrastructural approach by describing a series of efforts to restructure the Internet to facilitate mass-video distribution (*e.g.*, Netflix). The infrastructures that shape the availability and quality of online video, he argued, "are significant new political and economic battlegrounds where culture is controlled" [7]. The conflicts that unfold atop these battlegrounds have many dimensions, and seek to resolve numerous, sometimes clashing disputes. From users' perspectives, online video is deceptively simple, but this belies the complex systems of "labor, infrastructures, institutions, economics, and numerous stakeholders" involved in distributing video data across the Internet while navigating technical obstacles, consumer wants, licensing agreements and other challenges [8]. Each time one issue is addressed, it may create new problems or resurface old tensions elsewhere. Streaming video and audio now constitutes over 70 percent of Internet traffic in North America (Sandvine, 2015), and thus imposes considerable strain on network infrastructures. This makes online video a valuable site to investigate the tensions, negotiations, and dramas that shape these infrastructures. As well as demonstrating the importance of infrastructures generally, Sandvig's (2015) account illuminates several patterns, which are paralleled in the structures of Web applications that I focus on in this paper.

The original design of the Internet utilized a point-to-point infrastructure, where it was assumed each Internet user should have equal ability to upload and download data. Many of the designs of the network were influenced by J.C.R. Licklider, the first director of the Pentagon's Information Processing Techniques Office, who conceived of an interactive communication system that would, in contrast to television's lack of interaction, encourage individual participation and would appreciate "the interaction value of diversity among the personalities, [and] interest patterns, of individuals as well as the cohesion value of the community in language and cultural heritage"[9]. In adopting Licklider's model, Sandvig asserted, the Internet was modeled as an *anti-television*. When Tim Berners-Lee designed the Web — the part of the Internet accessed through Web browsers using HTTP, HTML, and related technologies — he similarly emphasized the ability of individuals to participate. The early users of the Web were assumed to be elite knowledge workers who would have reason to communicate with each other as equals, and indeed the first version of the Web was designed to appeal to researchers at the European Organization for Nuclear Research (CERN, or *Organisation européenne pour la recherche nucléaire*) (Berners-Lee and Fischetti, 1999).

To support the ability of Internet users to communicate as equals, the network infrastructure utilized a point-to-point structure. Each individual computer or device attached to the Internet was an equal node, and could upload or download data at the same rate. This differed substantially from the antenna broadcasting technology that drove mass television distribution. A broadcast antenna radiates a signal such that it makes no difference whether it is received by one person or one hundred, but the point-to-point structure of the Internet functioned more like the postal system. Just like postal distribution scales linearly, so that sending one hundred postcards requires one hundred distinct deliveries to be made (at one hundred times the cost and effort as sending one postcard), in a point-to-point network, distributing data to one hundred people requires one hundred distinct transmissions [10]. As a result of this point-to-point infrastructure, large media companies do not have the same advantages of scale as they do in mass-broadcasting. Producing

and distributing writing or other media on the Internet does not require a large upfront investment in distribution facilities, but instead individuals can distribute content more or less as simply as consuming it. This idea of universal accessibility has been one of the driving imaginations of the Internet, and plays a central role in arguments about the flattening world; from Thomas Friedman's (2005) influential *The world is flat* to mass-media descriptions of Web 2.0 (Grossman, 2006) and reddit co-founder Alexis Ohanian's description of his motivations for founding the Web 2.0 giant (Ohanian, 2013) the idea that the Internet's structure allows anyone to participate is a central narrative.

As the Internet became more popular, however, the majority of users were no longer 'elite knowledge workers,' but instead spanned a broad variety of people from all walks of life. And many of these people were more interested in consuming content than producing it. The point-to-point model assumed that communication on the Internet would be reciprocal, but as Sandvig (2015) noted, when an Internet user in Michigan downloads an article from the *New York Times*, there is little reason for a reporter from the *Times* to download an equivalent article from that reader. Instead, it is economical for large media producers like the *Times* to have greater distribution capabilities than most Internet users, who consume more content than they create. This is particularly evident for high-bandwidth applications such as streaming video. Users of these services want to access video at the highest possible quality without being interrupted by the ubiquitous (and infamous) spinning wheel that indicates buffering is required. In networking lingo, this requires a high quality of service (QoS) where every byte of data arrives promptly and in the right order, so they can be properly assembled into a video stream without interruptions.

To accommodate the increasing need for asymmetric data, Internet engineers made deliberate modifications to the network infrastructure. Among the most significant of these changes was the creation of content distribution networks (CDNs) by companies like Akamai and Amazon. CDNs distribute data to multiple servers located in close proximity to Internet users. By paying for CDN distribution, content providers can ensure their data is loaded from a nearby server to its recipient, so will load quickly. When users wonder why the advertisements before their video sometimes load faster than the video itself, the answer may be that the advertisements are distributed through a CDN (Sandvig, 2015). Those content producers that can afford access to a CDN can consistently deliver content more quickly to consumers than those without CDN access. Sandvig additionally expressed concern that, until Amazon entered the CDN market in 2008, the best CDNs refused customers who were not affiliated with major corporate content producers (now independent media producers can subscribe to Amazon's CDN services if they can afford them) [11].

Additionally, asymmetry between users and producers is expressed in the capabilities of broadband Internet services, where users' download speed is much greater than their upload speed. Moreover, Internet users are rarely permitted to run their own servers, and if they do, they must compete with content offered through prohibitively expensive CDNs, meaning "content from a mainstream, well-capitalized media company would load perceptibly faster than what the user offered" [12]. This shift toward asymmetric networking occurred alongside the emergence of Web 2.0. In the midst of excitement over new opportunities for user participation, ignoring the increasingly centralized control over distribution servers means ignoring significant constraints within which Web 2.0 participation must occur.

With regard to understanding the infrastructure of Web applications, Sandvig's paper is useful in two regards. First, it demonstrates how focusing on infrastructure can illuminate aspects of the story that would be obscured from other perspectives:

> Telling the story of Internet video without the above focus on distribution could wrongly make it seem that the development of online video was purely a matter of user preference. A future analyst might one day wrongly conclude that the story was: "For a while, early Internet users made and watched their own videos about cats, but then they wanted to watch mainstream media offerings like *Modern Family*." In fact, re-orienting the Internet audience toward mass offerings has been a coevolution of taste, massive infrastructural investments, and important technological achievements. [13]

Second, the story of Web applications complements Sandvig's story. The same networking structures that drive video distribution also drive access to Web applications. And many of the development techniques and tools involved in producing Web applications support a similar

asymmetric relationship, where a small number of corporate platforms provides service to a large audience of users.

Defining Web 2.0

One of the tenets of infrastructure studies is that breakdowns makes transparent infrastructure become visible (Jackson, *et al.*, 2007; Star, 1999). Perhaps the most abrupt and significant breakdown in the Web's short history was the dot-com crash of 2001. It was in the aftermath of the crash that the term Web 2.0 originated. This decidedly rough transition between Web 1.0 and Web 2.0 provides an entrance for understanding infrastructures of the Web more broadly.

In 2004, while planning a Web industry conference, O'Reilly Media's Dale Dougherty and Tim O'Reilly concluded that the Web had not been destroyed, but was in fact more important than ever. They believed the crash marked a critical turning point for the Web, and coined the term Web 2.0 to denote that the Web had become something different than it was before (O'Reilly, 2005). Exactly what the Web had become — what defined Web 2.0 — has been difficult to identify (Cormode and Krishnamurthy, 2008; Song, 2010). In fact, two years after O'Reilly (2005) provided the first comprehensive definition of the term, he remarked that Web 2.0 was actually "a pretty crappy name" for what was happening (O'Reilly, 2007). Part of the confusion stems from the fact that Web 2.0 was considered by different people to describe both a social and a technical/business phenomenon. Consider two contrasting perspectives of Web 2.0:

> For seizing the reins of the global media, for founding and framing the new digital democracy, for working for nothing and beating the pros at their own game, *Time's* Person of the Year for 2006 is you. (Grossman, 2006)

> Companies that succeed will create applications that learn from their users, using an architecture of participation to build a commanding advantage not just in the software interface, but in the richness of the shared data. (O'Reilly, 2005).

Mainstream media outlets such as *Time* defined Web 2.0 as "a massive social experiment" in which individual Web users had unprecedented opportunities for participation (Grossman, 2006). But the emphasis of O'Reilly's (2005) original definition was on "design patterns and business models".

While the phenomenon described by Web 2.0 is real, the term performs a number of discursive tasks that shaped perceptions of what sort of innovations would be possible in the new Web. The term Web 2.0 utilizes a metaphor of software versioning, and this implies a sense of being *new and improved*. There was a transformation underway in which participation was becoming increasingly central to the Internet, but this transformation didn't happen as abruptly or absolutely as the change of moniker from *1.0* to *2.0* would imply. Scholz wrote that discourse around Web 2.0 served to align conceptions of Web participation with market interests (2008). And Fuchs argued the main purpose of using the term Web 2.0 "seems to be a marketing strategy for boosting investment" [14]. These claims are supported by the fact that the Web 2.0 conference for which the term was originally coined was reportedly "full of prowling VCs looking for the next hot startup" (Graham, 2005). In this light, it is clear that the sense of 'progress' established by the 2.0 moniker served efforts to reassure investors of the Web's vitality after the dot-com crash.

The new Web was to be oriented around *platforms*, service-oriented Web sites that would facilitate user participation across multiple devices. The most successful platform owners, according to O'Reilly, would be those who successfully extend their data through user participation:

> The key to competitive advantage in Internet applications is the extent to which users add their own data to that which you provide. *Therefore*: Don't restrict your "architecture of participation" to software development. Involve your users both

> implicitly and explicitly in adding value to your application. (O'Reilly, 2005)

It is clear from O'Reilly's business-targeted definition of Web 2.0 that user participation is transactional. Users' data is exchanged in return for access to a platform. His definition is a guide for businesses, not for users. In this definition, the infrastructural qualities of Web 2.0 are front and centre: "Design for 'hackability' and 'remixability';" "Set inclusive defaults for aggregating user data as a side-effect of their use of the application;" "Support lightweight programming models that allow for loosely-coupled systems" (O'Reilly, 2005). It is with these design strategies in mind that I discuss the infrastructures of Web 2.0 applications.

Dynamic Web applications

The fluid user experience offered by Web applications is a major feature of Web 2.0. One of the most significant technologies for such experiences is Ajax (Asynchronous JavaScript and XML). The concept of Ajax was introduced in 2005 by Jesse James Garret (2005), co-founder of the Web consultancy firm Adaptive Path. Ajax was not a new technology, nor was it invented by Garrett, but rather he coined the term to describe how developers were bringing together a set of existing techniques to create new types of Web applications. The big innovation in Ajax was that it allowed developers to refresh part of a website without needing to reload the entire page, allowing more dynamic interactions. Garrett cites Google as one of the first companies to use Ajax:

> Take a look at Google Suggest. Watch the way the suggested terms update as you type, almost instantly. Now look at Google Maps. Zoom in. Use your cursor to grab the map and scroll around a bit. Again, everything happens almost instantly, with no waiting for pages to reload. (Garrett, 2005)

To achieve this "almost instant" responsiveness — a taken for granted feature of today's Web — Ajax applications maintain a dynamic, persistent communication between the Web browser and the server. The classic (pre-Ajax) model for Web applications involved loading a page in one fell swoop, then severing the server connection until the user made another request. At any point when new data needed to be loaded, a new HTML file needed to be retrieved from the server. This meant the whole page needed to be reloaded and the user needed to wait.

Ajax made it possible to exchange data with the back-end without reloading the whole page. When a developer uses Ajax, an intermediary Ajax engine is created between client and server. The Ajax engine is loaded in the background and can communicate with the server asynchronously, retrieving data and modifying the client display without requiring a page refresh. This makes it possible to reproduce the instant interactivity of desktop applications, so actions like auto-completing a search term or scrolling to display a different section of a map can be performed smoothly without waiting.

Ajax is a vital part of Web 2.0. It is no coincidence that the terms "Ajax" and "Web 2.0" originated around the same time. Paul Graham, a programmer and co-founder of Y Combinator (a Silicon Valley startup incubator) wrote in 2005 that Web 2.0 had three components: "Ajax, democracy, and not dissing users" (Graham, 2005). What does Ajax, a technical method for loading and displaying Web content asynchronously, have to do with democracy and respect for users? To answer this question, it is necessary to address that the technologies for Ajax existed since at least 1999, but the term was only coined in 2005 to describe the application-like functionality demonstrated in Google Maps.

The road to Ajax: Via Google Maps

In this section I briefly summarize some key historical developments leading toward dynamic Web applications, from the first text-based Web pages through several precursors and related

technologies to Google Maps, which Graham (2005) credits with "set[ting] off the whole Ajax boom."

The earliest version of HTML could be used to display static documents containing text and links, as demonstrated by the first Web site, a description of the World Wide Web project (Berners-Lee, 1992). The ability to display images in an HTML document was soon implemented in the Mosaic browser, and became a standardized function in HTML 2.0 (Raggett, *et al.*, 1998). In 1995 Brendan Eich, who was working for Netscape at the time, created the first version of JavaScript. JavaScript allows program code to be run within Web sites, for example to create interactive elements. Today, JavaScript is used on the majority of Web sites, and it is one of the core technologies powering Ajax. In 1996, Microsoft released Internet Explorer 3.0, which introduced support for inline frames, or *iframes*. Inline frames could create an area within a Web page in which a separate HTML file could be loaded. An HTML document displayed in an iframe could be refreshed without reloading the rest of the browser window. But this still did not enable anything like the fluidity of Ajax, since the asynchronicity was restricted to a rectangular area with a fixed size. Nonetheless, this was one of the first tools for asynchronous loading on Web pages. Three years later in 1999, Microsoft added another function to Internet Explorer, a JavaScript function called XMLHTTPRequest. This function allows websites to retrieve XML (Extended markup language) data from the server, and display it on a website without refreshing the page. It was shortly thereafter adopted by other browsers. Developers could use JavaScript's XMLHTTPRequest function to retrieve data from a Web server asynchronously, and at this point, all of the necessary technologies for Ajax were in place. So why did the term "Ajax" take another six years to arise, and why was it such a big deal?

Google Maps was one of the earliest Web applications to demonstrate Ajax, and did so compellingly. At the time Google Maps was introduced, other map sites were far less dynamic. To navigate to the left, a user would click a button then have to wait for the page to reload with the map shifted in position. Google Maps made it possible to click and drag on a map to fluidly change position. This was a significant step because getting that sort of fluid interactivity inside a Web browser was previously only possible using plugins such as Flash.

Adobe's Flash software had become the *de facto* standard for application-like interactivity in the Web browser. Unlike JavaScript, iframes, and XMLHTTPRequest, Flash requires installing extra software (*Flash Player*) that runs inside the browser. Once installed, Flash Player makes it possible to play games, watch videos, and otherwise interact with Web content more fluidly and quickly than most other methods that were available at the dawn of Web 2.0. Flash Player was free, popular and easy to get, and in 2004 was installed on more than 93 percent of Web browsers in the United States, Canada and Europe, and 88 percent of browsers in Asia (McAdams, 2005).

Why did Google not just make Google Maps using Flash? One reason is that Flash was owned by Macromedia (which was purchased by Adobe in April 2005), and Google may not have wanted to rely on another company's proprietary technology. In their platform study of Flash, Salter and Murray note that "Flash may demonstrate moments of user-driven evolution, but much is dictated in the top-down hierarchy of traditional software production" [15]. It is reasonable to surmise that Google chose an alternative to Flash in part out of concern that embracing Flash would make them overly reliant on a competitor. This is clearly not the only motivation, since for many years Google's YouTube platform relied on Flash for its superior technical capabilities with regard to online video, and it was only in 2015 that HTML5 video replaced Flash as YouTube's default video format (YouTube, 2015). More important for considering Web 2.0 is that Flash applications are not very lightweight or hackable, and one of the characteristics of successful Web 2.0 companies according to O'Reilly (2005) is that they "support lightweight programming models that allow for loosely-coupled systems". Flash became increasingly complex over time, as its co-creator Jonathan Gay described: "Flash was very lightweight originally. But over time people were developing content for these fast PCs; who cares if it takes 30MB to render this Flash movie, look what it does!"[16]. Additionally, whereas Web users can view the source code of HTML and JavaScript files (by choosing the 'view source' option in their browser) this is not possible with applications built in Flash, presenting an obstacle for developers who want to remix or build modifications of Flash files. This is significant because something that has made Google Maps not just a landmark Ajax application but also a landmark example of Web 2.0 is the way it was immediately remixed by hackers. Shortly after Google Maps was launched, they released programming tools allowing developers to use maps in other applications. But even before these tools were released, several developers had figured out on their own how to access Google Maps data and created mashups, such as Paul Rademacher's housing map that combined Craigslist housing ads with Google Maps (Singel, 2005). Moreover, the lightweight design of Google Maps made it possible to put a Google Map onto any Web page, allowing organizations and individuals to harness the abilities of Google's application.

Perhaps more important than its fluid interactivity was the fact that Google Maps relied on a lightweight, flexible data structure: XML. The X in Ajax stands for Extensible Markup Language (XML), an open standard for machine-readable data. The XML standard is defined by the World Wide Web consortium (W3C), and the standard document begins by articulating the goal, "XML shall be straightforwardly usable over the Internet" (Bray, *et al.*, 2008). In their book *Google Maps hacks*, Rich Gibson and Schuyler Erie expressed the significance of Google's use of XML:

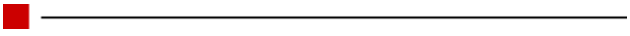
> "Yeah? So what?" you might say. Well, the "so what" is that these data sources are probably open and available for you to scrape. Sometimes they are published, and other times they are hidden in a page, but they are there. [17]

In subsequent years, many Ajax applications including Google Maps have started using JavaScript Object Notation (JSON) instead of or in addition to XML. JSON is a similar format to XML, emphasizing universality through "structured data interchange between all programming languages" (Ecma International, 2013). Google's use of flexible data formats has not only served to encourage mashups and third-party extensions of Google Maps, but also encourage users to contribute data. This is what makes Google Maps not only an example of a Web application, but also of a Web platform upon which users can build their own content. Some of these contributions are overt, such as when users build their own maps using Google's My Maps platform (google.com/mymaps). Others are less explicit, such as Google's automatic collection of anonymous location and speed data through Android mobile phones, which is used to create data about current traffic conditions (Simonite, 2012). Google Maps demonstrated that a Web application could run as smoothly as desktop software, and significantly, that Web applications built with lightweight and flexible data structures could capitalize upon users' data and ingenuity.

Fast and fluid participation

The speed and fluidity demonstrated by Google Maps has become standard in contemporary Web platforms. Web 2.0 platforms of all types, from Google Maps to Facebook to WordPress and beyond, utilize Ajax to facilitate fluid interactivity. Moreover, Ajax is not used just for presenting information to users, but also for transferring user content to the server. Web 2.0 platforms make it easy for their users to seamlessly contribute content. Submitting textual content is typically performed by filling out a simple form: *Enter your post title. Enter your post content. Click to add image. Click 'submit.'* Other types of interactions can be even simpler: *Click 'like.'* This simplicity is what drove people to become so excited about Web 2.0; easy to use platforms have been regarded as encouraging users to participate by creating and sharing content. However, as van Dijck (2009) argued, "It may be crucial to distinguish different levels of participation in order to get a more nuanced idea of what participation entails" [18]. Just what kind of participation are we dealing with?

First and foremost, by making it possible to transmit data between browser and server without reloading a whole page, Ajax allows interactions to occur without the user's intention. On static Web sites where loading new data required refreshing the whole page, initiating an interaction with the server tended to require a deliberate action such as clicking a link. But since Ajax allows data to be loaded without interrupting the user, a wider variety of discrete interactions can be used to trigger data exchanges with the server. These can include scrolling, mouse movement, keyboard input, the time spent on a Web site, or other actions. After capturing these types of interactions, sites can push new data to the user — such as continually pinging a server to check for incoming messages — and retrieve data from the user — such as creating heat maps of mouse movements (Edmonds, *et al.*, 2007) or logging when users paused, played, or fast-forwarded through Web videos (Farney and McHale, 2013).



Many (to platform) to many

That interactions between user and platform have receded into the background, and particularly that these can occur without the user's agency, is not part of the optimistic narrative of Web 2.0

as a wondrous social experiment. But it is a central part of the business model of Web 2.0 to "build a commanding advantage not just in the software interface, but in the richness of the shared data" provided by users (O'Reilly, 2005).

For users, this shared data is fluidly re-organized and presented in a dizzying array of configurations. For advertisers, this data is sorted into a flexible and granular classification system that allows personally targeted advertisements. Serving such personalized content while meeting expectations of 'almost instant' responsiveness requires significant resources. For example, in 2012 Facebook received 2.7 billion like actions and 300 million photos each day (Constine, 2012). Handling this kind of traffic requires massive servers and distribution networks.

The parallels between this description of Web applications and Sandvig's (2015) account of Internet video are explicit here. One of the most striking resemblances is that the traffic requirements of Ajax applications are similar to those of streaming video in that they are constantly loading new data as they run. Ajax applications are intended to run smoothly, constantly loading and displaying new information to mimic the functionality of desktop applications, and this imposes quality of service (QoS) needs onto their Web traffic. One study of Ajax Web traffic characterized it as "heavy (bytes transferred), chatty (many more requests), and greedy (actively pre-fetching data)" compared to overall Web traffic [19]. Like videos, Ajax applications delivered through slow network infrastructures may be interrupted by loading or buffering. Since the traffic demands of both video and Web applications are high, and both forms require a consistent and reliable connection to provide good user experience, the infrastructural developments that favoured corporate video also favour corporate Ajax applications. As a result, the predominantly large and corporate providers who have access to the best content delivery networks can provide a more attractive service than small businesses and individuals with fewer resources. While content creation has been decentralized among billions of Internet users, hosting this content is centralized to a small number of companies.

The same structures that make Web content flexible on the front end, so it can be rearranged by Web interfaces and remixed by users, make it suitable for big data analysis. User generated content is typically accompanied by heavily structured metadata, and interactions such as viewing or sharing content can also be quantified and stored. The analytic potential of this data is illustrated by the extent researchers have eagerly used Twitter data for sociological studies (Kwak, *et al.*, 2010; Takhteyev, *et al.*, 2012). However, researchers only have access to a small subset of the data captured and used by the platforms themselves. As Manovich (2012) asserted, "An anthropologist working for Facebook or a sociologist working for Google will have access to data that the rest of the scholarly community will not" [20]. The vast majority of large Web platforms are commercial enterprises, and the information they collect from users is a valuable resource. Fuchs (2012) found that, as of 2009, "92.3% of the most frequently used Web 2.0 platforms in the US and 87.4% of monthly unique Web 2.0 usages in the U.S. are corporate-based, which shows that the vast majority of popular Web 2.0 platforms are mainly interested in generating monetary profits and that the corporate Web 2.0 is much more popular than the non-corporate Web 2.0" [21]. The companies behind the largest Web 2.0 platforms do tend to be relatively open with providing access to their collected data, but they almost always collect far more information than they make publicly available.

Not only does this make it possible for a small number of companies to collect and commodify a huge amount of personal data, but it also installs them as gatekeepers. Consider this dominance in relation to the hopes and promises of the early Web. Benkler (1999) urged that efforts be undertaken so the great prize of cyberspace would be "the unmediated conversation of the many with the many"[22]. And Tim Berners-Lee (2010) emphasized that the decentralized nature of the Web meant, "You do not have to get approval from any central authority to add a page or make a link" [23]. In contrast, today's many to many communication is mediated through a small number of very large platforms and users must agree to terms and conditions to join these sites. In some cases, these terms have been enforced in ways that marginalize specific groups. A recent controversy about Facebook's real name policy provides an example. In an effort to prevent abuse by anonymous users, Facebook requires that users be identified by their real names. However, this policy has resulted in the suspension of some accounts belonging to Native Americans such as Lone Hill, whose legal name was perceived as fake by Facebook, and to drag queens such as Sister Roma whose profile name does not match her government identification (Holpuch, 2015). Not only do users need Facebook's permission to join the many to many conversations it facilitates, but if they fail to conform to Facebook's expectations of identity then obtaining that permission may be difficult. These cases draw attention to a breakdown where an infrastructural component — Facebook's real name policy — becomes a suddenly visible obstacle. Another instance where mediation of the Internet's many to many conversation becomes apparent is when governments ban social media sites to censor information, such as when Turkey briefly blocked access to Twitter to prevent images of a bombing from spreading, and to prevent Twitter users

from calling for protests against the government (CBC News, 2015). Communication conducted through the Internet may face both intentional and accidental restrictions as it becomes mediated through a small number of platforms.

Separating form and content: A division of labour

To serve the data analysis goals of advertising supported platforms, the data users submit through Web applications must be in computable forms. To create applications that instantly respond to a breadth of potential interactions, the depth of those actions must be constrained. To illustrate this, the following section will sketch a trajectory of how the activities involved in creating Web content have changed over time.

In the early days of the Web, creating content was more likely to involve coding an HTML document, and it was common to describe Web content using metaphors of documents and pages. These metaphors conjure imagery of material collections of information such as books or stacks of paper. These metaphors are not absent from Web 2.0, but have been replaced in many instances with terms like *content* and *data*. The difference between a document and data, and between a page and content, is that documents and pages consist of both form and content, whereas data and content themselves are information without form [24]. As John Perry Barlow quipped in the introduction to Cory Doctorow's book *Content*, "'Content,' Ha! Where's the container?" (Barlow, 2008). With most Web 2.0 platforms, the container is constantly changing. When a user writes a Facebook post, it is displayed on the author's profile alongside their other posts; and in a reader's news feed, in which case the post will be surrounded by other posts of relevance to the reader; and could be shared and commented upon by a different Facebook user altogether; or displayed on a third-party page using Facebook's API. The same is true for the majority of popular Web 2.0 platforms. User generated content is stored without form on a server, and only when retrieved from the server and displayed is it given form and placed in context with related information. The impact of this is that users usually have limited control over how that content is displayed or even who will see it.

In many ways, the simple submission interfaces for user-generated content extend from the What You See Is What You Get (WYSIWYG) HTML editors that became popular in the late 1990s. Designers who learned to code using WYSIWYG editors did not need to learn about code, but instead were faced with a variety of toolbars and a visual representation of what the Web site would look like to users. These tools made Web development accessible to non-coders, but the resulting code was often overly complicated, was inaccessible to people with disabilities, and conflicted with ideals that the Web would become interoperable. WYSIWYG editors were "also seen to detract from the caring, craft-like approaches to Web design that some practitioners were developing and advocating. It was much harder to be dedicated to doing a job well, to pursue excellence and to benefit from some of the other internal rewards of Web design practice with tools that prohibited doing good work" [25]. Essentially, these editors automate some conceptual aspects of structuring Web content — *e.g.*, writing code and choosing how to arrange elements of the page — and leave their users to execute more mechanical tasks such as choosing options from a menu or filling out forms.

For further illumination of how certain tools can prohibit *good work*, I turn to Franklin's (2004) description of technologies that divide production into discrete processes as *prescriptive technologies*. By separating form and content, both WYSIWYG editors and Web 2.0 applications exemplify prescriptive technologies. The advantage of these technologies is that they allow people to participate in complex endeavours with great efficiency. In the case of Web 2.0, people from around the globe can communicate in intimate or broad scales with far lower barriers than previously existed. This is a profound benefit. But the moral concession of packaging this global communication via prescriptive technologies is that "prescriptive technologies eliminate the occasions for decision-making and judgement in general and especially for the making of *principled* decisions. Any goal of the technology is incorporated *a priori* in the design and is not negotiable" [26]. WYSIWYG editors constrained opportunities for doing good work by limiting decision-making about the architecture of a Web site to choosing between a set of options chosen by the tool's creators. By presenting Web creators with a limited set of options for accomplishing their work, such tools bear resemblance to managers who dictated work processes to factory workers in Braverman's (1998) classic text on the degradation of work in the twentieth century. In both cases there is a distinction between artisanal work where experts craft goods with control over the entire process, and assembly work where less expertise is required because the process

is laid out in simple steps. Social media and other user-generated content platforms amplify this tenfold. Users can labour over tasks such as choosing words carefully or skillfully editing a video, but decisions about code, a vital component of the appearance and structure of Web content, have already been made for them. For many who are not interested in learning to code, the ability to post online with all the coding decisions already taken care of by some platform is a boon. But nonetheless this comes at a cost. What is troubling is that this cost is made invisible by systems that present it as natural. Studying the infrastructures that make this arrangement possible can help illuminate their costs and contribute to an understanding of how and why it came to pass.

Conclusion

As described earlier, Sandvig (2015) argued one of the reasons for the shift toward asymmetric network distribution was that, while many readers want to access information from content producers such as the *New York Times*, the *Times* does not need to retrieve an equivalent amount of information from each reader. So the *Times* and other major media organizations benefit from having powerful infrastructures for delivering content and most users have little need of such systems. In contrast, many popular discourses of Web 2.0 promised an alternate scenario where users would share content with each other and leave professional media organizations out of it.

Ironically, while the *Times* might not need to consume articles written by its readers, the business models of Web 2.0 revolve around capturing, analyzing, and selling information retrieved from users. Albeit, the balance of traffic is still weighted so that users download more data than they upload. Users increasingly download large files such as videos, and the information collected by Web 2.0 companies usually comes in the form of small bits of metadata such as location and interactional information. The transition to centralized network distribution infrastructures serves the interests of Web 2.0 companies because 1) corporate providers can generally afford to provide higher speeds than individuals or small competitors; and 2) centralized distribution infrastructures increase their ability to act as mediators in many to many communications. In this mediating role, Web 2.0 platforms have deployed fast, easy-to-use Web applications that are tightly integrated with flexible, accessible data structures. This encourages a style of participation where users frequently contribute simple content, and where that content and other user information can be captured and added to growing data collections.

Earlier in this paper, two stories of Web 2.0 were discussed. As a social phenomenon, Web 2.0 was about amateurs "beating the pros at their own game" (Grossman, 2006), and as a technical and business phenomenon it was about companies capitalizing on the richness of shared data (O'Reilly, 2005). Web applications can make many to many communication appear simple, but much is going on behind the scenes. When a citizen journalist beats the *Times* to a scoop, when an amateur musician becomes an Internet star, or when millions of people spend their leisure time tweeting instead of watching television, their ability to beat the pros (media companies) at their own game depends on the support of a different set of pros (Web platforms). These platforms may facilitate the creation of content by offering easy-to-use tools, and as described above this involves a balance between ease-of-use and creative autonomy. Moreover, it is almost entirely with the aid of platforms that users are able to distribute content to large audiences. Serving Web content to millions of people does not happen on its own, especially when audiences expect it to load quickly. Instead, amateur content creators rely on platforms with sophisticated distribution networks like Facebook, YouTube, or Twitter.

My purpose in this article is not to promote a pessimistic view of the contemporary Web as a bad deal for users. Albrechtslund (2008) was correct to argue, "It is important to not automatically assume that the personal information and communication, which online social networking is based on, is only a commodity for trading". The technological and business strategies deployed by Web 2.0 platforms have truly created new opportunities for participation and communication. These opportunities should be celebrated, but attention should also be paid to their boundaries. To understand the bargain that has been struck between users and platforms, it is necessary to study how the Web's underlying networks, standards, and other components perform significant work to make these boundaries appear seamless and natural, to the extent they appear at all.

About the author

Jack Jamieson is a Ph.D. student in the Faculty of Information at the University of Toronto.
E-mail: jack [dot] jamieson [at] mail [dot] utoronto [dot] ca

Acknowledgments

This research was supported by the Social Sciences and Humanities Research Council of Canada.