

We live in a highly networked world. Billions of people engage and interact with one another online. Never has the world seemed smaller. Global communication is instantaneous. We can converse across the world, learn about other cultures and meet new people without leaving our own home. The ways in which people get online are as varied as the people who populate the internet. People connect on social media, communicate using applications, build websites using everything from hand-coded CSS and HTML to content management systems.

The web is built on free and open source software: server-side software like Apache and Nginx, application frameworks like Ruby on Rails, databases like MySQL, and security cryptographic protocols like OpenSSL. This software is often built by hackers and hobbyists, who distribute it freely, hack on it, patch it, and share it with one another. But free software isn't just the domain of hardcore coders -- those who build the underlying fabric of the web. Millions of people, for example, use tools like the open source browser Firefox to browse the web.

And then there is WordPress, the user-focused publishing platform built to make it easy for anyone to get online. At the time of this writing, it powered almost 24% of websites, and is used by millions of people worldwide. Users vary from casual bloggers to international news sites, small businesses to big businesses, charities, non-profits, networks, and personal portfolios. What started out as just a blogging platform is now the most widely-used content management system on the internet and is even used by some developers as an application framework.

With so much of the internet powered by WordPress, it's no surprise that the people who use it are as diverse as those who use the internet itself. Each website represents a different story, a different individual or organization who has a stake in being online and in providing a window into their world. This book tells the story of WordPress, a tool created by a couple of hackers and developed by volunteers from all over the world. But before that story begins, it's worth looking at the stories of just a few of those who have been empowered by the software.

Fighting Human Rights Abuse *Thailand, near the Burmese Border*

Burma is the stage for one of the longest-running civil wars in history. Since 1948, different ethnic minorities have struggled for independence. The military junta has committed innumerable atrocities against the people of Burma, particularly in ethnic areas: torture and executions, rape, kidnapping, and chemical attacks on citizens. It has left millions of people living like refugees in their own country, displaced from their homes and under attack from their government.

Since 1997, the Free Burma Rangers (FBR) have delivered humanitarian relief to internally displaced persons (IDPs). Operating from regions around Burma's borders, relief teams of up to six people enter Burma's jungles to provide emergency medical relief, humanitarian assistance, and counselling to IDPs. The teams are made up of Burmese nationals, many of whom have themselves been displaced, who themselves have been victims of the government's military rule.



Medics from the Free Burma Rangers in the field

In addition to humanitarian relief, the teams gather documentary evidence about atrocities and human rights abuses in Burma. Every team has a member whose job it is to document atrocities using photography, video, and interviews. This evidence is trekked out of the jungle, over the border, and forms an archive of the human rights abuses committed by the junta.

For many years, these documents were simply stored on paper. Offices full of paper held evidence of years of abuse. The information was tabulated into spreadsheets and documents. Photographs and videos were stored on stacks of hard drives. To find information, a person had to spend hours looking for it. There was no way to look at the data in aggregate, to find patterns and trends.

With the help of a team of volunteers, the FBR's hundreds of thousands of records were converted into a digital database that allows rangers to quickly upload and disseminate information. The system needed to be usable by the rangers, many of whom had little computing experience, and many of whom speak different languages and dialects. When the project got to the point of user testing, the site was tested in a hut in the jungle on a laptop plugged into a solar cell, with a satellite internet connection.



User testing in a hut in the jungle

Reports are stored on two sites -- an internal site for storing and tracking all of the reports, and a [public-facing website](#) which is used to disseminate information to the public. Both of these websites are built using WordPress. While on a relief mission, a

team takes notes on human rights violations, documenting the location, taking photos, and recording interviews. They do the same thing for any patients they see, recording everything in a paper journal. When the relief mission is complete, the journal and SD card are trekked out of the jungle and the information is added to the website.

The digitization of all of these reports has meant that volunteers can spend less time digging for information and more time improving relief, training, and providing aid. It's made analysis more efficient and effective, allowing training teams to look for patterns that will improve aid.

But it's also helped the ease with which the FBR can disseminate information about atrocities to the world. In December 2012, one of the IDPs working with the Free Burma Rangers hiked through the jungle to deliver video and photographs of the Burmese military using attack helicopters and jet fighters in air strikes against ethnic rebels in the northern Kachin state. The images were [published on the blog](#). The next day, the [BBC picked up the images](#), broadcasting them around the world. These images increased international pressure on the Burmese military and played a significant role in the 2013 ceasefire.

Charting a dream journey *On the road, USA.*

For nearly five years, Maria Scarpello has been traveling the highways and byways of America, sampling beer, in search of that perfect pint. She travels in an RV she calls Stanley, a 1999 Class C Ford Jayco. Stanley isn't just an RV, it's her home. "Home is where my wheels are," says Maria. She parks the RV wherever she can, from campgrounds to street corners, from parking lots to friends' driveways. Stanley has been adapted to her life on the road. The bed has been removed from the back of the RV, replaced with a couch to give additional space to work, relax, and hang out with her two dogs Ernie and Buddha.

Her first blog, [TrippingWithStanley.com](#) charted the first year-and-a-half of her journey across America; from zip lines in Las Vegas to golfing in Phoenix, barren deserts in Arizona, to the rocky cliffs of the Oregon coast. When Maria left home, she had planned to travel for only six months, but the five years she has lived in her RV are a testament to how deeply she has fallen in love with life on the road.



Maria at Golden Canyon in Death Valley National Park

Her current website, [The Roaming Pint](#) is a travelog of all of the breweries that she has visited in America. Through the site, beer lovers can vicariously visit America's breweries and search the website to get information about the hundreds of breweries she's visited.

To support her trip, Maria had to find work that would work with her fluid lifestyle. To begin with, she planned to make money through her blog -- plenty of people fund their travels by blogging, so why not? However, making money purely from a blog is no easy task, and after a year on the road Maria decided money via the blog was not sustainable enough to solely support the lifestyle she wanted to maintain. For the past three years, Maria has worked at WordPress company WooThemes, as the Internal Community Manager. Armed with a mobile hotspot and her laptop, she has everything she needs to make money while she is on the road. Her office is the RV. It's been completely renovated to make it feel like home: the walls are painted bright green and are decorated with stickers, coasters, pictures, and other souvenirs from her travels. The view from her office is always changing. "Next month my office will be overlooking the beautiful white capped Rockies in one of my favorite states -- Colorado," says Maria "the following month we will be venturing even further west, eventually making our way back to the coast. There is no better office view for me than overlooking vast bodies of water, preferably with palm trees or mountain side cliffs and sandy beaches begging me to put my MacBook Air away and let the dogs play!"



Maria hanging out with her dogs and Stanley (by [sethkhughes.com](#))

Running a Small Business Fjærland, Norway.

Eivind Ødegård arrives at his office and heads straight for his coffee machine. Coffee in hand, he sits down at his computer and starts reading email. He is surrounded by books: they fill the shelves that line the walls of his room, they are stacked all over the floor. Much of his day is spent in front of the computer, handling money and bills, dealing with correspondence, and processing orders. It sounds like a job that could be done in any office, but his view is quite different from the normal nine-to-five. From his window he can see the ferries that bring tourists in and out to the tiny town of Fjærland. They come to Fjærland for the glacier, the majestic scenery, and [Norway's booktown](#), of which Eivind is the manager.

Fjærland is a rural community with less than 300 inhabitants. It is ringed by mountains and glaciers. Waterfalls run down the mountainsides along the fjords where dolphins and seals hunt for fish. In the summer, it is hot and sunny and busy with tourists; in the winter, the weather turns cold and the local inhabitants -- cut off from the rest of the world -- plan for the coming summer.



The booktown at Fjærland (photo credit: Trond J. Hansen.)

The booktown that Eivind manages was a local initiative to reinvigorate the town. A [booktown](#) is "a small rural town or village in which second-hand and antiquarian bookshops are concentrated. Most booktowns have developed in villages of historic interest or of scenic beauty." The first was in Hay-on-Wye in Wales, but currently there are booktowns all over the world. In Fjærland, the booktown was established in 1995 to solve a specific problem. Buildings in the town were derelict and falling into disrepair. Rather than watch the town fall into ruin, the townspeople filled them with books. Second-hand books fill stables, boathouses, a bank and a post office, a grocery shop, and a ferry waiting room. Each shop has its speciality, one even doubles as the tourist information office. Some bookshops are staffed, others operate by an honor system - customers leave money in an honesty book whenever they take a book.



A customer browses the books (photo credit: Trond J. Hansen.)

Some of Eivind's time every day is spent managing and updating the [booktown's website](#). The website is a window for the rest of the world into this secluded community. The site provides information about the booktown, while the blog has news from Fjærland. There is also an online store from which the booktown distributes publications from Sjørettsfondet, the [Norwegian Maritime Law Foundation](#). The website, maintained in both Norwegian and English, is built on WordPress.

Building the Web New York, New York

Some blogs have been around long before blog software even existed -- long before WordPress or Movable Type or Livejournal. People shared their thoughts and links from around the web on these hand-coded websites. These blogs chart not just the evolution of a person, but the evolution of the web. Jeffrey Zeldman's blog is one such example. It's a record of many of the important trends and changes in technology and design that have happened ever since the mid-1990s, complemented by asides and personal reflections that illustrate the world as it is has changed along with the web.

Before the advent of blogging as its own form, Jeffrey used his website to entertain and create. His archives are a treasure-trove of 1990s design artefacts: [banners from zeldman.com](#), [desktop backgrounds](#), and 8-bit [pixel art icons](#).

Reading through the blog archives is like touring the history of the modern web. It chronicles the early days of web standards, when many designers were building beautiful sites with Flash and others were crafting web pages with CSS, through the popularization of the semantic web, from the growth of social networks and the skepticism around them, to the introduction of responsive web design. These changes in the web are woven through the blog, with personal comments on them, from Zeldman's own particular perspective.

But there's more to Jeffrey's blog than web musings. Even while he was still hand-coding his website, he started a section called "My Glamorous Life," where he [wrote about himself](#). The tag "glamorous" is still what he uses to tag posts that are about him, rather than about someone else. Woven between posts about the web are posts about Jeffrey himself, and as with all diarists, there's a lot we can learn about him. He's a web designer who cares passionately about web standards. He's a publisher and an event organizer. He hosts a regular podcast called "The Big Web Show." But there's more than just that, more than just what he *does*. He has a daughter, Ava. He's divorced, though long after the divorce he still looks after his ex's two small dogs when she is ill (much to the annoyance of his neighbours). He has family secrets that, now and again, come out of the closet. He's a fan of Edward Hopper and John Coltrane. He surrounds himself by smart people he respects. He loves what he does and he wants you to like it too.

This blending of the personal and the professional is the mark of many blogs. After all, for the writer, these things are just different aspects of who they are. "I don't really see a big difference between sharing what's personal and sharing opinions about web design and sharing information about web design. It's all the same," says Jeffrey. He continues in the same fashion

today, blogging on WordPress about the web, sharing news about his different projects, and telling stories from his glamourous life.

Giving People a Voice Worldwide

All over the world, citizen journalists tell their stories on the web. The past fifteen years have seen publishing tools go from the hands of a few media outlets and into the hands of the many. Those who traditionally formed the audience, are now able to participate in the process of reporting the news. Individuals share their stories and unique perspectives from locales all over the world.

Global Voices Online is one of the largest international networks of bloggers and citizen journalists. Using WordPress, the team of writers reports from 167 countries, and their stories are translated into more than 30 languages. Founded in 2004 by Ethan Zuckerman and Rebecca MacKinnon with the aim of highlighting bloggers who build bridges between languages and cultures, its mission has evolved to "find the most compelling and important stories coming from marginalized and misrepresented communities," as well as speaking out against online censorship and helping people find new ways to gain access to the internet.

"People no longer only get their news from governments and the media, but have a wider variety of sources to enjoy. Especially when it comes to societies that have limited freedom of expression that's hugely important," says [Solana Larsen](#), former managing editor. "Ordinary citizens can help change the way current events are perceived and remembered in history."

Since its launch, bloggers on Global Voices Online have written about some of the most significant events happening world. In March 2011, bloggers in Japan [wrote with horror](#) about the 8.9 magnitude earthquake that resulted in over 10,000 deaths and damaged the Fukushima powerplant. The same year, [bloggers at Global Voices Online](#) were among those in Tunisia who provided eyewitness accounts of the revolution that toppled the government. In Egypt [bloggers wrote about the reolvention](#) that led to the downfall of Hosni Mubarak. In 2014, bloggers from West Africa wrote about the Ebola crises which has swept Liberia, Guinea, and Sierra Leone. Bloggers write to remind the world that they are more than just a virus, to highlight the work of scientists and medics, and to share [successes in the fight against the disease](#).



A resident walks past a mural about the dangers of the Ebola Virus painted on a wall off Tubman Boulevard in Monrovia (photo credit: [Morgana Wingard/ UNDP cc-by-nc-nd](#))

But it's not just the world-shaking events that get written about, bloggers write about what it's like to live in their countries, sometimes from repressive regimes where voices are so easily silenced. In Russia, a blogger writes about how the internet has become a battleground for LGBT rights. In Skopje, Macdeonia, shoppers protest against [government plans to makeover buildings in a neoclassical or baroque style](#). Syria prepares for its first TEDx conference. In Ecuador, the Guaycan trees come into bloom, flooding the world with yellow. Each of these stories is a window onto another part of the world, giving the reader a

glimpse of what it is like to live somewhere else, in a different set of circumstances.



The Guayacan trees bloom in Ecuador (photo credit: amalavida.tv cc-by-sa)

These are just a few of the stories of those who use WordPress, people who have been able to tell their stories and share their own unique perspectives on the web. But the software itself has a story, one that stretches back even before WordPress itself was launched. Like so many pieces of free software, It didn't start out with financial backing, it wasn't built by a company with aims and objectives. It was a small discarded blogging script that was picked up by two hackers who wanted to use it to power their blogs. So how did it come to dominate the web? Why is it so popular? And who are the people who made the software what it is today? This book charts the story of WordPress, from its humble beginnings as a few lines of code written in an apartment above a bar in Corsica, to the dominant web platform that it is today.

PART 1

1. Hello World

February 2002, in Houston Texas, Matt Mullenweg ([matt](#)) was home from school. Sitting at his homemade PC, surrounded by posters of his favorite jazz musicians, he downloaded a copy of Movable Type, installed it on a web server, and [published his first blog post](#).

I suppose it was just a matter of time before my egotist tendencies combined with my inherit geekiness to create some sort of blog. I've had an unhealthy amount of fun setting this up. This will be a nexus where I talk about and comment on things that interest me, like music, technology, politics, etc. etc.

A few months later and nearly five thousand miles away, in Stockport, England, Mike Little ([mikelittle](#)) sat in the converted cellar that served as his home office. Surrounded by hundreds of books and CDs, he sat at a desk swamped by a 17" CRT monitor and spent his Sunday installing a free and open source blogging platform called b2. [His first post](#) is not unlike Matt's. Mike tells the world about his blogging platform and his plans for his blog. "There will either be nothing here," he wrote, "or a collection of random thoughts and links. Nothing too exciting. But then again I'm not an exciting person."

These two unremarkable blog posts, *testing new system* and *welcome*, are like hundreds of first blog posts before them. A tentative first step, a test of the software, a pronouncement that "I am here" with a promise of writing to come. What sets these two posts apart, is that they don't just mark the beginning of two blogs, but of a blogging platform. A blogging platform that would support a community and an economy, and that would enable millions of people, all over the world, to take that first step too, to write their "hello world," and take their voices online.

Matt and Mike published their first blog posts at a time when more and more people were getting online and using the internet to express themselves. Blogging, while not quite in its infancy, was still maturing. A few years earlier, in 1998, there had only been a handful of weblogs. These early blogs were often curated collections of links accompanied by snarky or sarcastic commentary. These collections provided a filtering function. The author pre-surfed the web for readers who could then browse through links on weblogs that they trusted. One of the earliest bloggers, [Justin Hall](#), collected links to some of the darkest corners of the internet, but in addition to sharing the weird things he found, Hall poured his personal life online, and was a key figure in the transition from link log to personal diary.

As [Rebecca Blood noted in 2000](#), blogs evolved from "a list of links with commentary and personal asides to a website that is updated frequently, with new material posted at the top of the page." It was this type of blog that brought Mike and Matt online, a format with established conventions that we're familiar with today. Blogging software publishes content with the most recent at the top of the page. When a visitor lands, they see the latest post. This premise of self-publishing has been a feature of blogging from the earliest online diaries to weblogs to tumblogs and even to microblogging sites such as Twitter. When we see the most recent content first, we instantly recognize a blog.

Many of the first bloggers were those already involved with the web, software developers like Dave Winer, designers like Jeffrey Zeldman, and technologists like Anil Dash. As a result, much of the earliest blog content was about the web and technology, interspersed with more diffuse thoughts and commentary about the blogger's life. With this tendency toward meta-commentary, bloggers wrote about the tools used to publish their blogs and the improvements that they made to their sites. While blogging grew steadily, the community was still considered geeky and insular. [Journalists spoke disparagingly of bloggers](#) and rarely took their reporting seriously.

In 2001, blogging started to permeate public consciousness. American political blogs became popular in the wake of the

September 11th attacks on the World Trade Center and the Pentagon. Blogs such as [Instapundit](#) and [The Daily Dish](#) saw a massive surge in popularity. Andrew Sullivan, the blogger behind [The Daily Dish](#), said in an article in [The Atlantic](#) that people didn't come to his blog just for news; "They were hungry for communication, for checking their gut against someone they had come to know, for emotional support and psychological bonding."

This was one of the first indications of the power of blogging -- it gave people a voice online, providing a platform where people could come together to grieve. In his book *Say Everything: How Blogging Began, What It's Becoming, and Why it Matters*, Scott Rosenburg talks of how, despite the burst of the dot-com bubble, the final quarter of 2001 saw an increase in blogging. "Something strange and novel had landed on the doorstep," Rosenburg writes in his introduction, "the latest monster baby from the Net. Newspapers and radio and cable news began to take note and tell people about it. That in turn sent more visitors to the bloggers' sites, and inspired a whole new wave of bloggers to begin posting." Since that time blogging, and later social media, has played a pivotal role in politics, public life, and even revolutions.

As blogging evolved, so too did blogging tools. Services like Geocities and Tripod allowed anyone to create a website, but these sites had little in common with the dynamic stream of content on a blog. The earliest blogs were created manually, using HTML and FTP. Justin Hall, on his weblog, had a page titled [Publish Yo' Self](#) which taught people how to write HTML and publish online, claiming that "HTML is easy as hell!" But writing and publishing HTML got in the way of the actual writing process; the dream was to create a tool for writing content and publishing it to the web with one click. Dave Winer, whose popular [Scripting News](#) was one of the earliest blogs, set up UserLand Software which developed [Frontier NewsPage](#), a tool that enabled people to create news-oriented websites like [Scripting News](#).

In 1998, Open Diary, an online community where people wrote their diary online and communicated with other diarists, launched. Then came [LiveJournal](#), Xanga, and [Pitas.com](#) in 1999. In the same year, Pyra Labs launched [Blogger](#), the tool often credited with popularizing blogging. Blogger automated publishing a blog to a web server: the user wrote their content, and Blogger uploaded the page to the server after each post.

Movable Type, the platform which, at one time, was one of WordPress' biggest competitors, launched in 2001. As with so many popular platforms, Moveable Type came about as a result of developers scratching their own itch. While Blogger was easy to use, it was limited in its functionality -- it lacked post titles, rich text editing, and categories, for example, so Ben and Mena Trott set out to create something different for [Mena's blog](#). In October 2001, when they launched Movable Type it quickly became the most popular blogging platform, used by many of the major blogs at the time, including [Instapundit](#), [Wonkette](#), and [Boing Boing](#). In many ways, Movable Type raised the bar for blogging platforms. It was not simply a publishing platform, it was a publishing platform that people wanted to use. "I thought it was beautiful. I think in a lot of ways it foreshadowed the web 2.0, not the gradients and things, but the beauty and the white space," recalls [Anil Dash](#). Now, "all of a sudden my blog posts had titles and I could have comments and I could archive things by month and I could do all manner of really interesting things. And all mostly with just HTML. Pretty much as easily as Blogger, but with just so much more power."

The tools got better. Soon publishing content wasn't enough. Bloggers pushed at the boundaries of what their blogs were for. Communities grew around different types of blogs. An author's blog became a way for them to connect with people around the world. A blogger's first, lone "hello world" quickly evolved into a series of posts that interlinked with other bloggers across the internet. Sidebars were embellished with "blogrolls," lists of blogs that gave "link love" to favorite sites. Trackbacks were developed by Movable Type as a way of letting a blogger track the discussion of their articles across the internet.

It was into this arena that the nascent WordPress platform first said "hello world." It did so quietly, not in Texas, or in Stockport, but in the French island of Corsica. It appeared, first of all, not even as WordPress at all. In June 2001, months before Mike or Matt had even published their first blog posts, a developer in Corsica started [development of his own blogging platform](#). He touted it as a "PHP+MySQL alternative to Blogger and Greymatter." He called his PHP blogging platform b2.

2. The Only Blogger in Corsica

In a small apartment above a bar in Corsica, in late 2000, Michel Valdrighi, ([michely](#)) wrote a blog post. The "only blogger in Corsica" used a creaking dial-up AOL connection that disconnected every thirty minutes. He shared his apartment with his cats, No Name and Gribouille, who perched on a windowsill that looked out over a high drop and [whose antics he captured on his blog](#).

Like many bloggers, he experimented with different web publishing platforms -- he started out with HTML before moving on to Blogger. And, like many bloggers, he found that Blogger wasn't as fully-featured as he'd wanted. It didn't have a built-in comment system, for example. This was at a time when bloggers could sign up for external commenting services but these services were often unreliable and unstable. Michel signed up for two different commenting services, both of which disappeared along with all of the comments and discussions. Blogger was also plagued by stability issues. At the time, bloggers joked that Blogger was "sometimes up."

Michel was learning the server-side scripting language PHP. PHP is used to retrieve information from a database and display it in a browser. Unlike a language like Perl, PHP is relatively easy to learn, making it a useful tool for people who want to start hacking on software. One of Michel's first PHP projects was a Corsican-language dictionary: this experience taught him that he could use PHP to manipulate data, inspiring him to create his own blogging script. In June 2001, [Michel started developing b2](#) [^fn-1], a "PHP+MySQL alternative to Blogger and Greymatter." He wrote:

Not much new ideas in it, but it will feature stuff like a built-in comment system, good users management (with complete profile etc), user-avatars (got piccies ?), multiple ways of archiving your blog (even post by post if you like to do a kind of journal) (sic).

The installation will be easy, just edit a config file, upload everything and launch the install script. And there you go, but you'd like an admin control panel? It will be there, packed with options.

The PHP and MySQL combination was a good alternative to other contemporary blogging platforms. When a person visits a PHP/MySQL website, the script runs and PHP retrieves information from the database. PHP is suited to the dynamic nature of a blog, in which an author regularly posts content and readers regularly return to read it. In contrast, Movable Type, which was built using Perl, rebuilt the page every time someone left a comment or edited the page, often making for slow page-load times. b2 was touted as an easy-to-use, speedy blogging tool. "You type something and hit "blog this" and in the next second it's on your page(s)," [boasts the sidebar of cafelog.com](#). It goes on, "Pages are generated dynamically from the MySQL database, so no clumsy 'rebuilding' is involved. It also means faster search/display capabilities, and the ability to serve your news in different 'templates' without any hassle."

These features attracted Matt, Mike, and so many other bloggers. But Michel admits to being a novice developer, learning PHP and MySQL while building the precursor to one of the most widely-used blogging and CMS platforms in use today. He [recalls now](#) that as a fledgling PHP developer he didn't always do things correctly:

When you look at WordPress's code and think 'Wow, that's weird, why did they do that this way?', well often that's because they kept doing things the way they were done in b2, and I sucked at PHP.

Like other early blog software developers, Michel was scratching his own itch. "Scratching your own itch," is a familiar phrase in free software development. This means creating the tools you need, whether that's a blogging platform, a text editor, or an operating system. In his book, *The Cathedral and the Bazaar*, Eric Raymond [writes that](#) "Every good work of software starts by scratching a developer's personal itch." If a developer has a problem, they can write software to solve that problem, and then distribute it so that others can use it too. Many blogging platforms, including Blogger, b2, Moveable Type, and WordPress, started life in this way.

By being both the developer of the software and its user, the developer knows whether the tools they create meet their real needs. The developers are the users. When they write code, they use it just as any user would. They know what works and what doesn't. Scratching an itch gives the developer a closer relationship with the software's users -- it's grass-roots development -- bottom-up as opposed to top-down.

Michel wanted a blogging tool, not a CMS, so early b2 features were focused on creating a frictionless way to blog. Early enhancements made writing easier. [AutoBR](#), was added, for example, to add a `
` tag to create a new line every time a writer hit enter. As Michel's, and later the community's needs changed, improvements included [a basic templating system](#), an [options](#) page, [archives](#), an option to allow users to set their own timezone, and [comments](#).

Michel wanted b2 to be usable by everyone, from experienced developers to writers with little technical experience who just wanted to publish their content online. He [writes about “Joe Newbie,”](#) a persona he had in mind. Michel believed that the lowest common denominator should be able to easily publish their blog. This ideal of "making software easy for everyone" permeated the b2 community and would eventually become a core ideal in WordPress' own philosophy.

By the end of June 2001, Michel was [ready to say goodbye to Blogger](#) and move his blog to b2. His blog was the first website to run on the code that would become WordPress, grandfather to millions of websites and blogs. A few days later, Michel set up a website called [cafelog.com](#), and [released the first version of b2](#). The software was quickly picked up. [The second site running b2](#), the personal weblog of a schoolboy called Russell, was published in July 2001.

The screenshot displays the b2 blogging platform interface across three main sections:

- Post Editor:** The top section shows a form for creating a new post. It includes fields for "Title" (with a placeholder "My First Post"), "Category" (set to "General"), and a rich text editor toolbar with buttons for bold, italic, underline, strike, and various block elements like `<p>`, `<i>`, ``, ``, `b-quote`, `image`, and `link`. Below the editor are checkboxes for "Auto-BR" (which converts line-breaks into `
` tags), "Blog this!", and "Spellcheck". There is also a timestamp selector for "Edit timestamp" with a dropdown showing "28 November 2014 @ 10:10:15".
- Archive List:** The middle section shows a list of posts. It includes a search bar, a date range selector ("Show posts: 20 last posts OK" and "0 to 20 from the end OK"), and a "View" button. The list shows one post: "2013/07/16 @ 13:03:38 [1 comment - Edit - Delete] Hello world ! by admin (admin), in General". A note below says "This is the first post. Edit or delete it, then start blogging !".
- Footer:** The bottom section contains a footer bar with links for "b2 0.6.2.1 - ©2001-2003 michel v - tidakada.com".

The b2 user interface

Development didn't always go smoothly, although solving problems inevitably led to platform refinements. Brigitte Eaton, who ran a site called [Eatonweb](#), was the first prominent b2 user. Eatonweb was a hand-maintained list of blogs. It was a go-to place for discovering new blogs until the number of blogs exploded and hand-maintained lists became untenable. It was superseded by the ubiquitous blogroll and services like Technorati which started ranking blogs according to incoming links. Brigitte Eaton published regularly. When she imported her blog into b2 she found that the more posts she added, the slower the blog loaded. Michel did not know how to write code for retrieving months from the archives. His code parsed through every post on the blog to query whether the post had changed and then displayed it. For blogs with large numbers of posts, this meant a huge server workload, which quickly slowed the website. Solving this problem meant improvements for all b2 users.

Although b2 did not have an established developer infrastructure, it was open for contributions. The first major code contribution to the project was [pingback functionality](#), from developer [Axis Thoreau](#), who went by the handle Mort. [^fn-2] Movable Type had a similar linkback feature called trackback, which [Michel included a version of in b2](#). Trackback is a ping sent from the original

author's site to a site they reference with a link. Unlike pingbacks, trackbacks are manual so the original author has to choose to send the trackback, and can edit the excerpt sent. Trackbacks are much more susceptible to spam than pingbacks. [^fn-3] Whereas pingbacks ping back to the original site to verify that it is not spam, trackbacks do not carry out the same checks.

b2 was Michel's pet project but with users and developers frequenting the forums, a community quickly formed. The forums were a haven for people who needed help with the software. The most popular were the installation, templating, and how-to discussion forums, but there were also forums just for general chatter about the software and about blogging. Developers helped each other out with hacks for b2, and it was on the forums that the early WordPress developers first met.

Outside the b2 community, the software was criticized. Michel's lack of coding experience showed, especially to experienced developers. Blogger Jim Reverend wrote a post titled "[Cafelog: A Look at Bad Code](#)." In it Jim bemoans poor code quality in publicly released software, particularly b2. He criticizes the software for its lack of features [^fn-4] and poor coding practices.

Michel's lack of PHP experience meant that he wrote inefficient code and used techniques counter-intuitive to a more experienced coder. Rather than taking a modular approach to solve a logic problem, the code grew organically, written as Michel thought about it -- a sort of stream-of-consciousness approach to coding. Code wasn't so much a tool to solve a problem, but a tool to get the newest feature on the screen. This created multiple code interdependencies which caused problems for developers new to the project. A line of code would change, and break something that appeared unrelated. For Reverend, this type of "dirty code" was unforgivable; "What I do have a problem with," he wrote, "is my inability to use his code (without extensive reading and rewriting) to implement my own features." Writing extensible code is a fundamental principle of free software. If code is written in a stream-of-consciousness fashion, lacking in best practices, it becomes difficult for other developers to extend.

It wasn't all bad. [In a follow-up article](#), Reverend concedes that there are benefits to such poorly written code:

While programmers like myself start convulsing when we have to wade through such code, less experienced programmers actually like it, because it is easier for them to work with. Because the "template functions" are in the global namespace, they work anywhere. Because the data returned by the database is in a global variable, you don't have to use any tricks to get at it. It makes extending, enhancing, and modifying the code easier for newbies.

Michel's inexperience gave the code a level of simplicity that made it easy for other novice developers to understand. Whereas a more experienced developer might write complex code, Michel took the simplest route to solve a problem. "In a way it was beautiful because it was so simple," [says developer Alex King, \(alexking.org\)](#), "It wasn't elegant but it was very straightforward and very accessible. For someone who didn't have a lot of development experience coming in -- like me -- it was very comfortable understanding what was going on."

The article also highlights something important: people liked using b2. With b2 they could publish without hassle. "If you are a user of this product," writes Reverend, "please don't tell me about how cool it is, or about how well it works. If you read a site powered by this engine, please don't tell me about how easy you find it to use."

But people did find it easy to use. To users it didn't matter what was going on under the hood. It may have had its problems but it was a frictionless way of getting content online. Where users went, developers followed; even better if those users were novice developers themselves, fumbling at the edges of PHP, learning what they could do with code, which new features they could add to their website and share with other users. Even in these very early days, a schism started to open between developer-focused development and user-focused development. On the one hand a focus on logical, beautifully written code, on the other a focus on features that users want.

Despite having distributed b2, Michel prevaricated for some time over his choice of license. The free software movement was relatively young and many large projects had their own licensing terms (such as Apache, PHP, and X.Org). Until he chose a license, Michel distributed b2 with his copyright.

On the b2 blog, you can follow the chain of events that led to b2's distribution with a GPL license. In August 2001, Michel made a brief statement on cafelog.com, telling people that they could use his code provided he was given credit for it, stating explicitly that "b2 isn't released under the GPL yet." People were taking notice of b2 and some passed it off as their own. In October of

2001, a Norwegian agency claimed to own the copyright to b2 and Michel was forced to contact the Norwegian copyright agency. In the discussion around this incident, Michel made a statement that came as close to a license as he'd had so far:

You can use b2 for free, even if your site is of commercial nature. You're welcomed to buy me items from my Amazon.com wishlist if you're going to make much money from your b2-powered site or if you just like b2.

You can edit b2's source code.

You can re-distribute a modified or original version of b2. In no way your modifications make you author or co-author of the modified version, I'll remain b2's sole author and copyright holder.

Any help is welcomed. Feel free to submit fixes and enhancements, they might get in b2's code and your name or email address will be there as credit in the source code.

I guess this makes a b2 license for now.

Michel released b2 under this slapdash license until he realized that b2 needed an official license and started looking in earnest. It was important to Michel that b2 remain free, even if he stopped working on the project. He also wanted his code to remain free if other developers took it and used it in their own project. He recalls now that "at the end of that elimination process, GPL remained. It helped that there were already some projects using it, as I didn't want the code to end up abandoned and forgotten because of the choice of an exotic license."

Michel's choice of license was prescient. Under a GPL license, software can be forked, modified, and redistributed. If development stops (as it did with b2), the ability to fork, modify, and redistribute can prevent software from becoming vaporware.

All was going well until May 2002 when [Michel lost his job](#). In the months following, he continued to develop b2, but he [struggled with depression and health issues](#). His electricity was cut off, he [moved house](#), and [struggled to find a job](#). With so much going on in his life, Michel eventually disappeared. He [posted about spam in December 2002](#) but didn't post on his personal blog in 2003.

The users of b2 had no lead developer. There was no one to steer the project, fix bugs, apply patches, or add new features. People were concerned about Michel -- they liked him and were worried about what had happened to him. In March 2003, a [thread on cafelog.com](#) discussed Michel's whereabouts. Michel's sister, Senia, posted that he was well and that he was looking for a job. She promised to ask him to connect to IRC and MSN. The responses to Senia were mixed.

One commenter said:

Please pass on to Michel that not only has he created a really nice piece of software, but he has also inadvertently built a community [...] of people, a sort of commonwealth on the blogosphere. I don't know him the slightest bit, but I wish him well and hope that any soul searching or vision quest (or vision exodus?) he has embarked on helps him find what he needs. Yet also allows him to tie up any loose ends that he leaves behind.

Others had concerns about their own projects:

Anyone heard from Michel yet? His last post was 6 weeks ago. I want to install and promote b2 in two projects (one affiliated with a UN women's project).. but I am nervous about doing so if there will be no developer support available. I am sure he knows that b2 is included in the Fantastico auto-installer bundled with the cPanel virtual hosting tool. I want to write an blogging article for hosting clients. Michel? You going to be around?

Michel never went back to b2 with the same gusto with which he had started. Eventually stagnation made the software unusable. Software needs to be maintained -- bugs need to be fixed, security issues patched, new features need to be added. Blogging software needs to evolve with a fast-moving internet. b2 was a good platform, but without someone to fix bugs and add new features it was quickly becoming vaporware. It wasn't just the software that had been abandoned -- the community was adrift too. In the free software world, the community is as important as the software. The community is the garden in which the software grows and matures. Community members submit patches, fix issues, support users, and write documentation to

help a free software project flourish. But every project needs a person, or group of people, to commit patches, decide on new features, and generally steer the project. When that person disappears without a trace, community members can play around the edges, helping out with support and making hacks, but, without someone to step up and take that person's place, the community dissolves. People move on to other projects, or start their own. After all, if the person who owns the project shows no commitment, how can commitment be expected from anyone else?

But the GPL license meant that neither the code nor the community had to disappear. Developers, familiar names from the b2 community forums, forked the software. While b2 itself did not continue, it was the platform that connected Mike and Matt, and the software that would provide the foundation for WordPress. With its simple PHP and focus on usability and ease of use, b2 contained the rudimentary ideals that would form the heart of WordPress. But first, the software had to be forked, and, as bloggers are wont to do, they took to their blogs to do it.

[^fn-1] The name b2 is a combination of the word "blog" and "Song 2" by the British band Blur, which Michel had been listening to regularly at that time. He combined them to make "blog2," then "blogger2," until he arrived at b2. b2 was also known as Cafelog, which was the name Michel had planned to give to the 1.0 series as no b2 domains were available, and the name was too short for a project on SourceForge.

[^fn-2] Pingback is a linkback method which authors use to request notifications when someone links to their website. An author writes a post on their blog that links to a post elsewhere. The original site sends a XML-RPC request and when the mentioned site receives the notification signal, it goes back to the original site to check for the link. If it exists then the pingback is recorded. A blogging system can then automatically publish a list of links in the comment section of the post, or wherever the developer chooses to place it.

[^fn-3] Spam pings are pings sent from spam blogs to get links on other websites.

[^fn-4] Particular problems that the article calls out include the fact that b2 didn't cache calls to the database so when there was a new visit from a user, the page had to be loaded from the database and server, slowing the website. The article also criticized the lack of flexibility in the templating system and the absence of cruft-free URLs (known in WordPress lingo now as "pretty permalinks").

3. The Blogging Software Dilemma

The internet brings people from different backgrounds, countries, and cultures together around shared interests. On a small corner of the internet, b2 was forming one such community. People supported one another because they were interested in two things -- blogging and blogging software.

The founders of WordPress were among those drawn into the b2 community. From very different backgrounds, free software formed their common ground. In Houston, Matt Mullenweg wrote about politics, economics, technology, and his passions for jazz music and photography. Mike Little, from Stockport, wrote about blogging technology, the books he read, and his family.

Matt and Mike had very different entries into computing: Matt first started tinkering with computers at an early age, through his father who was a computer programmer. Mike, twenty-two years older than Matt, had his first computer experiences at school cut short when a teacher caught him and his friends smoking in the paper room.

It was a shared passion for music, however, that drew them both deeper into programming. Mike was involved in the Manchester music scene in the 1980s, working at a record shop and handing out 'zines for nightclub manager and record label owner Tony Wilson. He lived with a local glam rock band, and it was through the band that computing came back into his life. The lead singer thought it would be cool to have stacks of televisions with graphics on either side of the stage. Mike borrowed a ZX Spectrum and used a couple of programs from computing magazines: one of them bounced objects around the screen, the

second created 3D text. He hacked on them until the words he wanted bounced around a screen, but the plan fell flat when they couldn't figure out how to output the program to multiple screens at the same time. Despite not being a total success, this incident further cemented his reputation for being the go-to guy for all things computing and music.

By the time he arrived at b2, Mike had nearly twenty years of programming experience. That first faltering step with the ZX Spectrum had started a love affair with coding, which took him on a route through Basic, 6502 Assembler, Pascal, and C. He learned PHP during the transition between PHP versions one and two.

In Houston, Matt set up his first business while studying at the High School for the Performing and Visual Arts (HSVPA). While Mike's music passion was post-punk, for Matt it was jazz. He built computers and websites for his music teachers, including his saxophone teacher, [David Caceres](#). He also used the internet to connect other music fans in the Houston area, setting up a [forum on David Caceres' website](#). Matt used the forum software phpBB -- his first taste of using PHP to create a dynamic website.

With experience in PHP, b2 was the perfect platform for publishing their content and for letting their hacker tendencies loose. Matt experimented with other platforms. He tried Movable Type but at the time it didn't have pingbacks and comments were in pop-ups as opposed to being inline. Movable Type was written in Perl with a DBD database, which meant that customizing Movable Type was more difficult than a PHP platform.

PHP and MySQL allowed Mike and Matt to scratch their blogging and hacking itches. They hacked on their websites and customized as they saw fit. They shared those hacks and improvements with the community. Matt and Mike's first discussion was around the gallery software Matt used on his blog. Other developers had their own itches to scratch. Developers were talking about building their own platforms. Others, in the absence of Michel, were considering a fork.

b2 was free software, licensed under the GPL; anyone could fork the code and make use of it, provided their fork retained the GPL license. By early 2003, it was clear that Michel would not be back. No one was maintaining b2, no one was fixing security issues. The blogging software at the core of the growing community was no longer evolving. The web, and blogging, was moving forward, but b2 had lost its driving forces. In France, Francois Planque [forked b2 to create b2evolution](#). The lack of b2 development frustrated Francois and he wanted to continue to develop b2 for his own needs.

In Cork, on the west coast of Ireland, Donncha Ó Caoimh, ([donncha](#)), forked b2 to create b2++, a multi-user blog platform. Donncha discovered b2 while searching for a platform to create a blog network for his Linux user group. He found b2 small, basic, and easy to modify. However, he made major modifications to [create blogs.linux.ie](#). b2++'s templating system used [Smarty](#), which separated code and presentation, making it easier for users on the network to change their site's design. [Donncha didn't consider b2++ a fork of b2](#). "A fork gives the impression that it was competing -- it wasn't competing because most of what it did was add multi-user aspects to the project." b2 was a platform aimed at individual bloggers, but everything that Donncha did in b2++ created a better multi-user environment.

And on his blog, Matt wrote a post called "[The Blogging Software Dilemma](#)," in which he proposes forking b2. He wrote:

b2/cafelog is GPL, which means that I could use the existing codebase to create a fork, integrating all the cool stuff that Michel would be working on right now if only he was around. The work would never be lost, as if I fell off the face of the planet a year from now, whatever code I made would be free to the world, and if someone else wanted to pick it up they could. I've decided that this (sic) the course of action I'd like to go in, now all I need is a name. What should it do? Well, it would be nice to have the flexibility of Movable Type, the parsing of Textpattern, the hackability of b2, and the ease of setup of Blogger. Someday, right?

The next day, from Stockport, Mike Little, ([mikelittle](#)), responded:

If you're serious about forking b2 I would be interested in contributing. I'm sure there are one or two others in the community who would be too. Perhaps a post to the b2 forum, suggesting a fork would be a good starting point.

Today, the post that started WordPress gets a lot of traffic when people link to it on the software's anniversary. But for more than a year, that post sat there with just one comment, a marker of the very early days of the project, when for a few months just Matt and Mike, in their homes at opposite sides of the Atlantic, started creating WordPress. It was, at the beginning, just two

developers, working on a small script to make their blogs better. By forking b2 they could continue to use the software, and develop it for their own needs, to scratch their own itches. At that moment, in a small pocket of the internet, the right people connected. They may have been from completely different backgrounds, but a shared love of creating tools, playing with code, and publishing online, brought them together.

On April 1st, 2003, Matt created a [new branch of b2 on SourceForge](#), and, with the name coined by his friend Christine Tremoulet, called it WordPress.

PART 2

10. WordPress 1.2 "Mingus"

Those who switched from Movable Type to WordPress, did so at a time when the software itself was in flux. May 2004 was the month WordPress 1.2 launched -- a release that improved WordPress' flexibility.

Before WordPress 1.2, developers extended WordPress using "hacks," a concept inherited from b2. A "hack" was a set of files bundled with instructions on where to insert the code into the b2 core files. In b2, administration was carried out in a separate PHP file. The b2 user opened a text file, `b2menutop.txt` to add the name of the PHP file that they wanted in the menu. When the code ran, the new menu item would appear after the default menu items. To add a hack to the administration screens, the user needed to place the PHP file into the admin directory and add a reference to it in the text file. If the hack output was supposed to appear on the website, the user needed to edit b2's `index.php` file to put it in the right place. It was a convoluted process that intimidated users uncomfortable with editing code. Also, it meant that when a user updated b2, they had to save

the text file and the index file to ensure that their changes weren't overwritten, and integrate their changes back into the new files.

The [plugin system](#) brought dramatic changes. It uses hooks to enable developers to extend WordPress without having to edit core files. Hooks are triggers placed throughout the codebase that developers use to run their own code and modify the software. There are two types of hooks: filters and actions. Filters were already available in b2 for developers to create hacks, which changed content. For example, by using the `_content` filter, a developer can modify content under conditions they specify. Actions, which were first added to WordPress 1.2, allow developers to run code when events happen. For example, by using the `post_publish` action, a developer can run code whenever a post is published.

The plugin system is an example of Ryan Boren's influence early in the project. As many other developers' involvement trailed off, he dove straight in to development. Mike Little recalls how he hadn't noticed Ryan at first, that he'd been active, but quiet about his contributions. It was the plugin system that [made Mike really take notice](#): "it shone brighter than most of the other things that people were doing. And that's not to say that people weren't doing good stuff, they were, but that was a step change. The hook system was a step change in WordPress development, and it was probably the first step on quite honestly making it the superior product that it is."

The plugin system transformed WordPress for core developers and the wider community. It meant that the core product didn't need to include every developer's pet feature, just the features that made sense for a majority of users. [Ryan says that](#) the plugin system enabled core developers to implement the 80/20 rule: "Is this useful to 80% of our users? If not, try it in a plugin." Unlike hacks, which involved editing core files, plugins could be dropped into a directory in a user's WordPress install. Non-technical users were able to extend their blogs without having to mess around with PHP. The barrier to entry around extending WordPress inched lower.

The first plugin -- which is still bundled with WordPress -- the Hello Dolly plugin, randomly displays a lyric from the Louis Armstrong song *Hello, Dolly* in the top right of the admin dashboard. It was intended as a guide to making plugins. The second plugin was the [blogtimes plugin](#) which generated a bar graph image showing when posts were created over a certain period of time.

Internationalization was another major advancement in WordPress 1.2. From its very beginning, the WordPress community was international in nature. The original developers were from the United States, the United Kingdom, Ireland, and France, and a [forum thread from January 2004 shows how international the growing community was](#). Community members came from Hong Kong, Wales, New Zealand, Japan, and Brazil. With people from all over the world using WordPress, translations soon followed. The [Japanese WordPress site was set up in December 2003](#), only six months after WordPress launched. As WordPress wasn't yet set up for localization at that time, [Otsukare](#), a community member from Japan, [created a multilingual fork of WordPress](#). This was an internationalized version of WordPress that people could use to make their own localizations. It was popular among WordPress users from non-English speaking countries who wanted WordPress in their own language. Its popularity emphasized the necessity of internationalizing WordPress. A lack of proper internationalization tools in WordPress itself could have seen many community members use the fork instead. Maintaining two codebases in this way would have been inefficient and bug-prone.

Ryan used [gettext](#) to internationalize WordPress, which he had experience of from the GNOME project. It involves marking up translatable strings with the `gettext()` function so that a `.pot` file is generated containing all the translation strings. Translators translate the strings and generate `.po` and `.mo` files for localized versions of WordPress.

To internationalize WordPress, Ryan wrapped the translatable strings with the `gettext()` function and put them in a format that provided a full string to the translator which retained context. He went through the code, one line at a time, found everything that could be translated, and marked it up. This meant that when WordPress 1.2 was released, it not only contained the plugin API but was fully internationalized.

On May 19th 2004 -- before WordPress 1.2 was even released with the first official `.pot` file -- Pankaj Narula, ([panjak](#)), [released the first full localization in Hindi using the new gettext method](#). Following the release of WordPress 1.2, there was an explosion of WordPress translations, including [French](#) and [Norwegian](#).

Version 1.2 made WordPress much more accessible and available to a wider group of people. The plugin system turned

WordPress from a straightforward blogging tool into a publishing platform that anyone could extend -- all you needed was a bit of PHP knowledge. And if you couldn't write PHP, you still had access to the ever-widening commons of plugins, as developers created new plugins and distributed their code. Internationalized code mean that it was now possible for people all over the world to have WordPress in their own language, and a community of translators quickly grew around the software. Naoko Takano ([Nao](#)), who was an early member of the Japanese community, [recalls](#) that there were other free software projects that didn't take translations seriously and that WordPress' internationalization efforts encouraged her to join the project.

11. The birth of wp-hackers

WordPress 1.2 development discussions happened in the support forums, the IRC chat rooms, and via private chat and email. But the community was growing. The switchers from Movable Type had increased WordPress' user base, proper internationalization made WordPress more usable worldwide, and the plugin system allowed developers to extend the platform. The growing community needed more diverse communication channels with which to manage development, communicate, and support users.

Many community members were happy with just the forums and IRC, but some developers wanted a mailing list. This was the case particularly for developers with free software project experience. A mailing list is one of the most important communication tools in a free software project and at the time, WordPress had no development mailing lists. It took more than a year after the project's inception for WordPress' developer infrastructure to take shape. The [FAQ on WordPress.org reflects the approach to development at the time](#):

I am a programmer/designer/hacker — can I help with the development of WP?

Sure you can! Post your suggestions and requests for features in the forums. Design or alter some hacks to add functionality to the WP suite of tools. Got some cool ideas on an innovative design? By all means, build it and show it off! If you want to be directly involved in the daily development of WP, the best way is to show your competence by building clean hacks or patches that conform to the developer guidelines. Once you have some code out there, contact Matt and he'll talk to you about getting you involved more directly with development.

That was the process: post on the forums, blog about it, and email the lead developers. There wasn't a clear, single entry point for developers to go. The forum [didn't provide the proper infrastructure that developers were used to on free software projects](#). Ryan Boren had Linux development experience. [He posted to a discussion thread on the forum in 2003](#):

Communities are built around development mailing lists. That's where the bazaar really takes place. A BB [bulletin board] isn't nearly as good for sending and reviewing patches, performing UI reviews, and so forth. The BB is a nice resource that has its purpose, but a mailing list is better suited to development traffic. I would much rather use my favorite email client with its editing, sorting, and filtering capabilities than any web BB. Plus, the mail comes to me, not me to it.

Right now, I send all of my patches directly to Matt. I hope he gets them. If there was a development mailer, I would send the patches there so all interested parties could give them a look and see what people are working on. There was, in fact, a mailing list, but it was a private one with just Matt, Mike, and the other early developers. It wasn't publicly archived and it wasn't possible for anyone to get involved with the discussion. There were no plans to create a public mailing list as, at the time, they preferred development discussion to take place on the WordPress forums.

That discussion thread contains the seed of contention in the WordPress community -- the division between people who didn't write code and people who did. A developer mailing list could segregate the community into different groups -- on the one hand developers, and on the other, everyone else. "It's exactly the mentality that causes most OS projects to become these developer-heavy, 'in-the-know' kind of places that make them unpleasant (and unapproachable) for the average user," [wrote Cena](#).

But with a growing developer base a mailing list was inevitable. Developer mailing lists are one of the primary communication tools for most free software projects. They're a part of the bazaar-style development of a free software project, a place for the hubbub of development discussion. A mailing list allows developers to communicate with one another from their email inbox. Everyone can participate in the conversations that take place. They enable developers to ask questions about development, review patches, and discuss new features. A developer can send a patch to the mailing list and anyone can review it. It also helps to prevent the bottlenecks that can occur when a patch is sent privately to a developer and sits in their inbox waiting for review. Often, advanced users participate -- those who either have questions about the product or who want to answer others' questions. Mailing lists serve a different purpose than an IRC chat room. To participate in a chat room, a person needs to be online at the time a discussion takes place. They can follow up by reading logs (if the chat room is logged), but that's after-the-fact.

The first mailing list in the project, however, wasn't [wp-hackers](#), but [wp-docs](#) which was set up in November 2003 to discuss WordPress' documentation and wiki. It was active for six months before the [hackers mailing list](#) was set up in June 2004. This later moved to [wp-hackers](#). Development discussion shifted from the forums to the mailing list, leaving the forums as a place to provide support.

The [wp-hackers](#) mailing list exploded with activity, busy with heated discussions about issues such as [whether comment links should be nofollow](#) to discourage spammers, [the best way to format the date](#), and [how to start translating WordPress](#). Developers finally had a place to congregate. They embraced the new communication platform -- their new home in the project.

As predicted in the 2003 support forum discussion, the mailing list further cemented the division between those who provided support (whether by answering support tickets or writing documentation) and those who wrote code. Coders are usually more focused on solving code problems than helping with user support. But those providing support sometimes needed the input of those who wrote the code. It frustrated many support forum volunteers that the project's developers weren't usually available to help out.

Still, [wp-hackers](#) became an important place for WordPress development, particularly during the project's early growth. As is the general trend with mailing lists, it became less useful over time, but for the first few years after it launched there was ongoing discussion about WordPress development. Many important features were discussed and debated. It was the place where many of today's developers had their first taste of WordPress.

The WordPress Plugin Repository was the other development resource that appeared around this time. The [WordPress Plugin Repository](#) launched in January 2005. Hosted at [dev.wp-plugins.org](#), and powered by subversion and trac, it's quite different to the user-friendly plugin directory that we're used to today. The plugin repository literally was just a code repository.

In order to make it a little more accessible, a wiki was used to create a very early version of the Plugin Directory.

 Search[Login](#) | [Settings](#) | [Help/Guide](#) | [About Trac](#)[Wiki](#) | [Timeline](#) | [Browse Source](#) | [View Tickets](#) | [Search](#)[Start Page](#) | [Title Index](#) | [Recent Changes](#) | [Page History](#)

Plugin Directory

You can see all plugins hosted here in [the source code browser](#). Those listed below have extra documentation in this wiki.

- [AuthImage](#) - adds a captcha to your comments form
- [Author Highlight](#) - Prints out a class attribute if the name and e-mail address of the comment matches those set
- [AutoTrackbackByCategory](#) - allows admins to specify trackback URI's to categories
- [AutoTranslator](#) - adds a flag banner and language translator links to your WP sidebar
- [BasicBilingual](#) - adds template tags and extra fields in wp-admin for blogs with two languages
- [BlogTimes](#) - generates a timeline of your recent blogging
- [BreadcrumbNavigation](#) - makes it easier to navigate to container categories
- [BunnysTechnoratiTags](#) - add tags painlessly to your posts
- [ComPreVal](#) - Comment Preview and Validation (for 1.5), forces XHTML-compliant comments
- [CommentKiller](#) - deletes all comments in moderation queue
- [CommentPay](#) - asks spammers to pay for their advertising
- [Cricket Moods](#) - a post mood plugin that allows the selection of moods from a list during post authoring.
- [Cross-theme Stylesheets](#) - adds stylesheets to all themes.
- [CSS Compress](#) Automatically removes comments, new lines, tabs, and gzip compresses (GZIP) any CSS file called with "<?php bloginfo('stylesheet_url'); ?>"
- [Dokuwiki Markup](#) - Use the power and simplicity of the dokuwiki markup in your posts and pages
- [DonkieQuote](#) - Displays a random quote from a database on your blog.
- [DownOnMe](#) - Rewrites header tags one level down (h2 -> h3, etc.) in multipost blog pages.
- [FancyToolips](#) - Replace standard browser tooltips with customizable, dynamic ones.
- [favatars](#) - adds favicons next to comments/pingbacks/trackbacks
- [FlickrPost](#) - displays specially tagged Flickr images inside blog posts
- [Gatorize](#) - adds a Newsgator rating to each posting
- [GeoPlugin](#) - adds geographic information (latitude, longitude) to each post

The first version of the WordPress Plugin Directory, in late 2005

While being full of developer tools, it didn't have an easy interface to allow users to find the right plugins. Trac and wikis can often be difficult for non-technical software users to navigate. Users just want to get what they want, easily, and interfaces that look code-heavy can be confusing and off-putting. For developers, however, the repository nurtured the free software community's inclination toward code sharing, and was the first step toward the plugin directory on WordPress.org.

4. Guiding Principles

It's no surprise that WordPress' founding developers had so much in common. b2 attracted Matt and Mike because it was simple, hackable, and usable. WordPress needed to be simple, hackable, and usable too. When they forked b2, Matt and Mike didn't just inherit code, they also inherited b2's ideals -- ideals enriched by fundamental beliefs about software and software development that each founder embraced. These founding philosophies informed much of WordPress' early development, became underpinning philosophies for the project, and still inform decision making about software development and community building today.

Even before WordPress was WordPress, in its nascent state as b2, the focus was on making things as easy as possible for all users, and particularly for those who were new to the platform. From b2, Matt and Mike inherited a focus on usability. On cafelog.com, Michel kept a development log where he recorded his thoughts about developing b2. It was often just a list of things that he'd done that day, but some of the things he mentions provide insight into who he saw as b2's user base. For example, when he [talks about creating a templating system](#), he says that he wants to "make templates customizable by Joe Newbie," in other words, a templating system that any user could customize. Michel believed that a user shouldn't have to be a CSS genius just to customize their site.

One key feature of b2 was the install script, written by a b2 contributor, which made installing the software easy. This was in contrast to the most popular blog platform around at the time -- Movable Type -- which was considered difficult to install. A simple install script meant that people without much technical knowledge could install WordPress. By the time b2 ceased development, it hadn't reached the point of a seamless install, but the code and the intent provided the groundwork for WordPress' "famous 5-minute install."

Allied to this focus on usability, was a commitment to simplicity. Creating simple software means providing users with only exactly what they need to get the job done. Software is a tool and, like any tool, the simpler it is, the easier it is to use. By keeping things simple, a user can figure out the software from the interface itself, with as little external instruction as possible. The way to use software should be self-evident from the interface.

When Michel started b2 development, he made the decision to focus solely on blogging functionality. He didn't want to create a CMS, he wanted to create a blogging platform. This meant that all of his early enhancements were centered on simple tools for getting content on the screen. This focus on blogging ensured that, particularly in the early days of development, the platform remained simple.

Matt's early focus was on web standards. He wanted to ensure his own website's forward compatibility -- that it would work in future browsers and devices. [Jeffrey Zeldman's Forward Compatibility: Designing and Building with Standards](#) was a major influence on Matt. Zeldman's book advocates creating standards-compliant websites that work across browsers and devices. Even prior to forking b2, Matt had converted most of his site to XHTML 1.1.

This meant that many of Matt's first commits to WordPress were focused on HTML semantics and web standards. After setting up the CVS repository (the version control system the project used at that time), and uploading the files, Matt made basic semantic changes to the `index.php` file, fixed whitespace issues, and converted `<div>` tags into heading tags. Using correct tags to generate proper headings reinforces the content's semantic meaning on the page.

In a post just after the launch of WordPress 0.7, Matt outlined his thoughts on the future of WordPress on [WordPress.org](#). He wrote, "one thing that will never change is our commitment to web standards and an unmatched user experience."

Simplicity, usability, and web standards -- these are principles that guide WordPress development to the present day. Over time and as new people join the project, new axioms become a part of the community, augmenting and strengthening these guiding principles. Taken together, these are guiding principles for a user-first focus. Perhaps this came about because Mike and Matt were users of b2 -- they started to blog before they started developing blogging software. They started developing blogging software only because they were interested in making improvements to their own blogs. And as they made improvements they were drawn further in to the development community. But they remained, particularly in the beginning, software users, which meant that while they were developing software they retained an empathy for others. This was true also for other developers who became involved with the project. Developers didn't get involved with the project just because they felt like working on blogging software. They got involved with the project because they used blogging software. They had installed the software for their own blog, found that they wanted to change or improve something, and contributed those changes to WordPress.

What ensures that the user-first approach continues is a decision made prior to even the fork of WordPress, a legacy that the community is either sustained by or stuck with, depending on your perspective. That legacy is the GPL, the software license that Michel distributed b2 with, and it remains WordPress' license to this day.

5. The GPL

The [General Public License](#) (GPL) is the license that WordPress is distributed with. It contains the terms under which the software is distributed. There are few things in the project that have been more divisive. Matt and Mike, the founding developers of WordPress, supported the license. Prior to getting involved with b2, Mike had contributed to free software projects. He'd

submitted patches to the version control system CVS and the DJGPP compiler, and bug reports to database software MySQL. When it came to choosing his blogging software, the license played a big part in his decision. Movable Type, for example, was not an option because it wasn't GPL. As a coder, Mike was accustomed to software-sharing and to working on someone else's code to make it your own. It was while he was working on DJGPP that he first learned about GNU and the ideas behind the GPL. "I learned about Richard Stallman and read his story," [he says](#). "he instilled those four principles that just sort of inspired me."

The principles that inspired Mike have inspired thousands of software developers. They are ideas that resonate with hackers, that speak to freedom and a society based on sharing and collaboration. Communities, like WordPress, have grown up around an ethos which has influenced models of software development all over the world.

The principles are the clauses written in to the General Public License (GPL), the terms under which the software is distributed. The license was written by Richard Stallman for software he released as part of the GNU software project[^fn-1]. Exasperated by proprietary licensing -- which he saw as responsible for the decline of the hacker lab at MIT -- he wanted to distribute software he wrote with a license that protected software users' freedoms. The GPL protects four user freedoms that are at the heart of "free software." "Free" in this context does not apply to price; it refers to freedom, which is the underlying ethos that drives the Free Software Foundation.[^fn-2]

Free software protects four essential freedoms:

- 0 The freedom to run the program, for any purpose.
- 1 The freedom to study how the program works, and change it so it does your computing as you wish.
- 2 The freedom to redistribute copies so you can help your neighbor.
- 3 The freedom to distribute copies of your modified versions to others.

These can be summarized as "users have the freedom to run, copy, distribute, study, change, and improve the software." The freedoms are protected for all users. What this means in practice is that anyone can use a piece of free software -- they can install it in as many places as they want and give it to whoever they wish. They can hack on it, modifying it for their own needs. They can distribute any changes they make. When it comes to a piece of free software, the user has absolute freedom.

It's not enough, however, just to write freedoms into a license. Those user freedoms need to be protected. Otherwise, free software can be absorbed into proprietary software. Developers get the benefits of free software but don't pass on those benefits to others. In order to ensure that these freedoms are protected, the GPL operates using what Stallman calls "copyleft." Copyleft subverts the normal use of copyright laws to protect the terms under which the work can be distributed and redistributed. It's a method of making a work free and requiring that all extended and modified versions of the work are free as well. In this way, the copyright holder can ensure that their work does not end up being part of a proprietary model.

Copyleft works in the following way: 1. The copyright holder asserts that they hold the copyright to the work. 2. To this is added the terms of distribution. This says that anyone can use, modify, and redistribute the work, *provided they pass the same freedom on to everyone else*.

If a programmer wants to use a copyleft work in their own software, then that new work must provide the same freedoms as the original work. Copyright is turned on its head. It is used against itself, or, as [Stallman puts it](#) "we use copyright to guarantee their freedom." A copyleft license doesn't abandon copyright (i.e., by simply putting a work in the public domain), it asserts it and uses it.

The GPL is often described as a viral license. This is because any code integrated with GPL code automatically adopts the license. The GPL spreads. For free software proponents this is important. It means that the body of work that constitutes the commons is self-sustainable and self-perpetuating, thus preserving freedom.

To see copyleft in action, simply open up the license that comes bundled with WordPress to this very day. The head contains the following:

b2 is (c) 2001, 2002 Michel Valdrighi - m@tidakada.com - <http://tidakada.com>

Wherever third party code has been used, credit has been given in the code's comments.

b2 is released under the GPL

and

WordPress - Web publishing software

Copyright 2003-2010 by the contributors

WordPress is released under the GPL

It's the perfect example of how a copyleft license works. Michel asserted his original copyright for b2 and then distributed it under the GPL, which said that anyone was free to distribute and modify it, provided they pass those freedoms on. This meant that when it was originally forked, the developers had no choice but to license WordPress under the GPL. Michel's intention to preserve b2's freedom worked. It also means that anything that includes WordPress source code must also be GPL, so all WordPress users, no matter which form they use WordPress in, have the same freedoms. And when b2 was in danger of becoming vaporware, the license enabled Mike and Matt to fork it and use the code as a base to continue development. The commons, of which the code is a constituent part, continues.

Mike's passion for free software is an important foundation for WordPress' development. b2 was the first free software project that Matt had been involved in. While he later developed a strong belief in the role of free software, it was in b2 and then in the early days of WordPress that Matt first learned about the free software ethos, as a result of Mike's influence. "That's the thing I really learned from Mike," [says Matt in a 2010 interview](#). "b2 was the first open source project I was really involved with. I didn't even really understand what that meant."

The GPL complements a user-first development focus because the license emphasizes the freedoms of users. This is perhaps one of the biggest misunderstandings around the license. When the GPL talks about freedom, it is talking about the freedom of users, not of developers. And often the freedom of users comes at the expense of developers. Developers who want to use GPL code in their own software are restricted to using copyleft licenses for their own products. There are also restrictions upon the code they can integrate with their own GPL code. To use a library in WordPress, for example, that library must be GPL-compatible. This emphasis on freedoms has been a fault-line along which many debates in the project have happened.

The freedom of users is protected even further by the sheer number of contributors to the project. Even if there was a consensus among the project's leaders on changing the license, the freedoms of WordPress users would continue. The WordPress codebase is created by thousands of people all over the world. Each person who writes code for WordPress retains their copyright but agrees to license the code under the GPL. This makes it virtually impossible for the creators of WordPress to ever change the license. To do so, they would need to contact every single person who had ever contributed code to WordPress and ask them to agree to the change. This would include everyone from the most prolific contributors to those who contributed a single patch, from today's lead developers to Matt and Mike, and as far back as Michel. This means that WordPress will always remain free.

The choice that Michael made about using the GPL has been one of the most significant decisions in the project's history. It's meant that distribution terms of the software complements user-first development, ensuring that users are free to do what they want. But what is the cost of user freedom? This is a question that has come up again and again throughout the project's history as different groups have discovered their own rights and freedoms restricted, whether they be designers, developers, or business owners.

[^Fn-1] In his book, *Hackers: Heroes of the Computer Revolution*, Stephen Levy explores the Lab at MIT where Richard Stallman worked, and how the Lab's decline led Stallman to create GNU and write the GPL.

[^Fn-2] This choice of the word "free" in this context has dogged the Free Software Foundation throughout its life. The uninitiated think that "free" refers to cost. The Free Software Foundation often has to qualify "free" with the statement "free as in freedom, not as in beer."

6. WordPress' First Developers

Once the WordPress branch was set up on CVS, Mike and Matt started making changes. They were small and iterative, but they were the first steps that WordPress took away from its predecessor and mark the real beginning of the project. Mike's first commits involved repopulating files that were missing from the branch while Matt added [wpTexturize](#), a tool he created to properly format plain text (quotes to curly quotes, for example, and dashes to en-dashes). Mike's first feature was the excerpt functionality which allows users to display handcrafted post summaries in RSS feeds and in other places.

Over the coming months, Mike and Matt made over 100 commits to the WordPress repository. Notable commits included WordPress' branding, Mike's b2links hack, which remained in WordPress until it was no longer turned on by default in WordPress 3.5 (released in 2012), major changes to the administration panel, and installation process improvements. Creating a simple installation process was something that both the developers felt strongly about. It was important that WordPress have as low a barrier to entry as possible. Anyone should be able to get on the web and publish their content. Matt replaced the [b2install.php](#) file with a new [wp-install.php](#) file. The aim was to keep configuration to a minimum. In the first version, the user had to create a MySQL database, add the configuration details to [b2config.php](#), transfer the files to their server using FTP, and then run the script. The "famous 5-minute install" was refined over time as the developers worked to simplify the process.

While developing WordPress, Mike and Matt were still active on the b2 forums, talking about the new software. But there were a few months after the WordPress project was started during which it was unclear which fork would be b2's official successor. On May 23rd 2003, Michel [announced](#) that once WordPress was launched, it would become the official branch of b2.

On May 27th 2003, [the first version of WordPress, WordPress 0.7, was released](#). Users who switched from b2 to WordPress got some new features, most notably the new, simplified administration panel and the WordPress Links Manager, which allowed users to create a blogroll.

Once WordPress 0.7 was shipped, there was an effort to get other developers involved in the project, starting with Donncha O'Caiomh and Francois Planque, both of whom had created their own forks of b2.

On May 29th, 2003 [Matt emailed Donncha](#) to ask if he would consider merging b2++ with WordPress. Donncha agreed, raising the number of official WordPress developers to three. Matt also contacted Francois Planque to join the project and rewrite his b2evolution improvements for WordPress. Francois considered it but felt that "[it was too much work for too little benefit](#)."

Over the year, Dougal Campbell, ([dougal](#)) and Alex King joined the team. Although not a b2 user, Dougal had investigated using b2 for his blog, and [had blogged about writing his own blogging software](#). Alex had been more actively involved in the b2 community -- while he didn't have a background in PHP, he had learned it through the platform and the community. He particularly recalls Mike Little helping him to refine and improve his code.

It took Dougal and Alex some months to get properly involved. Dougal was busy with work, and Alex's first impressions of WordPress weren't positive. On his blog he wrote about [upgrading from b2 0.6 to either WordPress 0.7 or b2++](#). Installing WordPress didn't offer any significant speed improvements, while b2++ gave him a site that he couldn't log into. In the end, he decided to wait until the next version of WordPress to upgrade. [Matt responded, noting](#) that there would be significant improvements to database speed later. WordPress 0.71's release announced a 300% performance boost: "We're not kidding," [the announcement post reads](#), "this release will perform about three times (or more) faster than previous releases of WordPress and b2." It wasn't, however, [fast enough to convince Alex to upgrade](#). He and Matt kept in touch and in July 2003, Alex announced that he would [help Matt launch a hacks section on WordPress.org](#).

In the early days, WordPress developed organically. A new developer's first change tended to be a small, iterative step, before they moved on to working on their pet feature. Most developers focused on the aspects of web development that they already had an interest or a background in. Matt, for example, focused on semantics and usability.

Mike improved his b2 links plugin. He also introduced [wp-config-sample.php](#). At the time, all b2 and WordPress configuration information was stored in [b2config.php](#). This meant that [upon upgrade a user had to store the file and information safely](#). If they overwrote it, their configuration information would be lost, and they'd end up on the support forums looking for help.

Including `wp-config-sample.php` meant that there was no `wp-config.php` file bundled with WordPress -- the user renamed the file `wp-config.php`, protecting it from being overwritten. This configuration file protection was something Mike had done for previous clients, and while he recalls now that it seems like an obvious thing to do, it solved a problem that users had encountered repeatedly.

Dougal's major focus was the XML-RPC API which at that time supported Blogger's API. XML-RPC is a remote transfer protocol, which allows for remote calls via HTTP. This means that the user can post to their blog or website using a client. The Blogger API didn't cover all of the features that WordPress had. The Movable Type API and MetaWeblog API had additional features that built upon the Blogger API. Dougal added the new features so that the XML-RPC layer covered WordPress' entire feature set. At the time, people would check their RSS feed over and over again and it would regenerate just like a pageview. This could increase the load on the server, slowing the site down. Dougal worked on improving these capabilities, speeding it up by including a cached version.

Alex's first project was checking in a cursor-aware quicktag code. This enabled users to highlight a word in the text editor and use a hotkey to surround the text with HTML tags. In the end, the hacks section on WordPress.org that he had posted about on his blog was never built. Hacks were superseded by the plugin system.

In parallel to WordPress developments, Donncha worked on WPMU (WordPress MU). The original plan was to merge the b2++ codebase with the WordPress codebase but they ended up remaining separate and b2++ became WPMU. WPMU had its own version control system and, eventually, its own [trac instance](#). Donncha recalls that WordPress and WPMU targeted different audiences. "Most people just have one blog," [says Donncha](#), "they don't have half-a-dozen blogs running on one server so multiple sites wouldn't have been a requirement for most people." Over time this situation changed as it became easier and cheaper for people to host their blogs and websites but in 2003 it didn't make sense to have multi-user functionality available to every WordPress user. Instead, Donncha worked on WPMU alongside WordPress, and merged the changes from WordPress into WPMU. When a new version of WordPress was released, Donncha had to merge each file individually into WPMU. He used [Vimdiff](#) to load the two files side by side so could review changes and push them from one file to another. [It wasn't always easy](#). "I had to keep track of the changes that were made in case they broke anything. So at the back of my mind I'd be thinking 'did that change I made five files back, will that affect this change?'" As WordPress got bigger and bigger the merges became more difficult to manage.

In late 2004, a now long-standing WordPress developer took his first steps in the community. Ryan Boren, ([ryan](#)), was a developer at Cisco Systems. Like Mike, Ryan had contributed to free software projects before, particularly Gnome and Linux and is a big advocate of free software. When he found WordPress he'd been looking for a free and open source blogging platform. Ryan had been blogging for a number of years, and wrote [his first blog post in September 2000](#). His earliest blog posts were on Blogger then Greymatter until he decided that he "wanted something new and a little nicer." He liked WordPress' markup and CSS so he made the switch, scratching his own itch by writing a Greymatter importer to move his content to WordPress.

Almost straight away, Ryan had influence in the community, writing huge amounts of code and also providing advice on how WordPress development should be carried out. He brought more free software project experience than anyone else had had so far he was frequently found on forum threads sharing his thoughts on how things should be run. While he had little experience with PHP at that time, his coding background allowed him to quickly learn and it wasn't long before he was given commit access to the WordPress repository.

7. Inside the Bazaar

In these first days of WordPress, new developers were often given direct commit access to the code repository. This meant a developer could work on code and add it directly to the core software. There was no requirement for code review (though sometimes code would be sent around among developers before being committed). WordPress was still a small blogging script

used mainly by the people who'd written the software. Many of the early commits are from names absent from the commit logs today: `mikelittle`, `alex_t_king`, `emc3` (dougal), and even `michelvaldrighi`, who came back and contributed to WordPress.

There was no real process in those early days. It was the extreme of what Eric Raymond talks about in his work *The Cathedral and the Bazaar*. Raymond contrasts the open source bazaar-style development model with the "cathedral building" of traditional software development. The metaphor of the bazaar is apt, with its noise and bustle; people talking on top of each other, each with their own aim and agenda. A free software project operates in this jumble. Common sense tells you it's all wrong but, just like the bazaar, this throng somehow works.

The first two years of the WordPress project were marked by this laissez-faire approach to development. A developer saw a problem and fixed it. If a developer was interested in a feature, they built it. They used their own experience as bloggers, as well as their b2 forum activity, to guide them. "As bloggers, we had similar desires to those other people had," says Mike. "I seem to remember still sticking around the b2 forums and looking at what people were asking about and what people wanted while it was still in b2 and getting inspiration and ideas from that."

It wasn't until later that more hierarchy was introduced into the project, but in those first few years it hardly mattered. So few people were actually using the software. Many of the developers felt that they were working on blogging software for themselves and if other people wanted to use it, great. The legacy of that time, though, is that today's developers must maintain and work with all of that code.

A frequently asked question, particularly around the time of the fork, was whether the new WordPress developers planned to rewrite the code base. b2's emphasis on getting easy-to-use features on the screen as quickly as possible had often come at the expense of good coding practices. Michel [was learning about PHP](#) when he wrote b2; he tried to get new features on the screen as soon as he imagined them.

This simplistic, often chaotic, code base, put some developers off. Prior to joining the project, Dougal posted to the support forum asking [how far the developers intended to go with the rewrite](#): were they planning to rewrite the whole code base from scratch, or would something recognizably b2 still remain? [The response was that](#) they planned to structure the code more logically as they went along, with object oriented code as a long-term goal.

There wouldn't be a total rewrite of WordPress. The WordPress project was launched in the wake of the launch of Mozilla's browser, which was the result of a three-and-a-half year rewrite of Netscape. Using Mozilla as a negative example, many developers in the free software community feel that [rewriting software is a big mistake](#). While rewriting might produce an all-new code base, it lacks the years of testing and bug fixes that come from using a mature code base. It also leaves space for competitors to emerge while developers are focused internally on rewriting.

Instead of a wholesale rewrite, WordPress' developers worked to iteratively improve and refactor code. For example, in late 2003, major changes to the file structure of WordPress involved replacing "b2" files with "wp-", dubbed [The Great Renaming](#). Tidying up b2's files had been on Michel's agenda in 2001 and he had made some improvements already, but they lacked consistency. Now the WordPress project had decided to tackle the problem. When the files were renamed with the new `wp-` prefix, instead of the old b2 one, hack writers found that their hacks no longer worked, but it was believed that the upheaval was necessary to organize the file structure for long-term stability. WordPress' file structure morphed from b2 to the familiar WordPress file structure used today, with many files consolidated into the `wp-includes` and `wp-admin` folders.

These sort of iterative steps have been more of a feature of WordPress' development than any major restructuring or rearchitecting. And, over time, such changes have become harder to do as the number of people using the software means that many more sites are likely to break.

To facilitate development, the developers needed a way to communicate. The hackers mailing list wasn't set up for 17 months after project launch and the project's main developers communicated on a private mailing list. IRC was one of the first tools the community used for communication. b2/Cafelog had a channel hosted on the freenode network so it made sense for WordPress to have one too. The first of these was `#wordpress`.

An IRC channel provides a virtual meeting space where people can instantaneously communicate with one another. It's also

possible to log an IRC channel so that a record of what happened can be posted online. In the early days of WordPress, many community members spent all day hanging out in the IRC chat room. For some, it was the tone of the IRC chat room that got them more actively involved in the WordPress community. Owen Winkler ([ringmaster](#)) recalls:

I had stumbled on IRC channels for other programs before, and you know, you ask a question and no one answers or they make fun of you. WordPress was never like that. When I started out, if you came to the channel and you asked a question, people answered the question. After you learned a bit, if you stuck around, you would answer the questions too.

It was this camaraderie that caused people to stick around. Many of them were learning to write code, making software for the first time, and they were doing it together. Over time, WordPress spawned more IRC chat rooms. The `#wordpress` IRC chat room morphed into a place for user support, with a small community of regulars that frequent it. The `#wordpress-dev` channel became the place where WordPress development took place, including weekly meetings and development chats. There were also individual chat rooms for the teams that work on different areas of the project. [fn^1]

WordPress.org forums were the other communication tool that the project had right from the beginning. WordPress.org launched in April 2003; it was first of all home to the development blog, some schematic documentation, and support forums. The [original WordPress homepage](#) told the world that "WordPress is a semantic personal publishing platform with a focus on aesthetics, web standards, and usability." The site gave the WordPress community a presence and the forums provided a home.

Originally the forums ran on miniBB but as the number of people on the support forums grew, the software was unable to handle the increased load. In 2004, while stuck in San Francisco over Christmas, Matt took what he'd learned from WordPress and applied it to forum software, writing bbPress. Now, bbPress is a plugin, but when it was originally coded it was a standalone piece of software with its own templating system. [Matt wrote in the launch post](#) that he wanted to "bring some weblog and WordPress sensibilities to forum software."

Today, the WordPress.org forums are mostly used for providing support to users and developers, but back when they were first set up, they were the community's primary method of communication. The [first post on the forums](#) appeared before WordPress was even released, with a request to beta test the software. Word had gotten out that there was a fork of b2 and people were eager to use it.

The support forums were used to talk about everything related to WordPress: [the WordPress.org website](#), [bug reports](#), [troubleshooting](#), and to [request design feedback](#). It was also a place for people to post their [hacks](#) and later their [plugins](#).

Developers communicated on these open channels but anyone was able to join in. It didn't take long for people who weren't developers to gravitate to the project. These were sometimes people who were using WordPress but who didn't have the technical skills or confidence to actively contribute to the code. Others were more interested in the supporting roles that are essential to a successful project -- writing documentation, for example, or providing support. Some of these people would go on to have just as big an influence on the project as any of the developers.

[fn^1] In late 2014, the WordPress project moved its communication from IRC to Slack.

8. Support and Documentation

Many early WordPress developers got their start in the project answering questions on the support forums, and there was a similar entry point for people who worked on other aspects of the project: a user installs the platform, encounters a problem, and visits the support forum to ask a question. They stick around, surf the forums or hang out in the chat rooms. Then someone asks a question that they know the answer to and that's it, they're hooked. Mark Riley ([Podz](#)) was one of the first WordPress.org forum moderators. "You're a hero to somebody every day, aren't you?" [he says now](#). "There's nothing like somebody saying thank you. Yay, you fixed it! It works! You're thinking cool. I'll remember that. That was cool. I like doing this."

A contributor community grew in parallel to the development community. Developers wrote code, other contributors helped with support and documentation. People tried WordPress, liked it, and wanted to help out. Craig Hartel, ([nuclearmoose](#)), was an early contributor. He signed up at [WordPress.org](#) in November 2003. Like many contributors, he was interested in blogging and had programming experience. "I didn't have any specific skills," [he says](#) "but there was no better way than jumping right in. I decided I was going to find some way to get involved." He asked questions, dropped hints that he wanted to help, and after hanging out on the IRC channel, "[realized](#) that getting involved was a matter of just doing something."

The project's user-centric focus meant there were ways for people from a variety of backgrounds to help. Anyone could answer support forum questions, write documentation, or get involved with IRC discussions. Some contributors found that while developers aren't always good at explaining things to non-technical users, they could translate "developer speak" into "user speak." These community members acted as advocates for users; Podz, for example, became a go-between for the support forums and the wp-hackers mailing lists. As the developers of the software become more absorbed in developing the software and started using it less, this sort of user advocacy became increasingly important.

Not long after WordPress launched, blogs starting cropping up dedicated to the platform. The first, [Weblog Tools Collection](#) (WLTC), was a blog that Mark Ghosh ([laughinglizard](#)) initially launched to cover every type of weblog tool. It was the first of a long string of WordPress community blogs, followed closely by Carthik Sharma's ([Carthik](#)) [Wordlog](#) and Lorelle van Fossen's ([lorelle](#)) [Lorelle on WordPress](#).

These were places outside official channels where people congregated -- where enthusiasts could write about the growing platform, providing information, tutorials, and commentary. Some focused on tutorials, sharing guides on how to do things with WordPress, others created lists of plugins and themes that drew large amounts of traffic. Mark Riley's tamba2 blog, for example, was home to a number of popular tutorials, and Lorelle van Fossen, who wrote many popular tutorials, ported her writing over to WordPress' official documentation. The authors soon discovered that people were interested in what they had to say. "I suddenly got all of this attention for not knowing a lot and not really doing a lot," [says Mark Ghosh](#), "and that really pleased me." The community respect he got for running WLTC spurred him to help out more with the forums, write his own plugins, and get more involved. Posts on WLTC about platforms like Movable Type quickly tailed off and almost all of the posts are on WordPress, or on migrating from other platforms to WordPress.

At its peak, WLTC received 12,000 to 15,000 unique hits per day but Mark was never fully able to take advantage of the traffic. Running a niche community blog takes a lot of work and [doesn't result in a huge monetary payoff](#). "Most of the people who came to WLTC wanted news about plugins, or they wanted to know how to do X, Y, or Z. They were trying to find this information and the quality of audience was kind of low." People who visited WLTC were looking for ways to fix their blog or website, learn tricks, or find out about the latest free themes and plugins. They weren't necessarily valuable to advertisers. However, WLTC played a major role in the WordPress community's development, providing a home for discussion and debate away from WordPress.org.

With the project attracting so many writers and bloggers it's no surprise that six months after the project launch there were calls for documentation. Users needed it and there were people willing to write it. The people answering questions on the forums saw this need -- users asked the same questions over and over again. With good documentation they could help themselves. In November 2003, WordPress' first mailing list was set up -- to discuss WordPress' documentation.

The first schematic documentation was on the WordPress.org website, but this was merely an outline that lacked content. By and large they were holding pages for content that was promised in the future:

WordPress

HOME | ABOUT | Docs | DEV BLOG | HACKS | SUPPORT | DOWNLOAD

- [Introduction](#)
- [Tutorials](#)
- [Template](#)
- [Reference](#)
- [Developer](#)

Tips and Tutorials

This will be an index of locally hosted WordPress tutorials, which may range from a guide to tweaking the WordPress CSS to detailed instructions for setting up a blog in a specific way. Very open-ended.

WordPress Announcements:

[join](#)

The WordPress.org docs page in late 2003

In December 2003 the WordPress wiki launched. Now any contributor could help with the documentation. Free software projects often use wikis for their docs. The advantage is that anyone can easily edit the content. Some wikis require a login, while others can be edited by anyone. The downside to using a wiki is that contributors have to learn a new tool with a new syntax for creating content. This is particularly onerous when the free software project is a CMS. A further problem is that, without careful curation, it can become messy, out-of-date, and difficult to navigate.

Originally, the wiki was designed to complement the [official documentation](#). The [landing page informed visitors](#) that it was "designed for us to be able to work together on projects." While developers worked towards shipping WordPress 1.0 in January 2004, other community members worked furiously on the wiki.

This was in contrast to the aborted work on the official documentation. An [FAQ](#) and [template documentation](#) were created. But the [majority of documentation was written on the wiki](#).

While the official docs felt formal and rigid -- a place for only official documentation writers -- the wiki was informal and free-form, an experimental place where anyone could help out. By July 2004, the wiki was the main documentation for WordPress. It needed a name. In WordPress' IRC chat room, ([monkinetic](#)) suggested "Codex." The community loved the suggestion. Matt said it was "short, sweet, and we can totally own that word on Google."

Writing documentation for WordPress wasn't always easy, particularly in those first few years. In a post on WordPress.org, Cena Mayo, ([cena](#)), who had taken on the [role of reporting on the WordPress.org blog](#), outlined [some of the issues](#):

Part of the problem is the rapidly changing face of WordPress itself. The CVS is currently at version 1.2-alpha, with almost daily updates. 1.2, which will be the next official release, is much different from the widely used 1.0.1/1.02 series, and even more different from the still-used .72.

With changing file structures, new features appearing, new template tags, and new database tables, writing formal documentation must have felt, with this rate of change, like a pointless task. By April 2004 the software changed so fast that much of the documentation on hacks (the standard way of extending WordPress) was out of date. With WordPress 1.2 shipping in May there was a huge amount of documentation that needed to be written. But just before release, something happened that distracted contributors from writing documentation and writing code, and that brought together everyone in the community.

9. Freedom Zero

In early 2004, Movable Type was the most popular self-hosted blogging platform. The blogcensus.net service pegged Movable Type at [70% of the market share for self-hosted blog platforms](#) in February 2004. It was used all over the world, by everyone from individual bloggers to big media outlets.

On May 13th 2004, Six Apart, the company behind Movable Type, [announced changes to Movable Type's license](#). Movable Type 3.0, the newest version, came with licensing restrictions which meant that users not only had to pay for software that was previously free, but pay for each additional software installation. People who had been using the platform for free now had to pay. Movable Type users were upset and angry about the changes to the license, and they took to their blogs to tell the world. Anil Dash ([anildash](#)), who was Vice President and Chief Evangelist at Six Apart, [says now](#):

Nobody had ever had an audience where by definition every single one of your customers had a blog before. And so nobody had ever had a social media shit storm before. And now you can see a fast food company makes a stupid tweet, and they have like a checklist. They're like, oh, okay, we fired the intern, we're sorry, it won't happen again, here's the hashtag for how we're going to apologize, we made a donation... like you just run through the list. It doesn't even get attention unless it's something really egregious. But it hadn't happened before. And mostly because nobody else had a lot of customers that were bloggers before. So you might have one. But every single person we'd ever had as a customer was a blogger.

Respected programmer and writer Mark Pilgrim wrote one of the most influential posts on the license changes. In his post, [Freedom Zero](#), Pilgrim reminds his readers that while Movable Type had been "free" (as in free from cost), it wasn't free according to the [definition of the Free Software Foundation](#) (free as in freedom, not as in beer), and while the source code might be available, it wasn't open source [as defined by the Open Source Initiative](#). He described Movable Type as having been "free enough"; developers could hack on the code and add features, and while they couldn't redistribute their modifications, they could share patches, so everyone had been happy enough.

Pilgrim, like Movable Type's other customers, had watched as Movable Type 2.6 fell behind while Six Apart focused on their growing hosted platform -- Typepad. He, and others, waited for Movable Type 3.0 to appear, only to discover that the new features were lacking and, worse, there was a new licensing plan so that "free enough" no longer meant free in any sense.

To continue to run his sites, Pilgrim would have to pay \$535. Instead of paying that money to Six Apart, he donated it to WordPress. [He wrote](#):

Freedom 0 is the freedom to run the program, for any purpose. WordPress gives me that freedom; Movable Type does not. It never really did, but it was free enough so we all looked the other way, myself included. But Movable Type 3.0 changes the rules, and prices me right out of the market. I do not have the freedom to run the program for any purpose; I only have the limited set of freedoms that Six Apart chooses to bestow upon me, and every new version seems to bestow fewer and fewer freedoms. With Movable Type 2.6, I was allowed to run 11 sites. In 3.0, that right will cost me \$535.

WordPress is free software. Its rules will never change. In the event that the WordPress community disbands and development stops, a new community can form around the orphaned code. It's happened once already. In the extremely unlikely event that every single contributor (including every contributor to the original b2) agrees to relicense the code under a more restrictive license, I can still fork the current GPL-licensed code and start a new community around it. There is always a path forward. There are no dead ends.

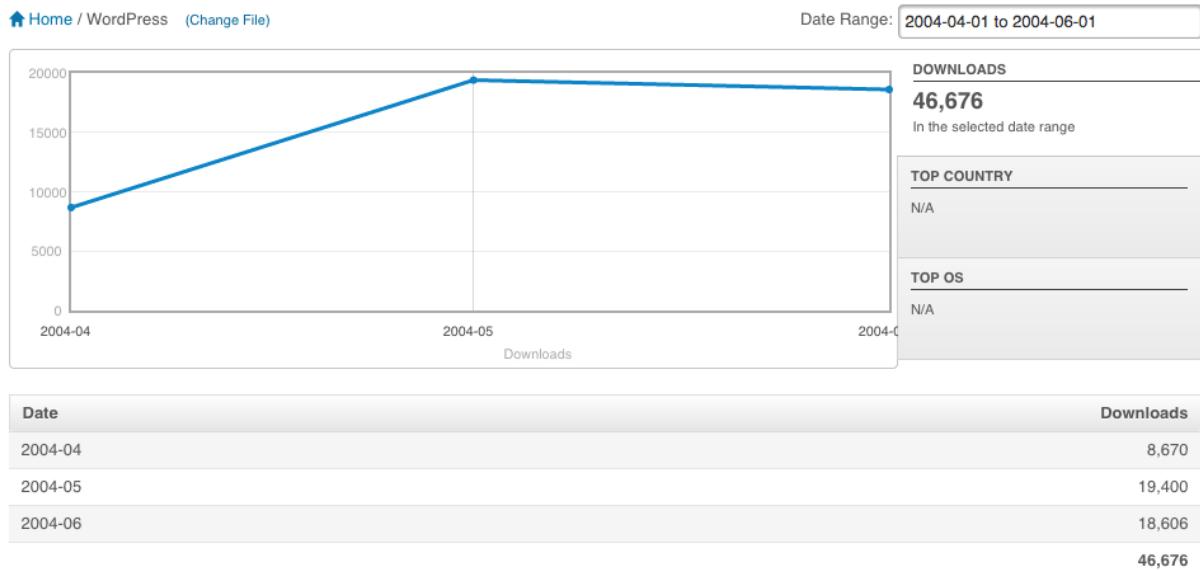
Movable Type is a dead end. In the long run, the utility of all non-Free software approaches zero. All non-Free software is a dead end.

This site now runs WordPress.

Pilgrim's post was one factor that instigated an exodus from Movable Type to WordPress. Even if users were willing to pay for their websites, what if Six Apart changed their licensing terms again? How much would Movable 4.0 cost? How many sites would users be able to run? It was too easy for Six Apart to change the rules of the game, but WordPress's rules could never change under its GPL license.

Six Apart's move galvanized the WordPress community. It helped grow the WordPress platform. All of those dissatisfied Movable Type users were in need of a blogging platform that was flexible and without restrictions. Mark Pilgrim pointed them to WordPress, and the community was only too happy to help people migrate. Andrea Rennick ([andrea_r](#)), who was a Movable Type user at that time, [recalls](#):

That's when I first heard people starting to say, 'Hey, there's this alternative' and then the buzz went around. There's an alternative. It's easier to use. You can set up multiple installs. You can't set up multiple blogs inside the platform but it was super easy to set up a whole new one on your hosting site. It was a lot easier to install on a shared host too.



The sourceforge graph showing the increase in the number of WordPress downloads between April and May 2004

WordPress downloads on SourceForge more than doubled, increasing from 8,670 in April 2004 to 19,400 in May. The IRC chat rooms were buzzing. Craig Hartel [recalls now](#) that "we saw an opportunity to bring people who were passionate about something to our passion for WordPress. Maybe they would find WordPress to be as good, if not better than Movable Type. We spent a lot of time on the forums and directly with people to show that we were a stronger community, that we weren't the kind of community that was just going to slap some information out there and then you were on your own."

The decision not to rewrite the platform was prescient: if the community had buried itself in a rewrite it wouldn't have been ready to welcome and support all of the new WordPress users. Instead, they were ready. For weeks, everyone was focused on helping "switchers." Developers wrote scripts to help people easily migrate from Movable Type to WordPress. Writers wrote [guides to migrating from Movable Type to WordPress](#). On the WordPress.org blog, a [post about Movable Type's licensing scheme](#) reminds users that the GPL "ensures that the full source is available free of charge, legally." And [WordPress had its first mention on Slashdot](#), as a free and open source alternative to Movable Type.

The WordPress community actively sought out people who were dissatisfied with Movable Type and suggested that they move to WordPress. Anil [remembers that](#) "I was responding in the comments on every single blog post that had complained and saw on every single blog post Matt going in and saying you should try WordPress. And I was livid."

Pilgrim was one of many prominent people who moved from Movable Type to WordPress. Another significant adopter was [Molly E. Holzschlag](#). Along with [Porter Glendinning](#), Molly had written a book about Movable Type, *Teach Yourself Movable Type in 24 Hours*, which was released the same week that Movable Type launched. There were people who switched for reasons other than licensing. They were frustrated by Movable Type's functionality. For Om Malik, [Movable Type 3.0 had simply too many bugs](#).

When Six Apart changed Movable Type's license it threw into relief the power relationship between developer and user. It became obvious that Six Apart held all of the power. At any time they could increase prices, change the license, and change the rules. The license protected the developers. WordPress, on the other hand, had a license that protected its users, and it was to this user-focused, user-driven community, that Movable Type users flocked. Many of those who moved to WordPress would go on to become long-standing community members. It was the first of many times that WordPress' license, the GPL, would ignite the community, and its positive effects saw WordPress go from a small fork of b2 to a competitor as a standalone blogging platform.

Endmatter

January 2004 - February 2005

WordPress 0.7, 0.71, 0.72, 1.0, 1.2, 1.5

Jazzers Miles Davis Charles Mingus Billy Strayhorn

Features WordPress 0.7 Texturize Link Manager XHTML 1.1 New Admin Interface Manual Excerpts

WordPress 0.71 Post status Comment status

WordPress 0.72 Password protected posts Cursor-aware quicktags RSS feeds for comments

WordPress 1.0 Pretty permalinks Multiple categories per post In-browser wp-config.php setup Comment moderation

WordPress 1.2 Plugin architecture Sub-categories Automatic thumbnails Encrypted passwords and cookies Gettext localization
Post preview Advanced comment moderation

WordPress 1.5 Theme system Pages Kubrick default theme Comment whitelisting Register for comments

Contributors

To be added

Community Activity WordPress Codex launched Wp-hackers launched Wp-docs launched CSS Style competition

PART 3

12. Themes

Templating was the next area ripe for improvement -- users could extend their blog, and have it in their own language, but there wasn't yet an easy way to change their site's design. Even in the b2 forums, many support questions were about changing a website's design. A blog is a visual representation of the author, their tastes, and their interests -- a blog is a home on the internet, and, just with any home, owners decorate it according to their own tastes. To change the look of a b2 or early WordPress blog, the user had to create a CSS stylesheet. This changed the colors and layout on the front end. However, it didn't offer any real flexibility in site design; a robust templating system was needed. Michel had looked into creating a templating system for b2 but it was in the transition between WordPress 1.2 and 1.5 that WordPress got its theme system.

A lot of research went into finding the best approach to templating. Smarty templates came up again and again. Smarty is a PHP templating system that allows the user to change the front end independently of the back end. The user can change their site's design without having to worry about the rest of the application. There [are a number of posts on the WordPress.org development blog discussing the merits of Smarty](#). [Donncha even imported it to the repository](#) (his first commit to the project). But, despite sharing the PHP language with WordPress, Smarty had a difficult syntax to learn. In the end [it was rejected for being too complicated](#). What WordPress needed was a system as easy to use as the software itself.

While templating system discussions continued, WordPress users got creative with CSS stylesheets. To make switching designs easy, Alex King wrote a [CSS Style Switcher](#) hack, which came with three CSS stylesheets. Not everyone who had a WordPress blog wanted to create their own stylesheet, and many didn't know how. Users needed a pool of stylesheets to choose from. To grow the number of stylesheets available, Alex ran a WordPress [CSS Style competition](#). Prizes, donated by members of the community, were offered for the top three stylesheets; \$70, \$35, and \$10 respectively.

Creating a resource similar to the popular [CSS Zen Garden](#) was the secondary aim of the stylesheet competition. Just as the CSS Zen Garden showed off CSS' flexibility, a CSS stylesheet repository would show off WordPress' flexibility.

The competition created buzz in the community. In total, there were 38 submissions. Naoko Takano won the first competition with her entry [Pink Lillies](#):

WP CSS Styles

1/20/2004
WordPress 1.0.1 Released
Filed under: [WordPress](#) — Alex King @ 9:43 pm

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Vestibulum id nulla. Aenean commodo, mauris a lobortis facilisis, magna tellus suscipit purus, in gravida justo turpis vitae risus. Nam porttitor, tellus viverra ornare ultricies, tortor sem ultricies mauris, quis adipiscing justo elit vel ipsum. Praesent venenatis diam vitae purus.

Pellentesque id magna nonummy nisl ornare varius. Nulla facilisi. Aliquam eu quam eget est tempor feugiat. Quisque laoreet, mi et pellentesque vehicula, nunc mi luctus pede, non tincidunt arcu orci eu eros. Fusce lectus lectus, dignissim ac, dapibus eget, adipiscing et, erat. Pellentesque id magna nonummy nisl ornare varius. Nulla facilisi. Aliquam eu quam eget est tempor feugiat. Phasellus vel metus et mi vehicula ornare. Nulla ultrices lorem vel felis. Maecenas vitae enim vitae mauris auctor laculis. Nullam aliquam, diam ac vulputate tincidunt, leo pede ultricies risus, quis egestas ante dui dul id nisl.

[Comments \(2\)](#)

1/20/2004
Woman Accidentally Deletes Internet
Filed under: [Technology](#) — Alex King @ 9:43 pm

Lorem ipsum dolor sit amet, [Woman Accidentally Deletes Internet](#) consectetur adipiscing elit. Vestibulum id nulla. Aenean commodo, mauris a lobortis facilisis, magna tellus suscipit purus, in gravida justo turpis vitae risus. Nam porttitor, tellus viverra ornare ultricies, tortor sem ultricies mauris, quis adipiscing justo vel ipsum.

▷ Praesent venenatis diam vitae purus.
▷ Quisque laoreet, mi et pellentesque vehicula, nunc mi luctus pede, non tincidunt arcu orci eu eros.

WordPress
Matt
Mike
Alex
Dougal

Categories:
[WordPress](#)
[Development](#)
[Web Standards](#)
[Technology](#)
[General](#)

Search:

Archives:
January 2004
December 2003
November 2003
October 2003
September 2003
August 2003
July 2003
June 2003

January 2004

S	M	T	W	T	F	S
					1	2
					3	
			4	5	6	7
			8	9	10	
			11	12	13	14
			15	16	17	
			18	19	20	21
			22	23	24	
			25	26	27	28
			29	30	31	

[« Dec](#)

Others:

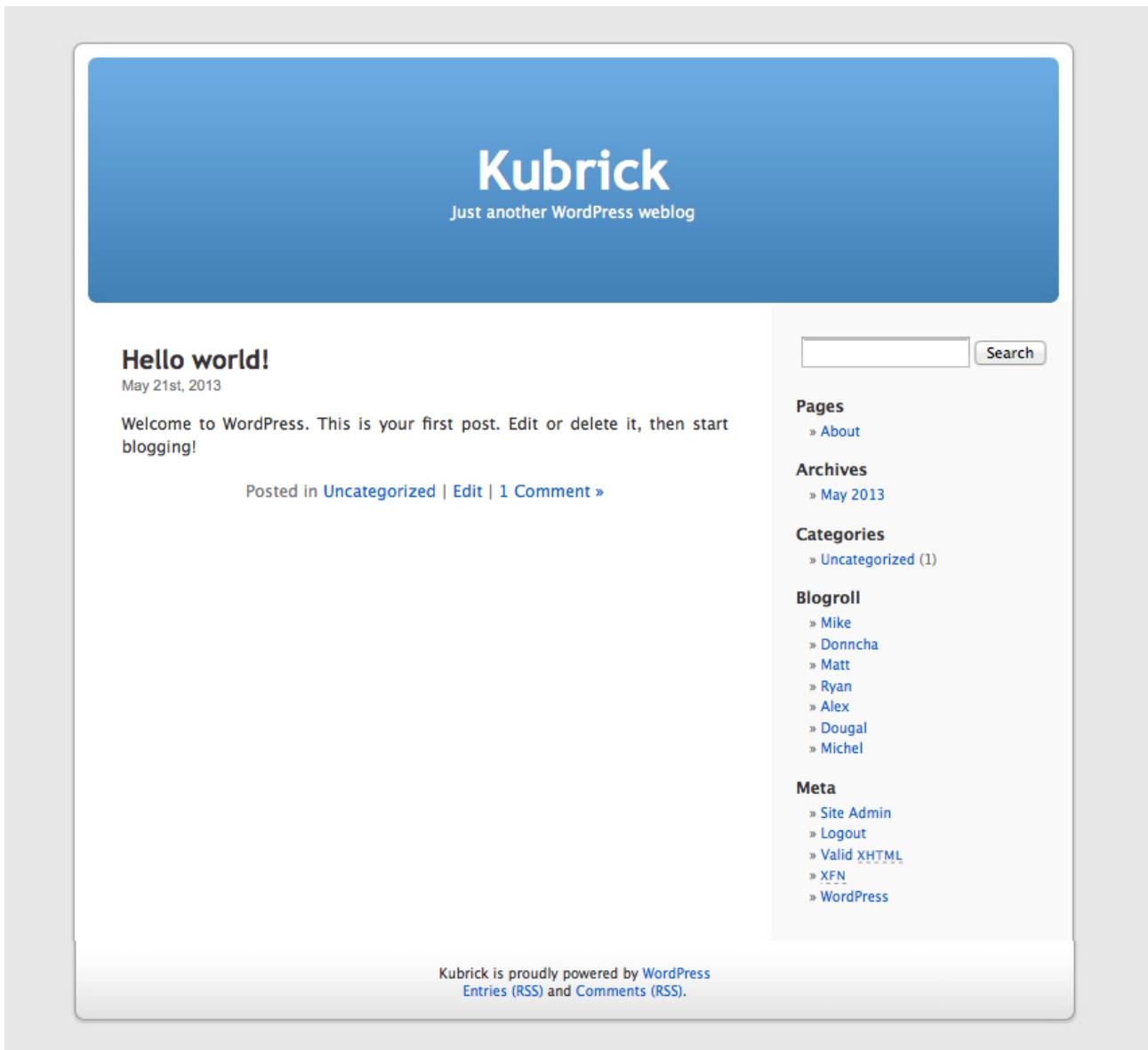
The Pink Lillies design, by Naoko Takano

The competition successfully widened the pool of available stylesheets, increasing the number from seven to 45. On his website, Alex launched a [style browser](#) to allow visitors to view the different stylesheets. The competition ran again in 2005, this time receiving more than a hundred submissions.

By the second competition, the theme system was in place and designers had more tools to design and build their theme. Alex's experience in the second competition, however, foreshadowed problems that would dog the community in later years. In hosting 138 themes on his site, [Alex had to review all of the code](#) to make sure there was nothing malicious, and to ensure that each theme used coding best practices. He decided not to host the competition again in 2006 due to the sheer amount of work it required. WordPress' growth meant there would be even more submissions, far too many for one person to review. This problem persisted as the project grew: how does one balance a low barrier to entry with achieving good code quality, particularly in third party plugins and themes?

Still, the 2005 competition showed off the flexibility of the brand new theme system, which appeared in February 2005, in WordPress 1.5 [fn^1]. In the end, the theme system was built using PHP which is, after all, a templating language itself. Using a PHP template tag system was fast and easy, particularly for WordPress developers and designers who were learning PHP. It was "cheap and easy, and well-known and portable," [says Ryan](#). The theme system breaks a theme down into its component parts -- header, footer, sidebar, for example. Each part is an individual file which a designer can customize. Designers use template tags to call different elements to display on the front end. This bundle of files is a WordPress theme. A native WordPress theme system, as opposed to a templating system such as Smarty, meant that designers could design and build themes without learning an entirely new syntax.

Bundled with WordPress 1.5 was a new default theme -- an adapted version of Michael Heilemann's Kubrick. While some welcomed the new theme, [others were unhappy that it was chosen](#):



The Kubrick Theme, which was WordPress' default theme until 2010

One of the problems with Kubrick was that it contained images; if a user wanted to change their theme they had to use an external image editor. Some felt that users should not be expected to download additional software just to change their site's design. Others thought Kubrick too complex; it had .htaccess issues, and that other (better) default theme choices existed. The [thread on the forums](#) reaches five pages with substantial flaming.

These types of fires ignite quickly in bazaar style development. They can get out of hand as people take their opinion to their blogs and social media. As with many debates in free software communities, the Kubrick debate burned ferociously for a short time before fizzling out. But the brief outbreak portended later debates about WordPress themes. There's something about themes that ignites passions more than other aspects of WordPress.

Still, by the release of WordPress 1.5, the software had two elements that define the project and the community: themes and plugins. These two improvements transformed WordPress from standalone software into an extensible platform. Extensibility creates the right conditions for an ecosystem to flourish. If a product is solid and attracts users then developers will follow, extending the software and building tools for it. The theme and plugin systems made this possible, in both features and design. It also meant that some developers could focus on improving the core product while others could work on their own pet project. core developers could focus on refining the software into something that millions of people wanted to use.

[fn^1] WordPress versions were skipped between 1.2 and 1.5. This was due to the length of time between the two releases.

13. Development in a Funnel

Following WordPress 1.5, the software development process changed. This was precipitated by a shift in the version control system from CVS to a more modern version control system, Subversion (SVN)^[^fn-1]. The most active contributors at that time were Ryan and Matt, and when the move was made none of the other original developers had their commit access renewed.

A committer is someone who can use the commit command to make changes to the group's central repository. The number of people who can do this on a project varies, with projects deciding on the number of committers depending on their own structure and philosophies. The question of who should have commit access to the WordPress repository comes up again and again. There are regular threads requesting that more people be given commit.

The move from CVS to Subversion marked a long period in which WordPress development operated through a funnel. Contributors created a patch, uploaded it to Mosquito (and later Trac) ^[^fn-2], and it was reviewed by one of the committers who committed the code to the main repository. Over time, the funnel narrowed as Matt's focus went elsewhere and Ryan drove development. This "funnel" development style has advantages and disadvantages, and it was frequently the subject of discussion among the community, particularly on wp-hackers. The advantage of a funnelling process is that disagreements about code happen on the mailing list, before a change lands in the repository. The disadvantage is that one person has to review every patch, which frustrates developers waiting for their patches to be committed.

This approach to development was heavily influenced by the Linux project, in which Ryan had participated. As WordPress began to take shape as a recognized free software project, with a defined development process and proper developer infrastructure, Ryan's previous experience guided many of these changes. WordPress did differ in the fact that [Linux has only one committer](#) -- project founder Linus Torvalds -- whereas WordPress had two in Matt and Ryan. Over the years this opened up to more and more committers. The other difference is that Linux has "maintainers" who maintain different subsystems before pushing patches upstream. This means that Torvalds doesn't review the thousands of patches that go into the kernel. He reviews a fraction of them and delegates review to trusted subsystem maintainers. Although WordPress in the future would move toward component maintainers, it never followed Linux in this manner.

There are a number of reasons that people request commit access: to speed up development, to prevent situations in which patches wait months for a review, to acknowledge the contributions of project regulars, to create a more egalitarian project structure. Adding a new committer to a project became a major decision and the project's approach changed dramatically from the wild west days of coding in WordPress 0.7. When the project started, anyone who demonstrated some technical expertise and a small amount of dedication was given commit access. All of those developers could commit code to the repository whenever they wanted. But as the project and the number of users grew, it became more important to have some sort of filtering mechanism. A committer is the filter through which the code is passed. When someone is given commit access to the repository it demonstrates a level of trust. It signifies that a contributor is trusted to know which code should make it in to WordPress.

[Karl Fogel describes committers](#) as "the only formally distinct class of people found in all open source projects." "Committees are an unavoidable concession to discrimination in a system which is otherwise as non-discriminatory as possible," he writes. As much as one tries to maintain that commit is a functional role, commit access is a symbol of trust, both in terms of coding skills and in a person's adherence to the project's core beliefs and ethos. A power dynamic exists between those who have commit access and those who don't.

Committees provide quality control. To decide who to give commit access to, a project maintainer has to first find people who not only write good code but who are good at reviewing other people's code. This means being good at recognizing places in which code can be improved and providing constructive feedback. But there are other, social skills that go along with being a committer. For one, the person needs to demonstrate an adherence to a project's ideals. Within WordPress, this means being fully committed to the project's user-first approach. Committers also need to "[play well in the sandbox](#)." Someone may write flawless code, but if they are unable to work with others they aren't going to make a wonderful committer. It's important to make

good choices about who is given commit access because once it has been given to someone it is difficult to take it away.

In an attempt to avoid an “us and them” mentality in which committers have higher status than everyone else, the WordPress project had only two committers. This meant that everyone was subject to the same review process and the same rules. Opening it up to more people may have created dissatisfaction among those who didn’t have commit access. But by not opening it up at all, developers who felt that they could never progress within the current project structure became frustrated. So few committers in a growing project also meant a mounting number of patches that languished on Trac as they awaited review.

There is a perennial tug of war between those who contribute to the project and those who maintain it, and even among the project leaders themselves. Who gets commit access and for how long? What sort of status is attached to commit? Should only lead developers be committers? What is a “lead developer” anyway? Over the coming years these questions play out at various stages in the project’s development.

[^fn-1] A version control system is a tool for managing the changes to a piece of software, document, or other collection of information. Each change is identified by a number and the name of the person who has made the change. They are often accompanied by comments about what that change is. A version control system allows a project to keep track of changes and also to revert changes as necessary.

[^fn-2] Mosquito was the first bug tracker used by the WordPress project. Later bug tracking moved to Trac.

14. A New Logo

As development progressed, WordPress’ design and branding took shape. Some of the first lengthy discussions about WordPress’ look and feel happened on the [wp-design mailing list](#). This was a private mailing list of designers and developers who were interested in crafting WordPress’ aesthetic. Michael Heilemann ([michael](#)), Khaled Abou Alfa ([khaled](#)), Joen Asmussen ([joen](#)), Chris Davis ([chrisjdavis](#)), Joshua Sigar ([alphaoide](#)), and Matt comprised the main group.

The first focused design discussion was about WordPress’ logo. Matt designed the original logo; it was simply the word “WordPress” in the Dante font:

WordPress

The original WordPress logo

Since the project was so small, community members had fun on the homepage, riffing on the logo with versions for seasonal holidays, birthdays, and other events.





Logos that were created for WordPress.org for special occasions

Eventually, however, WordPress needed a professional logo. Usage was growing and WordPress needed a logo that properly represented it. As a free software project, the community was the first place to look for a logo. Community suggestions were solicited. A mixed set of results came back, which were shared with the wp-design group for feedback.

The [first suggestions](#) were from Andreas Kwiatkowski:



WordPress logo suggestions from Andreas Kwiatkowski

A [second batch](#) came from Denis Radenkovic:



WordPress logo suggestions from Denis Radenkovic

Logo feedback was mostly lukewarm, though [some of the designers](#) liked Radenkovic's heart logos, describing them as "instantly recognizable."

The heart logo was iterated on, with another version posted to the design mailing list:



WordPress

Joen Asmussen's iteration of the heart logo

A version of the admin screen with it in situ was produced.

Blog Title

Welcome back [user] ([Log Out](#)) ([View site »](#))

[Dashboard](#) [Write](#) [Manage](#) [Links](#) [Presentation](#) [Plugins](#) [Users](#) [Options](#)

[Write Post](#) [Write Page](#)

Timestamp updated.

Title
[Text input field]

Excerpt ([help?](#))
[Text input field]

Post Body

[Text area with rich text editor toolbar: bold, italic, link, quote, image, code, Dictionary, Close Tags, More...]. Textarea: [Larger](#) | [Smaller](#)

[Delete Post](#) [Save](#) [Publish](#)

Advanced

Send Trackbacks ([help?](#))
[Text input field]

Custom Fields ([help?](#))

Select
[Select dropdown]

Key
[Text input field]

Value
[Text input field]

Preview
[Text input field]

Post Status
 Published
 Draft
 Private

Discussion
 Allow Comments
 Allow Pings

Password ([help?](#))
[Text input field]

Post Slug ([help?](#))
[Text input field]

Categories
Culture
Books
Movies
Music
Politics
Diary
Guides
Misc.
Nature
Noscope
Currently
Sidenotes
Software
Web
Browser
Bugs
Flash
Usability
Work
Installments
Noteworthy

Timestamp
Post author

WordPress 1.5
[Support Forums](#)

*Joen Asmussen's admin design with the heart logo in situ

Community members weren't the only ones tackling the logo. In March, Matt met [Jason Santa Maria](#) at South by Southwest and asked him to try redesigning the WordPress logo. They shared ideas about what they thought the logo should be: "the things that kept coming up were not only the idea of publishing but the idea of having a personal journal and a personal thing that might have some sort of tactile overtones," [says Jason now](#). "We were making links to things like letterpress and journaling and any sort of older representations of what it meant to publish something in a physical form." In April 2005, [some of the early versions were shared](#) with the wp-design group:



The first logo ideas sketched by Jason Santa Maria

There were a number of responses from the designers on the mailing list: "a little too aristocratic" was one of the comments. The designers felt that Denis Radenkovic's design was more in line with WordPress' brand.

More designs were posted to the group:



Further iterations on the logo design from Jason Santa Maria

The members of the mailing list didn't seem to agree on WordPress' aesthetic. On one hand, there were people who felt that the logo should represent warmth and community, and on the other hand, something classic and elegant. To reach a consensus, discussions happened offline. Khaled [reported back](#):

WordPress is meant to be the Jaguar or Aston Martin of Blogging tools. [...] that line sets the stage for what the design of the branding should be. Elegance, polished, and impeccably [sic] designed is where we should be aiming.

The logo was finally decided on May 15th, when Matt sent an email to the mailing list with the subject "I think this is it." Matt's message contained just one image:



The final design for the WordPress logo

The major change to the logo, other than the new typeface, was the mark. The creation of a mark gave WordPress a standalone element of the logo which, over time, would be recognizable even without the word beside it. This could and would be used in icons, branding, and t-shirts. It's become instantly recognizable, helped by its appearance on WordCamp t-shirts the world over.

15. WordPress Incorporated

As the number of people using WordPress grew steadily, more and more needed to get done. Matt took a job at CNET, which allowed him to work on WordPress alongside his day job. Ryan was working at Cisco, often 60 or 80 hours a week; he would come home to work on his hobby, WordPress. Apart from time spent on the project, there are all sorts of associated costs that come with running a growing project. Server costs, for example, that increase as the project's popularity grows.

At the start of March 2005, WordPress 1.5 had seen 50,000 downloads. Just three weeks later the number doubled to 100,000. To celebrate the landmark, there was a [100K party in San Francisco](#). On March 22nd, a group of WordPressers got together at the Odeon Bar in San Francisco.

Jonas Luster, ([jluster](#)) was at the party. Jonas was an employee at [Collabnet](#), the company behind Subversion. He and Matt had been talking about what a company built around a free software should look like. Matt had an idea for an organization called Automattic: an umbrella company that would include a number of different WordPress organizations. The first of these was WordPress Incorporated. At the party, Matt asked Jonas if he'd like to be involved. Jonas said yes.

Shortly afterward Matt jumped on stage to announce WordPress Inc. -- with Jonas Luster as employee number one. By the following morning, a [couple of blogs](#) had covered the announcement. A few days later, [Matt went on holiday](#) leaving WordPress' very first employee in charge.

Even as the party was going on, trouble was brewing. In February, a [user posted to the WordPress.org support forums](#) asking about hundreds of articles hosted at [wordpress.org/articles/](#). The articles were about everything from credit and healthcare to web hosting. The thread was closed by a forum moderator. Blogger Andy Baio discovered the thread. He contacted Matt to ask

about what was going on.

Web Images Groups News Froogle Local New! more »

site:wordpress.org inurl:articles Advanced Search Preferences

Web

Results 1 - 100 of about 168,000 from [wordpress.org](#) for inurl:articles. (0.06 seconds)

[WordPress](#) > Get More Web Traffic Through Writing Articles

Writing your own **articles** is one of the very best ways for you to get free traffic to your web site. You can also use **articles** to brand yourself and your ...
[wordpress.org/articles/ web-hosting--get-more-web-traffic-through-writing-articles.htm](#) - 9k -
[Cached](#) - [Similar pages](#)

[WordPress](#) > An Article about Articles on the Mortgage Industry

Let's say that you are looking for information on the internet about problems, lawsuits & complaints about or against certain mortgage lenders.
[wordpress.org/articles/ home-buying--an-article-about-articles-on-the-mortgage-industry.htm](#) - 8k -
[Cached](#) - [Similar pages](#)

[WordPress](#) > How Writing Articles Can Boost Your Online Home Based ...

There's one simple technique you could be using to help raise the online profile of your home based business - writing **articles**. When you decided to start a ...
[wordpress.org/articles/ home-business--writing-articles-boost-online-home-based-business.htm](#) - 10k -
[Cached](#) - [Similar pages](#)

[WordPress](#) > Small Business Online Marketing Using Free Articles

Conventional advertising online has never taken off and many small businesses have lost valuable business funds on online advertising and marketing that ...
[wordpress.org/articles/ home-business--small-business-online-marketing-using-free-articles.htm](#) - 9k - Mar 29, 2005 -
[Cached](#) - [Similar pages](#)

[WordPress](#) > How-to Articles to Increase Your Web Site Traffic

Want a sure-fire way to increase return traffic to your web site? Who doesn't?
The best way to ensure that your visitors come back again and again is to ...
[wordpress.org/articles/ web-hosting--how-to-articles-to-increase-your-web-site-traffic.htm](#) - 9k - [Cached](#) - [Similar pages](#)

The Google search results that show the articles hosted on WordPress.org. (Google and the Google logo are registered trademarks of Google Inc., used with permission.)

Matt explained to Andy that he was being paid by a company called Hot Nacho to host the articles on WordPress.org and that he was using the money to cover WordPress' costs. "The /articles thing isn't something I want to do long term," he told Andy, "but if it can help bootstrap something nice for the community, I'm willing to let it run for a little while." In addition to responding to Andy, Matt re-opened the support forum thread and [left a response](#):

The content in /articles is essentially advertising by a third party that we host for a flat fee. I'm not sure if we're going to continue it much longer, but we're committed to this month at least, it was basically an experiment. However around the beginning of Februry [sic] donations were going down as expenses were ramping up, so it seemed like a good way to cover everything. The adsense on those pages is not ours and I have no idea what they get on it, we just get a flat fee. The money is used just like donations but more specifically it's been going to the business/trademark expenses so it's not entirely out of my pocket anymore.

On March 30th, while Matt was in Italy, Andy [published an article about it on his blog](#). The reaction from bloggers was bad: no matter what Matt's intentions, people saw the articles as a shady SEO tactic. A free software project should know better than to work with article spammers. Not only was WordPress.org hosting the spammy articles, CSS had been used to [cloak them](#). If everything was above board, why not do it out in the open? The articles also had an influence on WordPress.org's hosts, Textdrive, who hosted WordPress for free on a server along with apps such as Ruby on Rails. Their servers were overloaded with hits from the links in the Hot Nacho articles.

With Matt out of town, Jonas dealt with the fallout from Andy Baio's article on Waxy.org. It wasn't clear where the [line between WordPress.org and WordPress Inc.](#) lay, but Jonas was the only official-sounding person and was the one who fielded the tech community's ire and he spent the next few days putting out fires. He [posted on his blog asking for calm](#), and that Matt be given the benefit of the doubt.

The Hot Nacho articles were perceived as the first move by WordPress Inc. to make money -- an indication that Matt and Jonas

were't going to monetize WordPress in an ethical manner. Jonas stressed that the articles had nothing to do with WordPress Inc., but since it wasn't clear at that time what WordPress Inc. actually was, people naturally lumped them together. By the next day, Andy had updated his post to report that WordPress' PageRank had dropped to 0/0. WordPress.org had been removed from Google's search engine results.

The story escalated as major tech resources like ([The Register](#), [Slashdot](#), and [Ars Technica](#)) picked it up. On March 31st, [Matt posted briefly on his blog](#) to say that he had just learned of what had happened and that he would get online to address the questions as soon as possible. The comments are littered with messages of support from [people who would later be central to the project](#) and even from Matt's rival. Matt also received [support on WordPress' support forums](#).

On April 1st, [Matt issued a full response](#). In it, Matt explains that he was struggling to find a way to support the free software project, that he felt that his options were limited:

To thrive as an independent project WordPress needs to be self-sufficient. There are several avenues this could go, all of which I've given a lot of thought to. One route that would be very easy to go in today's environment is to take VC funding for a few million and build a big company, fast. Another way would be to be absorbed by an already big company. I don't think either is the best route for the long-term health of the community. (None of these are hypothetical, they've all come up before.) There are a number of things the software could do to nag people for donations, but I'm very hesitant to do anything that degrades the user experience. Finally we could use the blessing and burden of the traffic to wordpress.org to create a sustainable stream of income that can fund WP activities.

The decision to host the articles on WordPress.org was an attempt to mitigate some of the costs of the free software project. For many people, Matt's response satisfied their concerns and questions. [But for others, it wasn't enough](#). If a free software project has to resort to turning its highly ranking website into a link farm, what is the future viability of that project? Aren't there other ways to support it? Matt had been at the [web spam summit](#) in February, which had [specifically addressed](#) "comment spam, link spam, TrackBack spam, tag spam, and fake weblogs." Hadn't he, at that point, realized that there might be something dubious?

As with most storms on the internet, once the articles were removed and Matt had issued his apology the anger subsided. It did, however, have a lasting influence. The experience [changed Matt's thinking about spam](#). Instead of viewing it purely as something that appears in an email inbox, he now saw it as web spam, which can quickly pollute the web. This influenced his thinking when developing Akismet, Automattic's anti-spam plugin. And while Matt had learned a harsh lesson about ethical ways to make money on the web, it would return to haunt him again and again. Any time WordPress.org clamped down on search engine optimization tactics, in themes for example, irate community members would bring up Hot Nacho.

Jonas' first few weeks at WordPress Inc. got off to an explosive start, and over the next few months he continued to work unpaid as WordPress Inc.'s only employee. He got in touch with major web hosts and created partnerships so that WordPress.org could make money by recommending hosting companies. He met with venture capitalists to talk about where WordPress was going.

At an [IRC Meetup in August 2005](#), Matt discussed some of the ways he wanted to see the project supported. Instead of fundraising, he wanted to see a company that was responsible for generating revenue. He said that "the goal, eventually, is for it to be the biggest contributor someday, supporting community members to work on WP full-time." To protect the project from any of the liabilities of the company, donations would be kept separate. This would ensure that if the company was sold, the project would be safe. In this chat, the "inc." that Matt refers to isn't WordPress Inc., but the as-yet-unannounced Automattic. When asked about the employees, he says that the only one is Donncha, with Ryan Boren planned for the future. WordPress Inc. itself petered out, disappearing with much less fanfare than it arrived. There wasn't money available in WordPress Inc. to pay Jonas a salary, especially when Matt was using his own salary from his job at CNET to pay Donncha to work on WordPress.com.

The wider community was [confused about what had happened with WordPress Inc.](#), especially those who had watched Matt on stage announcing the company's launch. In response, Matt [posted a page on the WordPress Codex](#), explaining what had happened:

At the WordPress 100k party in March 2005, I talked about the formation of a "WordPress Inc." with Jonas Luster as the first employee. That never really got off the ground, I continued my job at CNET and Jonas (who I couldn't afford to pay a

salary) ended up going to work for Technorati. It wasn't even planned to be announced at the party (since it wasn't clear logically how it would happen) but everyone was really excited about it and I had an extra G&T (or two), we all got carried away.

The first movement toward making money to support WordPress was more of a stumble than a step. It did produce some important lessons, however, about how to run a business alongside a free software project. In any attempt to run a business based on a free software project, the company is beholden to two players: the company itself and the community. In terms of the company, investors have to be kept happy and employees have to be looked after. The company needs to make money. On the other hand, it is the community that is invested in growing the free software project and assuring its integrity and independence. Walking the line between the two can be difficult for any community member, but particularly for a project founder or lead who has multiple reasons for securing the future of both.

16. WordPress.com

As WordPress Incorporated dissipated, Matt pitched a WordPress-based blogging network to his employers at CNET. Many of the major internet companies at the time had a blogging network, including Google, Blogger, Yahoo 360, and Microsoft Spaces. CNET also owned several domains perfect for a blogging network, such as online.com. CNET decided against the idea but the idea itself didn't disappear. Matt decided to build a blogging network himself. [^fn-1].

First, Matt needed to acquire the WordPress.com domain name. At the time, [Open Domain](#) owned it. Open domain registered free software project domains. They gave projects permission to use each domain in return for a website link on or some other form of acknowledgement. It was unclear to the WordPress community and the wider free software community whether open domain was a [domain squatter or an entity that genuinely wanted to help free software communities](#). The organization also registered Drupal.com, though they [donated the domain to Drupal without major incident](#). The WordPress community was perturbed over someone squatting on WordPress.com, despite good intentions. It was important to acquire the domain, rather than leave it with Open Domain and point the DNS at it. There was little security in a blogging network where a third party held the domain. How could anyone be certain that the third party wouldn't take the site down? After months of wrangling, Matt acquired the WordPress.com domain name.

With the domain name secure, Donncha, as the developer of WPMU, was the perfect choice to start building the network. At the time, his work bored him. He was looking for something more interesting. When Matt emailed the WordPress security mailing list to see if anyone was interested, Donncha got in touch.

When it first started, WordPress.com ran WPMU trunk. When changes were made to WordPress.com, Donncha pushed them upstream to the core software. Development was fast; Donncha worked on two servers and live edited code. Without any users to worry about, he moved quickly. He improved user-focused functionality and built a network administration screen to properly manage the network. The WPMU community helped, submitting patches to clean up admin screens. Features developed for WordPress.com but which didn't necessarily sit well in the WPMU core software were put into plugins like Domain Mapping and Global Tags.

Andy Skelton ([skeltoac](#)) was the second person to work on WordPress.com. Andy had developed a service and plugin called Blogs of the Day. This self-hosted stats plugin produced a list of the most popular blogs each day. The plugin offered website stats and a centralized server that recorded data. This produced the list of most popular blogs. Andy's script loaded the PHP and ran every time there was a hit on someone's blog. After meeting in Seattle, Matt brought Andy on board to help create a stats plugin for WordPress.com. Andy used the knowledge he gained building Blogs of the Day which, for a number of years, was featured on the front page of WordPress.com. This made Blogs of the Day Automattic's first acquisition.

WordPress.com opened to signups in August 2005. The WordPress.com developers were concerned about scalability, so to

control user growth on untested servers, signups were [by invite-only](#). Many people who were involved with the WordPress project got a WordPress.com blog, including [lorelle.wordpress.com](#), for example, and [mark.wordpress.com](#). Every person who signed up got one invite to pass along. Demand quickly outstripped supply; one invite even landed on eBay.

Downloading the Flock browser was the other way to sign up for WordPress.com. Flock was a social-media focused browser with social networking built in. “I thought Flock was the future,” [says Matt now](#). “First we did the invites, and then we thought well, we’ll allow you to bypass an invite if you’re using Flock because then we’ll know that you’re kind of a social in the know cool person.” Flock was integrated with WordPress.com so users could use it to post to their WordPress.com blog. Both of these measures -- invites and Flock -- allowed WordPress.com to grow in a steady and sustainable way. Putting in sign-up barriers controlled the flow of people and helped ensure scalability.

It wasn’t just WordPress.com’s servers that had to scale. New-user influx meant higher demands on support. This, first of all, influenced the WordPress.org forums as users found their way there looking for help. There wasn’t a clear distinction between WordPress.com and WordPress.org and many people who needed support for WordPress.com went to the WordPress.org forums. This caused some discontent among the WordPress.org support forum volunteers, who felt that they were providing free work for a service which would eventually become commercial.

To solve this problem, a support email address was set up for WordPress.com users. Originally WordPress.com’s developers did support, which diverted their attention from writing code to helping users. “I found that I would spend half my day replying to users,” [recalls Donncha](#), “and then being completely wrecked in that your mind is completely taken off programming new features, or improving things, fixing bugs, just because this whole thing is so tiring.” After the mailing list, a community support forum was set up, and eventually a ticketing system.

Mark Riley was active on the WordPress.org support forums. He was hired to do WordPress.com support. For a long time he was solely responsible for support at WordPress.com. The next person hired for support was in 2008. In that time 49,486 support tickets were closed. Nowadays, WordPress.com users have a community on their own forums and a huge team of “Happiness Engineers” supporting users across Automattic, but even today, it’s not unusual for users to land on the WordPress.org forums looking for help.

Creating a user-focused blogging platform like WordPress.com made sense in the context of the WordPress project. WordPress has always been committed to users. It has always had laypeople at its heart, people who don’t necessarily know the mechanics of how a server works but who want a website. WordPress.com was built for people who just want to fill out a form and have a blog, but still have the power of WordPress. This means that WordPress.com is a balancing act. There will always be those who want WordPress.com to be simpler, with less functionality, and there will always be people who want it more like the WordPress.org software, with plugins. To prevent people running code on their website, WordPress.com does not allow users to install plugins. Instead, WordPress.com provides a middle ground for people who don’t want to set up a WordPress website but who wanted to be able to blog.

However, a hosted service like WordPress.com needs to have a business behind it to pay for server costs, run the software, provide support, and build new features. The free software project, with its haphazard donations and income from hosting companies, couldn’t maintain such a network on a wide-scale. This business needs to be focused on WordPress’ main audience: software users. While this business was beginning to come together around the developers who had built WordPress.com, Matt was working on another product that would influence the WordPress community: an anti-spam plugin called Akismet.

[^fn-1] CNET went on to be one of the first investors in Automattic.

17. Akismet

Spam has always been a problem; it's the old annoyance of unsolicited mail in your mailbox, but on a huge scale. In the internet age, spam has gone from the actual mailbox to the virtual mailbox, filling inboxes all over the world. Because blog comment boxes leave a gap open for anyone to enter data, blogs are notoriously susceptible to spam. Any opportunity for user-entered data, whether that's a comment box or trackback, is a doorway for spam.

WordPress, like every other blogging platform, is susceptible to spam. In 2005, there were ongoing discussions about how to deal with comment spam. At the time, there were a few plugins around, such as Spam Karma and Bad Behavior, and Matt was also working on his own anti-spam solution. His first attempt was a JavaScript plugin which modified the comment form and hid fields, but within hours of launching it, spammers downloaded it, figured out how it worked, and bypassed it. This is a common pitfall for anti-spam plugins: once they get traction, spammers pay attention and they quickly figure out how to bypass it. In late 2005 [Matt launched the Akismet plugin for WordPress](#). Akismet (which stands for Automatic kismet) is a way of dealing with spam in aggregate, a crowd-sourced spam-fighting tool.

Each time someone posts a comment to your website, Akismet checks it against all the comments in the database. If it's spam, it's deleted, if not it lands on your site. If spam gets through and a user marks it as spam, the comment is added to the database. So the pool of spam comments is constantly widening, making Akismet into something that is more effective over time. "It's like all the kids on the playground ganging up against a bully," [says Matt](#), "Collectively we all have the data and the information to stop spammers, certainly before they're able to have a big impact."

In November 2005, there was a discussion on the wp-hackers mailing list [about plugins that could be bundled with core](#). An anti-spam solution topped the list. Akismet was discussed, though [not everyone was comfortable bundling a plugin that had a commercial element to it](#). Akismet is free for non-commercial use, but commercial users are asked to pay. This is based on an honor system and commercial users need to be up-front about their website's intentions. Some questioned [the data Akismet collected and how it was stored](#). Additionally, you needed a WordPress.com account to use Akismet, though WordPress.com hadn't officially launched yet. It was still an invite-only service and not every WordPress user had an invite. Those who didn't have an invite could download the Flock browser but not everyone wanted to download a new browser to get anti-spam protection.

Despite some pushback, there was an [equal amount of support](#). Some didn't find the pay-what-you-want honor system too jarring. Akismet is a service and has associated server costs which need to be covered. [In November, Akismet was checked in to core](#). When WordPress 2.0 beta came out that month, it came bundled with Akismet. WordPress users had a new weapon against spam. So that everyone could get the WordPress.com account they needed for Akismet, [WordPress.com opened for public signups](#) two days later.

Concerns continue that a plugin that makes money is shipped with WordPress. People feel that this gives Automattic an unfair advantage. Discussions about whether Akismet should continue to ship with core surface perennially in the community, including among core developers and at core team meetups. "It seems an unfair advantage and it cuts against WordPress' goal of openness," [says Mark Jaquith \(MarkJaquith\)](#), "That being said, spam is still a huge problem and Akismet is still the leading product, even though there are now alternatives."

The open nature of WordPress plugins makes it difficult to develop a widely-adopted anti-spam plugin in which every element of the plugin is public. As soon as a plugin has wide-spread adoption, spammers study it and learn how to bypass it. Dealing with spam via a service means that the code can all be open source but the algorithm for dealing with spam need not be. While there are often discussions about adopting a model in which a number of different anti-spam options are recommended, it isn't yet viable. "The moment we recommend five plugins," [says Andrew Nacin \(nacin\)](#), "the spammers will all target the other four that don't have the ability to evolve and learn like Akismet does." It creates a challenging intersection between business and free software. Including a freemium plugin with the software doesn't gel completely with the software's openness goals, but, at the same time, taking it away could have a detrimental effect on users, which goes against the project's user-first principles.

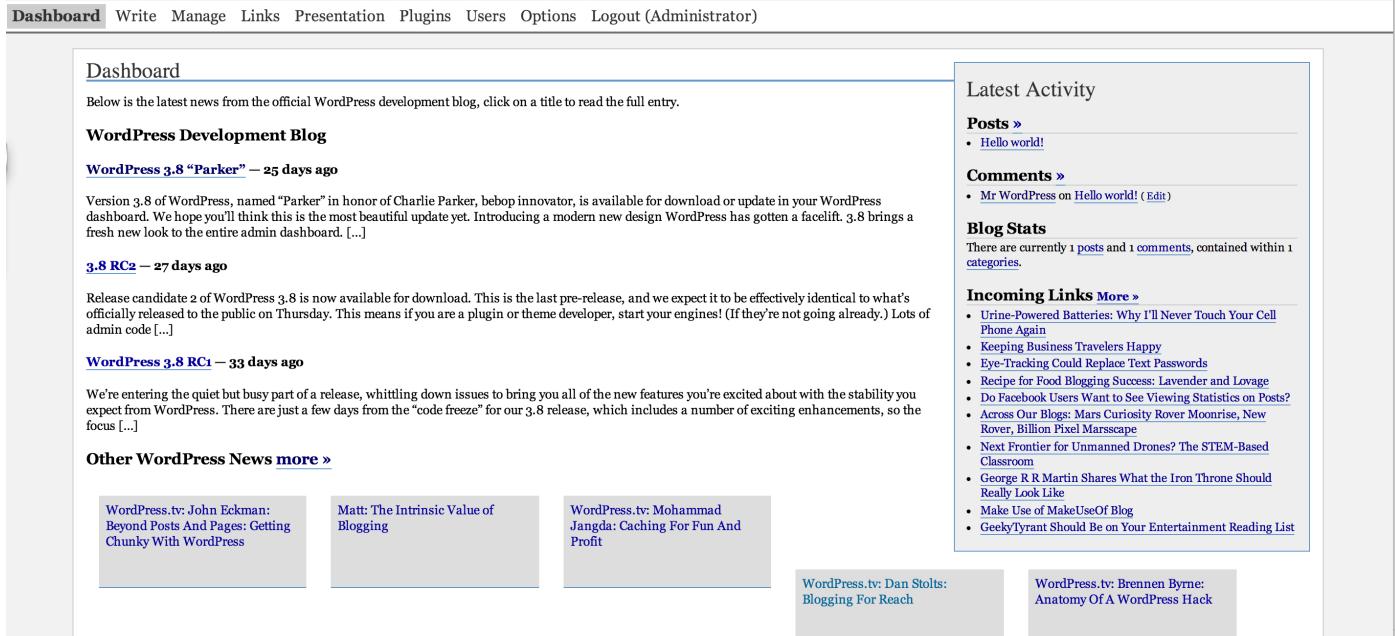
Akismet continues to be bundled with core, and the issue continues to be discussed.

18. Shuttle

As people experimented with how to make money with WordPress, experiments also took place in the interface design. WordPress inherited its look from b2. Between 2005 and 2006, [wp-design mailing list](#) members overhauled the admin screens, in a project called Shuttle. Their aim: to create a coherent brand for WordPress by redesigning wp-admin.

The group focused on an aesthetic refresh to reimagine and modernize wp-admin -- as opposed to a functional refresh -- i.e., iterating the design rather than re-architecting the interface or adding new features. They started work in the wake of WordPress 1.5, which came out with a set of admin screens in need of some love.

Blog (View site »)



The screenshot shows the WordPress admin dashboard under the "Shuttle" theme. The top navigation bar includes links for Dashboard, Write, Manage, Links, Presentation, Plugins, Users, Options, and Logout (Administrator). The main content area is titled "Dashboard" and contains news items from the "WordPress Development Blog". The first item is "WordPress 3.8 ‘Parker’ — 25 days ago", which discusses the release of WordPress 3.8 named after Charlie Parker. The second item is "3.8 RC2 — 27 days ago", announcing the release candidate 2. Below these are other news items and a link to "Other WordPress News". On the right side, there are four boxes for "Latest Activity": "Posts" (with one entry), "Comments" (with one entry), "Blog Stats" (showing 1 post and 1 comment), and "Incoming Links" (listing several external links). At the bottom, there are five smaller thumbnail previews of blog posts by John Eckman, Matt, Mohammad Jangda, Dan Stoltz, and Brennen Byrne.

Unlike WordPress core development, the Shuttle group communicated via private mailing list. This list was open to other members but the list archives weren't public. To get involved, a contributor had to be added to the group, and the group had to agree to add the new member. Discussions among members indicate that they weren't keen for too many people to join.

This highlights one of the tensions between design -- which is usually a private process that leads to a finished design presented to the world -- and free software development. Linus's Law ([as outlined by Eric Raymond](#)) is that "given enough eyeballs, all bugs are shallow." If there is a problem, get lots of eyes on it; it will be transparent to someone. This approach works with bugs or a coding problem with a clear answer.

Design, however, is much more subjective, and the Shuttle project designers felt that opening up the design process to anyone would lead to a "too many cooks" situation. "A mailing list like that," [recalls Chris Davis now](#), "would become so much noise and so little signal so quickly that there would be no way we could move forward."

To avoid this scenario, the group remained small, with three main designers (Michael, Joen, and Khaled) and coders responsible for realizing the design vision. The group worked from a set of PSDs and sent designs around among themselves. They offered feedback on each other's work and iterated on the design. They worked on design specific elements, mainly the Post screen (`post.php`). Over the project, 28 different versions circulated via the mailing list.

Version 8

Blog Title

Welcome back User! ([Log Out](#)) ([View site »](#))

Dashboard Write Manage Links Presentation Plugins Users Options

Write Post [Write Page](#)

! Timestamp updated.

Title

Excerpt [\(help?\)](#)

Post Body

b i link "quote" code Dictionary Close Tags [More >](#) Textarea: [Larger](#) | [Smaller](#)

[Delete Post](#)

[Save](#) [Publish](#)

Advanced

Send Trackbacks [\(help?\)](#)

Custom Fields [\(help?\)](#)

Select

Key

Value

Preview

Post Status

- Published
 Draft
 Private

Discussion

- Allow Comments
 Allow Pings

Password [\(help?\)](#)

Post Slug [\(help?\)](#)

Categories

- Culture
- Books
- Movies
- Music
- Politics
- Diary
- Guides
- Misc.
- Nature
- Nosope
- Currently
- Sidenotes
- Software
- Web
 - Browser
 - Bugs
 - Flash
 - Usability
- Work
- Installments
- Noteworthy

Timestamp

Post author

WordPress 1.5
[Support Forums](#)

Version 14

[Dashboard](#)[Write](#)[Manage](#)[Links](#)[Presentation](#)[Plugins](#)[Users](#)[Options](#)[Write Post](#) | [Write Page](#)

Title

Excerpt [\(help?\)](#)

Post Body

b i link "quote" code Dictionary Close Tags [More »](#)

Textarea: [Larger ^](#) | [Smaller ^](#)

[Delete Post](#)[Save](#) [Publish](#)

Advanced

[\[\]](#)Send Trackbacks [\(help?\)](#)Custom Fields [\(help?\)](#)

Select

Value

Key

Preview

[\[\]](#)

WordPress 1.5
[Support Forums](#)

Version 21

Dashboard Write Manage Links Theme Plugins Users Options

Write Post Write Page

! Timestamp updated.

Title
[Text input field]

Excerpt [\(help?\)](#)
[Text area]

Post Body
[Text area with rich text editor toolbar: bold, italic, link, quote, code, Dictionary, Close Tags, More...]. Textarea: [Larger »](#) | [Smaller «](#)

Post Status
 Published
 Draft
 Private

Discussion
 Allow Comments
 Allow Pings

Password [\(help?\)](#)
[Text input field]

Post Slug [\(help?\)](#)
[Text input field]

Categories
 Culture
 Books
 Movies
 Music
 Politics
 Diary
 Guides
 Misc.
 Nature
 Noscope
 Currently
 Sidenotes
 Software
 Web
 Browser
 Bugs
 Flash
 Usability
 Work
 Installments
 Noteworthy

Delete Post [Save](#) [Publish](#)

Advanced
Send Trackbacks [\(help?\)](#)
[Text input field]

Custom Fields [\(help?\)](#)
Key: admin (Michael Heilemann) Value: [Text input field]

Timestamp [\(help?\)](#)
Post author [\(help?\)](#)

Preview [\(help?\)](#)

Shuttle
Welcome back Stanley.

[Log Out](#) [View Site](#)

[Dashboard](#) [Write](#) [Manage](#) [Links](#) [Themes](#) [Plugins](#) [Users](#) [Options](#)

[Write Post](#) [Write Page](#)

! Timestamp updated.

Title

Excerpt [\(help?\)](#)

Post Body

Textarea: [Larger](#) | [Smaller](#)

B **I** **ABC** | | | |

[Delete Post](#) [Save](#) [Publish](#)

Advanced

Send Trackbacks [\(help?\)](#)

Custom Fields [\(help?\)](#)

admin (Michael Heilemann)	
Key	<input type="text"/>
Value	

[WordPress 2.0 Support Forums](#)

As the design process continued, elements of Shuttle were implemented in WordPress. One of the earliest Shuttle designs increased the size of the title field in the `post.php` edit screen.

In WordPress 1.5:

Write Post

Title

Post

TrackBack a URI: (Separate multiple URIs with spaces.)

In Shuttle:

Title

Excerpt [\(help?\)](#)

Post Body Textarea: [Larger](#) | [Smaller](#)

[► Advanced Options](#)

WordPress 2.0:

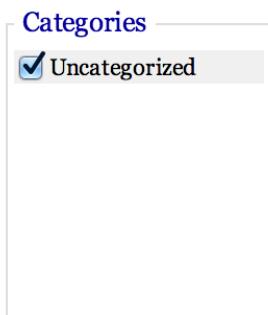
Write Post

Title

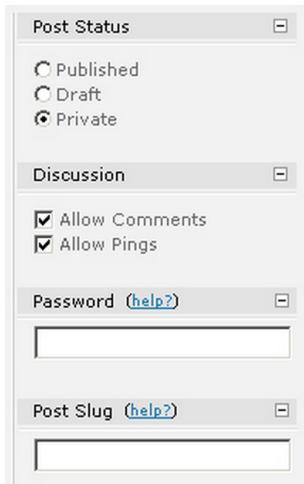
Post

Pods in the `post.php` screen, which collapsed elements such as post status, categories, and author, was another iteration.

In WordPress 1.5:



In version 8 of Shuttle:



In WordPress 2.0:



When WordPress 2.0 shipped with Shuttle-inspired changes, the feedback wasn't entirely positive. [Molly Holzschlag](#) wrote that "what WP2.0 has gained in interface appeal it's lost in some practicality too." For the Shuttle team, the problem with WordPress 2.0 was the [piecemeal implementation of their vision](#).

The project was beset by other problems. Despite having a closed mailing list to avoid the "too many cooks" problem, that problem arose anyway. The group's structure made no one person responsible for the project's overall vision. Ideas were discussed and sent around. When one reads the mailing list now it feels like a design-focused discussion forum rather than a team with a clear task. As the group discovered, having a group of designers, each with their own ideas, can cause as many problems as opening a design mailing list up to a bunch of hackers.

It took the group a long time to complete work. They discussed minor design elements such as rounded corners and gradients

for lengthy periods, rather than pose fundamental questions about WordPress users' needs and wants. "I don't know if we were cooperating enough on getting a unified feel and a unified understanding of everything before we were trying to actually apply our ideas to the problem," says Michael now. Besides that, the contributors had jobs that absorbed their time, so work happened in fits and starts. While the original plan was to complete the admin redesign within three months (by mid-April 2005) this slid to September. The team eventually missed the deadline for WordPress 2.0 in late 2005. The next deadline (for inclusion in WordPress 2.1, which itself never materialized) was the end of January, though it was March 2006 before a complete set of mockups arrived.

It was then that Khaled sent out a complete set of screenshots with his vision for the new WordPress admin:

The screenshot shows the WordPress 2.1 admin dashboard. At the top, there's a blue header bar with the title 'Welcome back Stanley' and links for 'view site', 'log out', and a lock icon. Below the header is a navigation menu with 'Dashboard', 'Write', 'Manage', 'Links', 'Themes', 'Plugins', 'Users', and 'Options'. A 'Your WordPress' section on the left includes a 'Welcome to WordPress' message, a sidebar with links like 'Write a post', 'Change your site's look or theme', and 'Add a link to your blogroll', and a 'WordPress Version' section with a 'Update Available' button. The main content area features several widgets: '5 Most Recent Posts' (listing 'Essential Firefox 1.0 Extensions', 'Sigur Rós - Glósóli Video', 'Hurricane Rita', 'On Posting Music Samples', and 'Random Photo'), a 'You have 24 Comments in the moderation queue' notification, 'Scheduled Entries' (listing 'Essential Firefox 1.0 Extensions in 20 days' and 'Sigur Rós - Glósóli Video in 2 years'), 'Your Draft Entries' (listing 'Essential Firefox 1.0 Extensions', 'Sigur Rós - Glósóli Video', 'Hurricane Rita', 'On Posting Music Samples', and 'Random Photo'), and an 'Incoming Links' section (listing various external links from sites like hellfroze, Broken Kode, Planeta WordPress, ericsetiawan.com, Binary Bonsai, and hellfroze). At the bottom, there's a 'Shuttle Blog Stats' section showing current post and comment counts, and a footer with links for 'Open WordPress News', 'support forums', and 'documentation'.

Welcome back Stanley

view site log out

[Dashboard](#) [Write](#) [Manage](#) [Links](#) [Themes](#) [Plugins](#) [Users](#) [Options](#)

[Write Post](#) [Write Page](#)

About Posts
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonumy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Adding Images
Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at

Adding Links
Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum

Password Protecting Posts
Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum

Title of post

Post Status
 Published
 Draft
 Private

Discussion
 Allow Comments
 Allow Pings

Write

Save **Publish**

Upload Image **Browse Images** **File** **Upload** **Cancel**

Title **Description**

Categories

- Culture
- Books
- Movies
- Music
- Politics
- Diary
- Guides
- Misc.
- Nature
- Noscope
- Currently
- Sidenotes
- Software
- Web
 - Browser
 - Bugs
 - Flash
 - Usability
- Work
 - Installments
 - Noteworthy

Timestamp **Post author**

WordPress 2.1 support forums | documentation

The rest of the group loved the designs. They set the developers to coding. As with the rest of the project, development dragged on. In mid-April, [Michael Heilemann withdrew from the project](#), saying that he had to prioritize other commitments. The same month, [Khaled asked](#) whether Matt or Ryan would ever get around to implementing the design. [The response](#) placed the redesign as a medium priority. Changes would be iterative.

On May 14th 2006, [Khaled posted to his blog](#), finally sharing the complete set of designs with the WordPress community, bringing the Shuttle project to a close. He was still under the impression that the mockups would be implemented in due course. They never were. Khaled and other members of the group felt disenfranchised. They drifted away from the community. Chris Davis and Michael Heilemann made the switch from WordPress to the Habari project (which was conceived in September 2006).

The biggest failure around the Shuttle project, wasn't the designs, nor even design implementation, but the process itself. To solve one problem, the group closed itself off from the community but in doing so new problems arose. Cut off from the rest of the community they lost touch with the development process. The project's closed nature meant that when life took over and activity slowed there was no opportunity for another enthusiastic party to take on the work. The wider community had diverging feelings about the project -- its secret nature gave it a sense of mystery. There was hype and an expectation for a spectacular WordPress redesign. But for others there was growing resentment that a blessed group of designers was working on something that everyone had a stake in.

In one of the final emails on the wp-design mailing list, [Matt outlined](#) some of the things that he learned about design-oriented free software projects:

- Work should not be done in private
- Design by committee doesn't work, better to break up tasks and let individual people focus on one section
- Focus on lots of incremental changes, rather than giant redesigns (you end up in the same place, and probably sooner)
- Document the process and decisions along the way
- Code concurrently with the designs (and iterate)
- Don't hype it, expectations get out of control
- Avoid scope creep of features into designs
- Set a deadline and stick to it

These things influenced the relationship between design and development, and future design projects tried to avoid the difficulties faced by the Shuttle group.

19. Automattic

WordPress.com and Akismet needed a home -- a company to cover costs, with dedicated employees to work on the projects. WordPress, like many free software projects, started out as a script built in a hacker's spare time. The code attracted people with a common interest. The community grew. More and more users adopted the software. And then the hacker realized that it was more than just a hobby, it could be a business.

A free software project's business model determines the success or failure of that project, and vice versa. The business and the project are interdependent. The business generates income and provides contributors and support to the project, while the project creates the software which forms the business' foundation. The business needs to grow in a non-destructive, sustainable way, allowing the project to grow, mature, and continue to attract a diverse group of contributors.

Running a business alongside a free software project presents challenges that a traditional business doesn't face: on one hand, the business must consider investors and employees, on the other, the community of contributors that have no stake in the business but a huge stake in the project. Decisions come under scrutiny from both sides with differing perspectives. Walking the path between the two is a balancing act, and if the founder of the company also happens to be the founder of the project it can become a tightrope.

In 2005, as Matt considered setting up a business around WordPress, there hadn't yet been a large number of businesses run alongside free software projects. But there were a few, such as Red Hat and the Mozilla Corporation, which had its formative years in 2003-2005, about the same time the WordPress project was taking its first steps.

What companies based around free software projects all share is a belief in the power of free and open source software. They produce free software (both free as in freedom, and as in beer), and have found ways to generate income while keeping the core software free. Making money is just one part of their aim: many want to popularize free software for the benefit of society. And so perhaps it made sense for a business to spring up alongside WordPress, one that shared its commitment to the open web and to democratize publishing, a business that would help support and sustain the project.

In December 2005, as the memory of WordPress Inc. faded, [Automattic officially launched](#) as a new home for WordPress.com, Akismet, and the developers who built them. It marked a beginning for its new employees, who now had to balance the free software project's aims with the company's commercial concerns. It's a balance that has affected both WordPress and Automattic throughout their close histories.

From the beginning, Automattic has espoused the open source principles of distribution, openness, and collaboration. As the project grew so it supported Automattic, and Automattic put resources back into the project. The company launched with four employees -- Donncha O'Caiomh, the original developer of WordPress MU, worked on WordPress.com's infrastructure, Ryan Boren, Matt, and Andy Skelton. They left their jobs and put their faith in WordPress -- that it could grow beyond its roots as a

small project to a platform that could sustain a blogging business.

In January 2006, Toni Schneider joined as CEO, or "adult supervision." Toni was a developer who became a CEO at OddPost, a startup which was successfully acquired by one of the net's giants. OddPost became the basis of Yahoo! mail and Toni became a vice president at the company. After setting up the Yahoo! Developer Network, Toni left for new challenges at Automattic.

It was at OddPost that Toni had had his first exposure to the power of open source. At a time when OddPost didn't have a spam filter, Paul Graham came out with his idea of [Bayesian spam filtering](#). He open sourced the idea, and Toni's team implemented it at OddPost. "It just showed me that this is a really powerful model," [he recalls](#). He was attracted to Automattic by the challenges of building a product around free software. WordPress already had brand recognition but no one was yet making money from it. In a [2006 interview he said](#):

WordPress is an interesting new challenge because it's not like most startups, where the world still hasn't heard of you.

WordPress is way past that stage. On the other hand, there is no business yet. Until Automattic came along, there was nobody working for it. It was all volunteers. So taking that product momentum and somehow turning that into a business will be really interesting.

Other aspects of the project attracted Toni: this was a product that wasn't just a central, proprietary service, it wasn't owned solely by anyone, but by all of the people involved, and users could get involved in the process and have a stake in it -- in a way that they couldn't with many of the other hosted services.

The fact that anyone can set up a WordPress blog is powerful. People are proud that they can get it to work. Non-developers can set up WordPress on a server and feel empowered. It's far different from just signing up to a service like Gmail or Facebook; WordPress is something that you make your own.

Automattic brought together the founders' experience in free software development and Toni's experience in business. There were certain things that worked in the project that they wanted to maintain within the company -- principles of open source development that underpin Automattic today.

The first principle was and is a distributed workforce. Contributors to a free software project come from all over the world and collaborate online. They find or build tools that suit their needs. The internet is perfect for this sort of development. Automattic wanted to stay distributed, enabling it to hire people from all over the world, based on their skills, not their location. [Toni recalls how](#), in the early days of Automattic, people expected the distributed nature of the company to break down. It goes against traditional business wisdom which keeps employees supervised and in one place, with a focus on hours instead of output.

The early employees who came from the WordPress project reinforced the company's distributed nature; they were already experienced at working in a distributed environment. Those who found it difficult to work this way didn't last at the company. Part of the recruitment process ensures that potential hires understand what it's like to work distributedly. Eventually, Automattic developed a hiring process in which potential employees do a trial, working in the distributed environment to see if the company is the right fit.

Automattic has a rapid release model, in which developers constantly push small releases straight to the user. They eschew the traditional [waterfall model](#), a development process which involves a strict sequence: drawing up specifications, design, construction, integration, testing and debugging, installation, and maintenance. At Automattic, developers write patches, build features, and deploy them straight to users.

This [continuous deployment](#) model meant that by May 2010, Automattic had over 25,000 releases, averaging around 16 releases a day. Rather than optimizing for perfection, the process optimizes for speed. Developers break down features into small components and launch code incrementally. Developers and designers can quickly push their products to millions of users within a few seconds.

It also helps to ensure that the people who build the product have a direct connection to the customer, doing away with as many levels of mediation as possible. This development method keeps product people happy because they get to constantly push new ideas to customers. It keeps customers happy because they're constantly getting new toys to play with and their feedback helps to refine the product.

Following the initial launch of Automattic, the bulk of hiring was done from within the free software project. Seven of the first ten employees of Automattic were from the WordPress community, all of whom had an intimate knowledge of both developing and using the software. Automattic had, at its fingertips, a wealth of developers who were not just experienced in the product, but who were passionate about it and who were committed to it, evidenced already by the considerable volunteer time they'd put into it.

They did need to be careful that the community would not become disenfranchised from the software. From the very start, it wasn't clear what was part of Automattic and what was part of WordPress. Matt and Ryan, the two developers who led WordPress, were both employees of Automattic, which appeared to make it an Automattic project. It was unclear even on the original [Automattic about page](#), which lists WordPress as an Automattic project.

Perhaps this was because, at the start, before WordPress Inc. was even launched, Automattic was intended as an umbrella company for a number of different organizations, that would include Moebius Ltd. and WordPress Inc. It was also true that, at that time, much of the work on WordPress was being done by employees at Automattic, particularly through Ryan Boren.

The confusion was compounded by the name of Automattic's blog network: "WordPress.com." "We gave the company this advantage of being able to call its service WordPress.com," [says Toni](#). "It's been a curse and a blessing." On one hand, it confuses everyone from the press to WordPress. Many people just don't care that there is a difference. The mainstream tech press, for example, frequently describes Automattic as the "[maker of WordPress](#)," which does a disservice to the hundreds of contributors who aren't employed by Automattic. The WordPress.org support forums are littered with questions about WordPress.com, and WordPress.com users are confused as to why they can't install plugins.

On the other hand, Automattic has put millions of dollars into growing the WordPress brand and introducing the product to people who would instead have gone to Blogger or Tumblr or some alternative blogging service rather than going down the self-hosted route. A survey carried out in 2013 by WordPress hosting company WP Engine found that 30% of respondents had heard of WordPress. It's unlikely that the free software project would have had the money or the drive to do such extensive branding. There are many people whose first exposure to WordPress is via WordPress.com, and some of those eventually move to the self-hosted software.

Automattic experimented with different ways to make money. Creating an enterprise version was only considered in passing -- since WordPress itself can run a small personal blog or a huge blogging network like WordPress.com, producing an enterprise version didn't make sense. Instead they offered support services to enterprise-level clients. Big, prominent sites such as CNET, About.com, and the *New York Times* were using WordPress, and other sites, such as GigaOm and TechCrunch, were shifting to WordPress.com.

Automattic courted these big sites, initially as a marketing strategy. There is nothing that says "WordPress can scale" better than having big, high-traffic sites hosted on it. If they run WordPress successfully, then it's the best possible illustration that WordPress can run high-traffic, complex websites. Toni approached big websites to offer to host them on WordPress.com. The first included Scobleizer and the Second Life blog.

What many of these large sites needed was dedicated support from WordPress experts. At the time, Automattic saw Typepad as its competitor, which charged \$49.95 for their enterprise version. Automattic tested different subscription levels. When they [launched the Automattic Support Network](#) their subscription level was \$5,000 per year. Also in 2006, they offered enterprise-level hosting -- [when WordPress.com VIP hosting launched](#) it was priced at \$250 per month.

But just because software is free, it doesn't mean that services around that software have to be inexpensive. Companies are prepared to pay enterprise prices to ensure that their website scales and stays online. Automattic, and other companies in the WordPress ecosystem, just had to develop confidence, build up their pricing structure, and charge what they were worth.

20. Growing Pains

In January 2006, a Google authoring survey showed that [WordPress powered 0.8% of websites](#). This usage growth meant the project had grown too, and with that growth came growing pains. Growing pains are common for organizations -- and especially for bazaar-style communities in which everyone is encouraged to participate and have a voice. Every free software project experiences conflict. In the end what matters isn't conflict specifics, but how the conflict is resolved and how the project moves on. Many of the difficulties within the WordPress project come down to a few broad issues: different approaches to development and community building, communication, and differing opinions on who the software is for -- users or developers, for example, or users and business owners. Tensions arose from the wealth of different contributor backgrounds -- from formally taught developers, self-taught hackers, designers, people doing support and writing docs, to business owners and bloggers. Diversity is one of WordPress' strengths, but it doesn't come without challenges, and it's along these frontiers that many issues in the project have arisen.

While the Shuttle project worked in private, contributors were concerned that work was going on behind closed doors. They felt cut out of an important part of WordPress' evolution, although [Matt relayed that](#) "Shuttle is a few designers who throw photoshop comps back and forth to each other."

Co-founder Mike Little was one of those vexed by the process. In June 2005, [changes were checked in to the write screen](#), introducing the new pods functionality, which enabled users to collapse and expand meta boxes on the screen. The changes weren't discussed ahead of time on wp-hackers or IRC, just on the closed wp-design mailing list. This precipitated a long discussion on wp-hackers about [openness in the project](#). While Matt intended the commit to be a discussion point, it was perceived to be a wholesale change made without discussion. Many in the community felt that discussion should happen ahead of the commit, or that the community should have been informed that the Shuttle group was working on interface design. That was, after all, how free software project should work.

[WordPress IRC chat room](#) discussions focused on the fact that only Matt and Ryan had commit access. All issues had to go through them. Discussions ensued about how to make the process more open. As a result both Mark Jaquith and Sean Evans ([morydd](#)) became bug gardeners. They received maximum privileges on Trac.

Inline documentation was another area that generated heated debate. There was strong community support for improving WordPress' inline documentation. For many developers, digging into the code is their best learning tool. In February 2006, Rich Bowen ([drbacchus](#)) proposed that he document WordPress' functions and asked if there was a philosophical objection to having inline documentation. [Carthik responded that](#) inline documentation was frowned upon because it would lead to bloat. He referred to a thread from May 2005, [which said](#):

Commenting is tricky. Some "well-commented" code I've seen had a bunch of lines of repetitive filler that "documented" what you could easily see by just looking at the code itself and doubled the size of the program. APIs should be documented religiously, but I think spending a ton of time on redundant comments for code only a few core hackers will ever look at will bloat the codebase and waste everyone's time. I also believe that well-written code usually doesn't need comments unless it's doing some sort of voodoo or workaround.

The inline documentation proposal, however, had wide community support; some developers offered to [take on the labor themselves](#). Rich's initial proposal started a long discussion about the best way to generate documentation from the code and the best inline documentation format, but the conversation [ultimately didn't go anywhere](#). Matt argued that documentation shouldn't be auto-generated, pointing to the Codex as the best place for documentation.

Over the coming months [inline documentation received community support](#). Tickets were opened but left ignored, and Owen Winkler created his own [developer function reference](#):



Functions

Some functions are documented here.

There are several ways to access these functions:

1. Add the function name to the url:
<http://redalt.com/fn/{function name here}>
2. Add the file name to the url:
<http://redalt.com/fn/file/{file name here}>
3. Start typing in the form below.

Function Search

Criteria:

Search Scope:

- [wordpress_source](#)
- [wordpress_moved](#)

Results

Current statistics:

- Total functions: 1278
- Documented functions: 6.42%
- Total parameters: 1425
- Documented parameters: 3.65%
- [Please help!](#)

This function documentation, the functions themselves, and any incorporated Codex data are licensed under the GPL. This license extends only to the content of the function reference pages of RedAlt, and not

The inline docs proposal's demise made many community members unhappy. Scott Merrill ([Skippy](#)) outlined his issues on the wp-forums mailing list. The docs at that time, were not in a good condition. After the first flurry of activity on the docs mailing list, activity died down and some months saw [barely any posts at all](#). Documentation did continue to grow but in a haphazard and messy way. Using a poorly-managed wiki as a host for documentation that should be inline would only make matters worse.

Tensions also arose between those from a traditional coding background and those from the self-taught hacker background that embodies so much of WordPress' culture. While the line isn't clear-cut, many of the original WordPress developers were entirely self taught. The get-it-on-the-screen approach pioneered by Michel in b2 continued to filter down through the project, often at the expense of coding practices that more experienced developers took for granted. Inline documentation was seen by some as superfluous to writing code, considered even to be bloat as it increased the file size. It would take some time before it became standard practice in the WordPress community, though when it did there were community members who pushed hard to get WordPress properly documented.

The documentation problems weren't limited to inline documentation. Mark Riley one of WordPress' longstanding forum moderators, [pointed out in 2006](#) that the WordPress Codex was failing users. For example, the [WordPress Codex](#) recommended that users hide which version of WordPress they were running -- but the code itself contained a comment asking users to leave the information for statistical purposes. Mark also complained that the Codex's content could be confusing to WordPress users. He cites a section on permissions which states "All files should be owned by your user account, and should be writable by you. Any file that needs write access from WordPress should be group-owned by the user account used by the webserver." Because it was so easy for non-technical users to download and install WordPress, Mark argued that the documentation needed to be

What is Red Alt?

Red Alt is an outlet for [Owen Winkler](#)'s web tools and provides a consolidated and organized archive of other online web development tools and resources.

[Plugin.com: digital art tools](#)
[Content for your site and blog](#)
[PPC Management](#)
[Free WordPress Blog Hosting](#)
[WordPress Hosting at \\$6.95/mo](#)
[Free custom web design quote](#)
[Hire a web designer](#)



[» Blogs that link here](#)



written in a language that they could understand.

These discussions highlighted the tension between developers and non-developers on the project. WordPress encouraged people from all backgrounds to get involved with the project but developers weren't always interested in supporting them. This aligns with the free software do-it-yourself attitude. Many developers in the project were self taught -- if they could understand PHP why couldn't others? The tensions between developers and the support team is played out on the wp-hackers mailing list. Again and again, Mark Riley posts to the mailing list with questions that come up on the support forums, trying to get help from developers. There is a sense of frustration as he tries to get the answers from the developers that the users need.

Release dates were a clear fault line between developers and support. Almost every release cycle, Mark asked to be informed of the release date in advance so that he could ensure that the support forums were properly manned when the new version came out. Again and again, a release would arrive and surprise him. "All I am trying -- and yet again complete failing -- to ask," [he pleads](#) "is that IF you want support for the product it would nice to at least let the forums know. You guys just don't see it do you? You really don't have a clue." This feeling of frustration is clear as community members try to get the information that they need to properly support WordPress users.

These sorts of challenges are unsurprising when so many people come together to work on a project. A free software project is a melting pot of people with different ideas, opinions, and backgrounds. Conventional wisdom says that this approach to running a project and creating software shouldn't work, but somehow it does. Despite the challenges that come with the bazaar-style model, people stick around and help the software and the project to grow.

21. WordCamp 2006

The WordPress community has a long tradition of getting together. Even before WordPress held official events, there were small gatherings of WordPress community members and users. The first of these was [in June 2004, in San Francisco](#). As well as meeting up to socialize, [people held upgrade parties](#), before the days of the one-click upgrade, when users had to upgrade their sites using FTP. As well as helping users to upgrade, contributors also helped people to migrate from other platforms to WordPress.

Many free software projects host dedicated user conferences, where community members get together to learn, talk about the software, and socialize. Matt, along with Tantek Çelik, had been involved in setting up another conference series called Barcamp. Barcamps are informal, user-generated conferences focused on web technologies. They're normally open, and workshop-style -- the attendees help create the schedule. In July 2006, Matt announced that he would be [host a Barcamp style event called WordCamp](#). "BarCamp-style" is a code phrase for 'last minute.' ["he wrote](#).

On July 9th he announced that the event would be on August 5th, and as yet had no venue or schedule. When they did get a venue, it was the Swedish American Hall. More than 500 people signed up for the event, and people came from all over the world: Donncha flew in from Ireland and Mark Riley from the UK. As well as attending the WordCamp, those who had started work at Automattic got together for a meetup to work on some Automattic projects.

The [WordCamp schedule](#) reflects the project's concerns at the time and also the breadth of people involved in the project. Mark Riley gave the first ever presentation at a WordCamp on how to get involved with the project. These are a staple of many WordCamps as community members try to inspire users to help out. Andy Skelton presented on the widgets feature that he was working on for WordPress.com. There were presentations about blogging and podcasting, about journalism and monetizing your blog. Donncha spoke about WPMU, and Mark Jaquith presented on [WordPress as a CMS](#), which he later reported was [one of the most-requested sessions](#).



The first WordCamp (cc license Scott Beale ([Laughing Squid](#)))

WordCamp San Francisco 2006 was the first WordCamp at which Matt did his State of the Word presentation. This has become a feature of every WordCamp San Francisco. Matt talks about what's happened in the past year and what's upcoming for WordPress. In the first State of the Word, he [talked about how](#) the software needed to be kept simple. The plan was to simplify installation, and to make themes and the admin page more usable. He mentioned Mark Riley, who had been extremely helpful on the support forums, and said that documentation and support needed a lot more contributors. Matt discussed updates in the Q&A afterwards -- a major issue. Matt referenced Firefox, which alerted users to each new release. To know of a WordPress update, users had to follow the development blog. WordPress needed to be easy for people to update.



Matt giving his State of the Word presentation (cc license Scott Beale ([Laughing Squid](#)))

As well as the presentations, there was barbecue for lunch and a party in the evening, and the event, despite the short lead time, was a success. It was the first of a WordCamp movement that ignited in countries across the world. The second WordCamp was in 2007, again in San Francisco in the Swedish American Hall. The first WordCamp outside of San Francisco was WordCamp Beijing in September 2007, which was quickly followed by WordCamps in Israel, Argentina, and Melbourne, Australia. These events -- and WordPress events just like them -- were organized by volunteers in the community who wanted to get WordPress users together to geek out on the software. In the beginning, there was no formal structure to the WordCamp program -- if someone wanted to organize one they just did it. They were very informal user conferences, much like Barcamps themselves. It wasn't until later, when the project had grown and the number of WordCamps across the world had exploded, that WordCamps started to get some structure.

22. Speeding up the Release Cycle

Shuttle wasn't the only aspect of the project dragging on. More than a year elapsed between Shuttle-inspired WordPress 2.0 and the next version, WordPress 2.1. Matt describes the year between 2.0 and 2.1 as "a dark time in WordPress development history, a lost year."

WordPress' release cycle was often sporadic, particularly in those early days. Sometimes releases arrived every few months, other times longer. Between 1.2 and 1.5, for example, 265 days elapsed. The codebase had so many changes that they decided to skip a few version numbers, going straight from 1.2 to 1.5. This was at a time when version numbers were attached to releases in a haphazard way. The project's approach is that the current release is the first two numbers, and each release is a major version. Versions 1.5, 2.0, 2.9, 3.4, and 4.0, all have the same weighting as each other. Patch releases have an additional decimal point -- 2.0.1, for example. This differs from the approach taken by many other software projects. Drupal 6 and Drupal 7, for example, are major releases, but the releases with decimal points are not. WordPress uses this approach to avoid version

inflation. Over a period of years, the software version number can quickly get high. Decimalization avoids that.

Between WordPress 1.5 and 2.0, however, the project hadn't quite settled on its approach to version numbering. Version 1.6 was inflated to 2.0. partly because of the number of features in it, and because 314 days had elapsed between releases. But the delay between 2.0 (released December 31st, 2005) and 2.1 (released January 22nd, 2007) had been the longest so far.

The mailing list was active, and more developers than ever were attracted to the project. WordPress itself was downloaded 1.5 million times. But while emails were being sent, no code was being shipped. The [release post for WordPress 2.1 with its a plethora of new features](#) demonstrates the problem. There wasn't a rift in development or any major changes in the project, just a constant desire to slip in just one more feature, a refusal to wait for the next release. The release may have been packed with features but it took more than a year to get there.

The delay frustrated developers not simply because the release cycle was taking so long but because commit access was still restricted to Matt and Ryan. For many developers, the development delays could partly be attributed to the restrictive commit policy. There was a clear difference in opinion about how commit access in the project should be managed. Every free software project approaches commit access differently, and in WordPress itself there was a mixture of people who supported the Linux-style commit policy and those who didn't. Ryan argued that [the funnel](#) avoided situations in which "bridge burning arguments happen in the repository rather than in ticket comments." This way, approach and implementation discussions took place before code touched the repository. Once code lands in the repository it's hard to remove. An open commit policy brings its own challenges and some who'd been involved with open commit free software projects [supported the funnel](#). "Loose control leads to spaghetti code and flame wars over inconsequential issues," wrote Douglas Daulton, in a post to wp-hackers.

In the context of these discussions, Ryan codified elements of the project's process that remain part of the development process today:

- every commit must be accompanied by a ticket.
- before commit, the code is reviewed by at least two people -- one for a code review and the other for an integration/architecture review.
- these reviews are carried out by trusted contributors.

By March, 2006, incremental, positive changes were happening. Ryan suggested that a #wordpress-dev channel could help facilitate discussion around development. The #wordpress channel had become a place for general chatter and support. This meant that the focus wasn't primarily on development. Mark Jaquith [pointed out the Bug Gardening scheme's success](#) which had started in 2005 during previous discussions about opening up commit. By opening up Trac privileges to "bug gardeners" who were responsible for triaging tickets the workload had become distributed, leading to "a whole slew of people who are autonomously fixing bugs, submitting patches, and having them committed." The Trac mailing list was another positive change, which transformed Trac from simple bug-tracking software to a place for discussion. This kept discussion focused on specific tickets and issues, somewhat curtailing the tendency on wp-hackers for discussion to meander into rabbit holes.

As work on WordPress 2.1 went on, Ryan posted to wp-hackers about [getting WordPress 2.0 into Debian stable](#). Debian is a free and open source operating system distribution. The stable version is the latest official version of the software. WordPress was already in the testing and unstable distributions which both contain packages lined up for inclusion in the stable version. To be included in the stable distribution, WordPress 2.0 would need to be maintained for three to five years. This involved backporting security issues to ensure that all versions remained stable. Mark Jaquith took on the task of maintaining the WordPress 2.0 branch. The project committed to maintaining the version for three years, until 2010.

Shortly after he began leading the legacy branch, [Mark Jaquith received commit access](#). Mark had been involved with WordPress since 2004, coming to the project from Movable Type. In his two-year involvement with the project, he'd made many contributions, including to the successful bug gardening scheme. It was a sign that the funnel was starting to broaden, and the beginning of a period when -- slowly but surely -- more committers were added.

Commit access was offered sparingly; it wasn't just given out based on coding expertise or how much work a developer had done. Adding a new committer meant trusting them to uphold the project's user-first ethos. Then, and now, that's a key factor in deciding who is given commit access. "It's not just that you've shown a commitment to contributing," says Peter Westwood ([westi](#)) now, "but you also have shown an understanding of the ethos of the community and their shared beliefs, and their

philosophies."

Despite adding another committer, the development cycle wasn't getting faster, and the team decided to look more seriously at the development process. Since WordPress' first release, new versions arrived haphazardly, and as the number of users and developers grew, the outside impression was that things were somewhat chaotic. There was no real structure or development process.

A discussion on [moving to a 120 Day release cycle](#) started on wp-hackers. Matt shared his thoughts on what a release should look like:

2 months of crazy fun wild development where anything goes. 1 month of polishing things a little bit, and performance.

Feature freeze.

1 month of testing, with a public beta release at the beginning.

The community's response was positive. This provided a concrete deadline for them to work toward. Having shorter release cycles also meant that if a feature held up a release it could be pushed to the next one. In theory, this should prevent additional delays -- in practice this often depended on what the feature was, who had spearheaded it and how far along it was.

When WordPress 2.1 shipped, it was the first to launch with a concrete release date for the next version: [April 23rd for WordPress 2.2](#). The release date's publication meant that the project now had a firm deadline. Anyone who followed WordPress' development would expect the next version on that date. A deadline, among some in the project, came to be seen as a promise to users -- a release date promise that users could plan around. The project didn't quite make the next deadline but were only a few weeks off; [WordPress 2.2 released on May 16th, 2007](#).

23. Trademarks

In [March 2006 Automatic filed to register the WordPress trademark](#). Trademark registrations can be slow processes; the registrant has to demonstrate that they're pro-actively protecting that trademark. Toni Schneider spent much of his early years at Automattic doing just that. [He recalls now that he was trying to figure out](#) "how can we make this a very open brand that lots of people can participate in but it doesn't just turn into some mess that doesn't mean anything in the marketplace because everybody just uses it in different ways." The large ecosystem around WordPress can make it difficult for users to know who they're dealing with. Are they dealing with the official project or someone else? If the WordPress trademark became diluted it would be hard to identify official project sites from non-official sites.

Just before filing registration, a domain policy was added to WordPress.org. The policy requested that [community members not use WordPress in their domains](#). This was the first meaningful attempt to protect the trademark. To successfully register a trademark, the owner has to demonstrate that they enforce the trademark consistently. This means setting a consistent standard followed by everyone. For WordPress this meant a domain policy -- a ban on using WordPress in top level domains.

The problem with being permissive with a trademark is that the word or name can, over time, become generic: the meaning of the word changes from being an identifier for a specific product or brand and becomes a generic word. A familiar example of genericization is the word "aspirin," which has become generic in the United States but is still a registered trademark of Bayer in 80 other countries. The onus is on the trademark owner to protect it and prevent this from happening. Photocopying company Xerox, for example, ran an advertising campaign in an attempt to stop people from using "xerox" as a verb.

Once the trademark application was submitted, Automattic had to go about protecting the WordPress trademark. This meant getting people who used WordPress in their domain name or product name to change it. WordPress had been a community project for two years and many people had a stake in it. Many websites used WordPress in their domain names -- some of these were community websites, others were businesses selling WordPress products and services. There was confusion about what

people could and couldn't do. WordPress could not be used in a top level domain, but could community members use WordPress in the name of their blog or website? Was it okay in subdomains? In a comment thread on [Lorelle van Fossen's blog](#), community members wonder whether they can use the label "on WordPress" (as in "Lorelle on WordPress") on their blogs. After lengthy discussion in the comments, Lorelle updated her post to add:

After many discussions with Matt and the WordPress Community and staff, it is official. It is a violation of trademark to use WordPress in your site's domain name. You may use it in the title and in blog posts, however, please note that using WP or WordPress in the site title implies the site specializes in WordPress content. Don't disappoint.

The types of websites affected by this new policy were varied. Everything from community sites to businesses. Websites set up for the community in different locales were particularly affected. WordPress user groups set up their own websites -- [wordpress.dk](#), [wordpress.fr](#), [wordpress.pt](#), for example -- which served as community gathering points. They often had translated documentation, links to download localized versions of the software, and support forums where they offered support to people in their own language. There was a general sense among many in the project that the WordPress trademark -- the name and the logo -- did not belong to any individual but to the project itself. This meant that anyone should be able to use it. When the domain policy came into force, people in these communities felt frustrated that it had come as a top-down order, without any real discussion with the community.

The site [wordpress-arena.com](#) which was [running a WordPress theme contest](#), was another site affected by trademark enforcement. After Alex King's CSS style competition back in 2005, theme competitions continued to be popular in the community. They increased the pool of themes, allowed theme developers to show off their skills, and pushed the boundaries of theme development. In the comments, Matt left a message of support, along with a note that said "Unfortunately for every cool usage (like this competition, potentially) there are a dozen scammers and spammers misusing WordPress, selling spamming scripts or copies of WP itself."

Six months later, cease and desist letters were sent to people who continued to use WordPress in their domain. Attention centered on two commercial websites that had been running for some time with WordPress in their domain name: [wordpressvideos.com](#), run by Brandon Hong, and [wordpresstutorials.com](#), run by Sherman Hu. Both sites were registered prior to WordPress' trademark policy. These types of domains are more of a challenge to deal with than sites that straightforwardly sell plugins and themes. The main test for whether a website is in violation of a trademark is "likelihood of confusion." International community sites could easily be confused for the main project, as could sites selling plugins, themes, and other CMSs. Tutorials and books are not one of the main products provided by WordPress so it is more of a gray area. There are similar sites in the Adobe community ([photoshoptutorials.ws](#) and [photoshopessentials.com](#), for example) which are not owned by Adobe but which are not seen as trademark violations.

Sherman Hu and Brandon Hong received cease and desist letters and, as is often the case in the blogging world, a [blogger wrote about it](#). In his post, Andy Beard remarks on Automattic's tardiness in actually registering the trademark and preventing others from using it. He also takes issue with the two sites being singled out: one of them had been online for a year and both provided WordPress services and had cultivated their own communities, they weren't, he felt, the "spammy products" that Matt thought they were. Besides, just a year earlier, there had been a link farm on [WordPress.org](#).

People jumped to defend the two sites. Even those who supported Automattic's trademark position got angry when Matt branded the two sites as "snake oil." Customers and friends of Sherman Hu flocked to the blog post to voice their support for Sherman. Outside of that blog, prominent internet marketers wrote about it on their own blogs. They took issue with the branding of long-copy sales letters as spammy and scammy. [Brian Clark, at copyblogger, pointed out that while a website's design](#) might not appeal to some people, it will appeal to others.

Eventually the sites came to an agreement with Automattic. Sherman agreed to add a trademark symbol beside every use of the word WordPress and eventually [WordPress Tutorials came out with a redesign](#). Sherman shut the site down in 2008 after moving on to focus on his consulting business.

Since then the WordPress trademark has been consistently enforced. The advantage of Automattic enforcing the trademark was that, as a commercial entity, it was able to put resources into enforcement -- resources that the free software project just didn't have. The disadvantage was that it looked very much like Automattic was doing it for its own commercial benefit -- not the

community's. It wouldn't be for a number of years that the trademark situation was properly resolved.

24. Habari

By three years in, the project included developers, support forum volunteers, documentarians, and other people interested in helping out. Many had been around since the early days, and while the project and software had grown and changed, some felt that the governance structure had not. A free software project can be run in many different ways. WordPress has always had a Benevolent Dictator for Life (BDFL) structure, meaning that the final decision often rests with the BDFL -- Matt. In practice, it was often Ryan Boren who made and implemented most of the technical decisions. However, some contributors felt that the project could benefit from a committee structure, with a team of people driving the project's direction and decision making. There was a difference of opinion in how a free software project should run. Some were unhappy with how Matt lead the project, and [left](#). It was in this context that the first major fork of WordPress happened -- though it wasn't a fork of the software, but a fork of the community.

The unhappy contributors happened to come together in September 2006, at the Ohio Linux Fest in Columbus. They had lunch together at a Buca di Beppo. Among the diners were four WordPress developers -- Owen Winkler ([ringmaster](#)), Rich Bowen ([drbacchus](#)), Scott Merrill ([Skippy](#)), and Chris Davis ([chrisjdavis](#)). During lunch they talked about their project grievances. The same story came up again and again -- patches were rejected because they didn't fit the project's vision. Conversation kept returning to the possibility of setting up a new free and open source (FOSS) project. By the time the meal ended, the four WordPress developers had decided [to stop talking and act](#); they would create a new blogging tool. A few weeks later at ApacheCon, they chose a name: Habari, which is a Swahili word for "What's up?" or "What's the news?" At the same gathering, they created a governance structure and development ethos for the project, and started work in earnest.

Habari launched in January 2007, and many of Habari's founding philosophies are in direct opposition to those of the WordPress project. The [Habari Motivations](#) page lists the reasons for being involved with Habari. Each of the motivations addresses something that the founding developers believed was wrong within the WordPress project -- things that they wanted to do differently. They reflect the different approaches to running a free software project, and also highlight some problems afoot in the WordPress project three years after its launch. Every free software project -- indeed every group or community -- has its problems, especially an established community. Differences that may have been ignored during the initial heat of enthusiasm become entrenched, and as enthusiasm fades, those differences surface.

Governance style was one major issue. Many weren't happy with the BDFL model. A BDFL governance model is common in free software projects. Notable BDFLs are Linus Torvalds of Linux and Rasmus Lerdorf of PHP. A BDFL is a free software project leader -- the one person who has final say about the project's direction and vision. In reality, as a project grows there are often many influential people who make decisions about project direction.

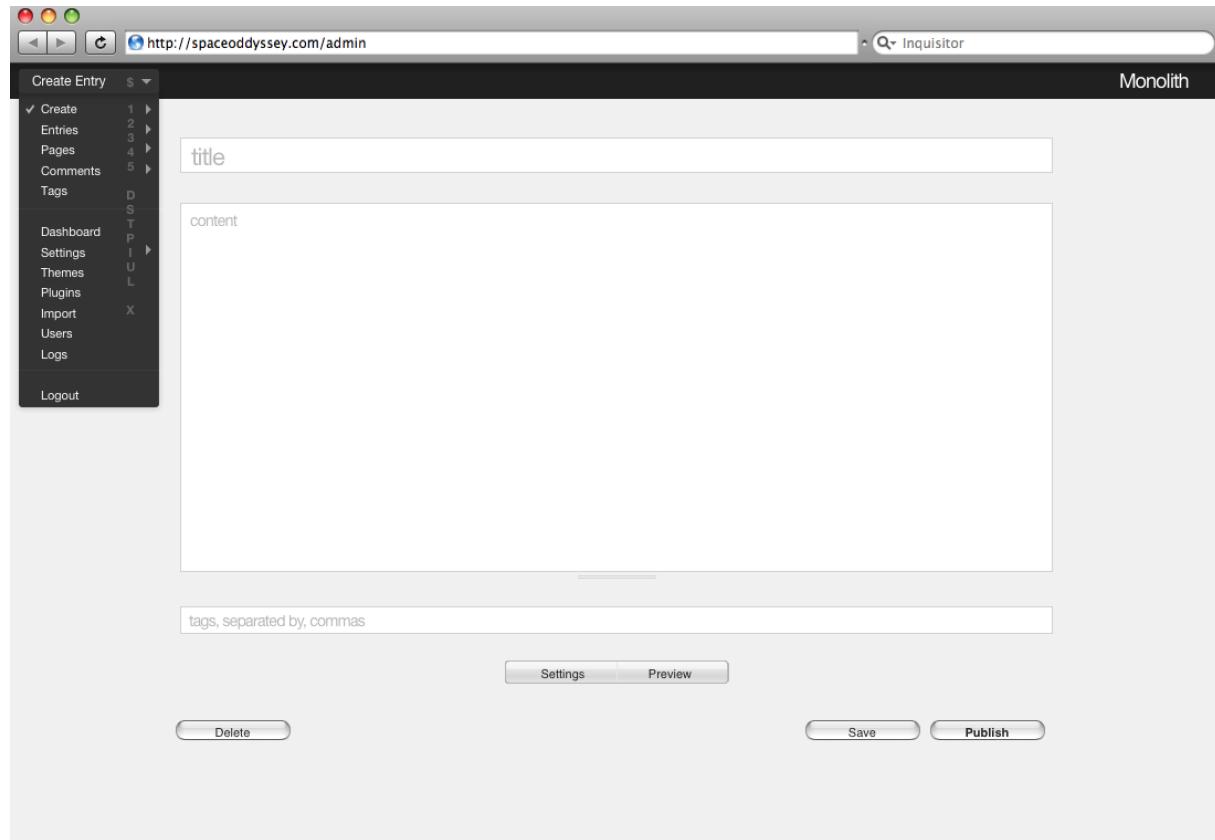
Rich Bowen, one of the developers who was dissatisfied with how WordPress was run, came from the Apache project. Apache has a committee model. The committee comprises developers and non-developers, all of whom have a say in how the project runs. Unlike WordPress, which requires that commits be approved by the committer and/or the patch reviewer, commits in Apache require consensus. Habari launched with a committee structure, much like Apache. Decisions are made by committee consensus.

[Designer Michael Heilemann](#) was one person enticed by Habari. He'd designed the Kubrick theme and was looking for a new challenge after the Shuttle project failed before implementation.

Michael redesigned the Habari interface -- which he says now was both a good and a bad experience. [He had a good experience designing and implementing it](#). He had a bad experience getting it approved, because of Habari's committee structure. A lot of time was spent discussing and approving the new admin interface. People with no design background

weighed in with opinions and everyone received equal weight. This left Michael feeling like he couldn't get his work done and he ended up leaving Habari. When asked about whether he prefers a BDFL or Apache-style model, [he says now](#):

It's easy to end up in very long discussions if everybody has equal footing. And that makes for a great democracy, but it's also very hippie, 60s, everybody gets to sit around and share their opinion, but that's not always something that's really worthwhile. You don't actually, necessarily, get a better product out of it. And so often you need somebody with vision, or at least somebody with a point of view with opinion to weigh in.



The Habari Monolith interface, designed by Michael Heilemann

Commit access was another flashpoint. In the Apache project, a clear path to commit access to the repository existed. In WordPress in early 2006, getting commit access was not possible for most people. Only Matt and Ryan had commit access to WordPress; they acted as a funnel through which all code would be reviewed. More committers were eventually added, but it was a slow process and one that led to frustration, particularly among prolific contributors.

Coming from Apache, Rich brought a different perspective to the project -- one that chimed with the more dissatisfied developers. For Rich, WordPress didn't constitute a true meritocracy -- there was no opportunity for those with ability to gain power and responsibility. The final decision over who did and didn't get power and authority lay with the project's leader -- it was up to Matt's discretion to determine who had ability and deserved power. When Matt, in a discussion thread, said that WordPress is a meritocracy because commit access is provided to "the best of the best of the best contributors who prove their worth over time with high quality code, responsiveness to the community, and reliability in crunch times," Rich responded by pointing out that because the final decision lies with the BDFL it prevents the community from becoming a true meritocracy.

What lay at the heart of the discussion was a fundamental disagreement about how a free software project should be structured. In *Homesteading the Noosphere* Eric Raymond discusses [project structures and ownership](#), looking at how a project structure emerges over time. The emergence of WordPress' project structure very much follows what Raymond outlines; the project has a single owner/maintainer, or in the case of WordPress it had two -- Mike Little and Matt Mullenweg. Over time, Matt assumed project leadership. In WordPress' case, commit access carried with it an implicit level of authority. Those with commit access are the arbiters of the code that ends up in core. Matt [made clear on a number of occasions](#) that he didn't "want it to be a status thing," but this didn't prevent it from becoming so.

Matt posted to the [mailing list in 2005](#) saying "Committing != community status. It simply is a way to ensure that all the code that goes into the core is reviewed before being distributed to the world." But community members naturally ascribe authority to those with commit access. People with commit access are trusted above others. It was unclear what it took to become a committer or a lead developer. Instead, the roles developed organically. For the community, authority naturally followed commit access to the repository, and commit access followed high quality code submissions along with an understanding and commitment to the project ethos. Commit is a symbol of trust; with only two committers it appeared that the project leads did not trust others in the community.

This had another consequence: a lack of acknowledgement for non-code contributions. Because implied authority naturally followed code it meant that there were no defined leadership roles within the project for non-coders. Commit access may have been given out sparingly to developers but authority and leadership was still in theory achievable to anyone who could write code. For those whose role was in the support forums, writing documentation, providing translations, or any of the myriad supporting activities that go on around software development, there was no formal way for those who worked hard and demonstrated a commitment to the project to progress, gain status, or be acknowledged. While coders receive props for code accepted into core, there was no system of acknowledgement for those who worked on supporting activities. So when there were rumblings of discontent, it wasn't just coders who were complaining.

Habari's approach was the opposite of WordPress. Commit access was achievable by anyone. The [Habari motivations page says](#) "Our contribution model is a meritocracy. If you contribute code regularly, you will be granted permission to make contributions directly (commit access)." The project takes this approach even further -- it isn't just coders who received commit access. Owen Winkler, one of Habari's founders, [says](#):

There are some people who are committers, who are part of the primary management committee in Habari who I would never want to actually touch the code because they don't really do development. But we give them access to it because they've demonstrated that they're part of the community and they're actively trying to advance Habari as a project.

In Habari, commit access is a clear sign of trust and responsibility in the project. In a project in which commit access equals responsibility and authority, it makes sense to give access to anyone who holds such a position within the project. However, if you're trying to create a project in which the two do not go hand-in-hand, keeping commit access as purely a checking mechanism makes sense. That said, if it's not clear to people within the project what constitutes authority and what doesn't, then commit access is one of the few indicators of authority.

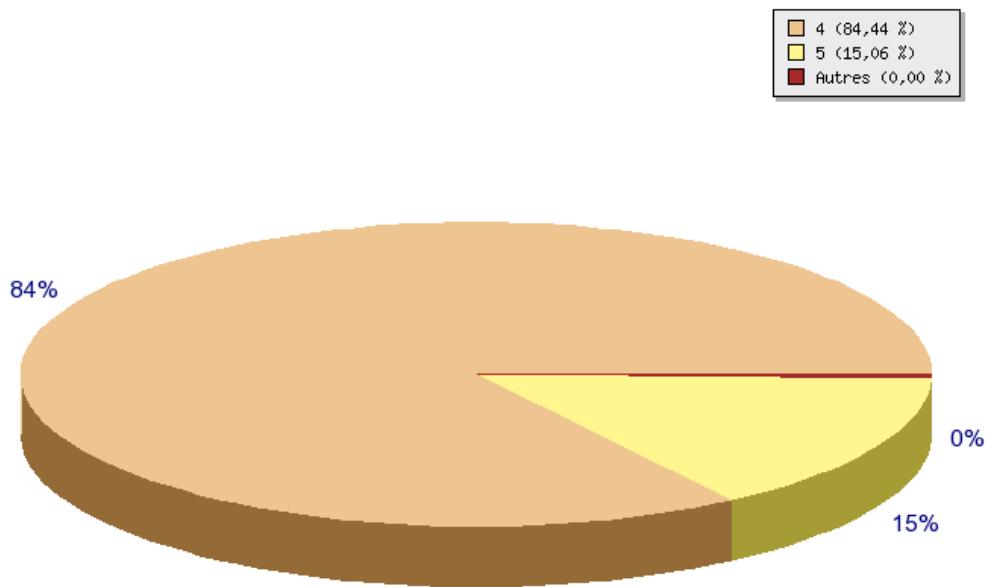
WordPress' focus has always been on users, and maintaining a good experience for them. Habari, on the other hand, was intended to be useful to users, but at its heart it is an educational tool for developers. One of the driving ideas behind Habari was to create a tool that taught people about development, development best practices, and being part of a free software project. Rich Bowen, who didn't know a lot of PHP, said that he hoped that the new project would teach him about it. This meant not just teaching developers about PHP, but educating them about best practices around coding methods and being part of a FOSS community. They eschewed WordPress' low barrier to entry in favor of an approach that would cultivate fewer but stronger developers.

Habari was written in PHP 5.1, giving developers access to [PHP Data Objects](#). This gave Habari a level of database independence lacking in WordPress. On one hand, Habari developers found fault with WordPress for not using the latest coding practices, on the other, WordPress developers felt that the Habari approach put users second. As [Ryan Boren put it](#):

Screwing users because developers want to play is not cool. It's a good way to become irrelevant. A big reason WP is so popular is because it is not infatuated with the latest, greatest developer lust objects that require users to upgrade their platform.

In February 2007, when Ryan wrote that comment, PHP 4 was still running on 88.44% of websites. PHP 5 adoption was slow and many web hosts didn't support it. If WordPress had made the switch to PHP 5 it would have meant that it was suddenly unavailable on a huge number of hosts -- breaking users' sites around the world.

Major version repartition



Nexen Services (<http://www.nexen.net/>) © 2005-2007

PHP major distribution statistics in February 2007

Habari's license choice also signifies fundamental differences in FOSS beliefs. WordPress has a GPL license, which is intended to secure the freedoms of users. Habari, on the other hand, uses the [Apache License](#). This permissive license allows developers to take the code and use it in any way -- even in a proprietary product. In the Apache license, the emphasis is very much on developer freedom. [Owen Winkler outlined the reasons](#) why the Habari project chose not to use a GPL license:

There's the idea of developer freedom. If you give the software away you should be able to give it away with no requirements at all. If you want to take it and make software that you sell, then go ahead and do that. Hopefully you won't. Hopefully you'll contribute. Or if do, you'll contribute what you make back to the community somehow.

All of these differences were, in the end, unresolvable, despite the fact that things were starting to change. Change was slow and the developers became more and more disenfranchised, but change was happening. Mark Jaquith, who was given commit access while the other developers were quietly working on Habari, [recalls](#) "I felt like [...] Matt was right on the verge of loosening the reins, it felt like things were really starting to get good and they left. And I was like oh, if you'd just held on."

But they didn't hold on. By the time the reins loosened they had founded their own project with their own ideals and focus. FOSS communities can be run in many different ways. Sometimes the emphasis is on free software, sometimes open source. Sometimes the emphasis is on users, sometimes developers. Sometimes the license is permissive, sometimes copyleft. The wider free software communities are broad and encompass people from different backgrounds. Each project has its own set of ideals and cultural norms, and sometimes project members gel with those ideals and sometimes they don't. When there is a critical mass of people within that project who don't accept a project's core ideals then there is always the possibility of creating a fork. This can be a fork of the software or, as was the case with Habari, a fork of the community. This is not, necessarily, a [Bad Thing](#). Rather than struggling with constant debates and infighting that comes from a clash of beliefs, sometimes it is better for two groups to separate. Then there are two projects, each with a laser-sharp focus on the things that matter to them and on the software and project that they want to create.



Chris Davis, Owen Winkler, and Nicole Evans at Ohio Linux Fest 2007 (Image CC license Sean T Evans)

Endmatter

February 2005 - January 2007

Releases: WordPress 2.0, 2.1

Jazzers Duke Ellington Ronan Boren Ella Fitzgerald

Features

WordPress 2.0 Shuttle-inspired design changes WYSIWYG editing User roles Header customization Pluggable object cache Theme functions (functions.php) Akismet bundled with core

WordPress 2.1

Autosave Tabbed editor Spell checking Search engine privacy Static front page Redesigned login (from Shuttle) Draft and private pages Upload manager

Community

- WordPress logo
- Shuttle project
- first WordCamp San Francisco
- WordPress trademarks registered
- 2.0 legacy branch
- Habari launched

Companies launched

- Automattic

PART 4

25. Creating a Folksonomy

In the wake of the exodus to Habari, the project began to evolve. In March 2007, Robin Adrianse ([rob1n](#)), WordPress' first temporary committer, received commit access for three months to help Ryan Boren address languishing trac tickets.

Also in March, the plugin directory launched. Before the plugin directory, many developers just hosted their plugins on their website. The plugin directory gave them exposure to a huge number of WordPress users. Samuel Wood ([Otto42](#)), [recalls how the plugin directory encouraged him to distribute his code](#). “I was writing them before, but I didn't give them to anybody. It encouraged me to release plugins because I had a place to put them.”

[* Extend Home](#)[* Themes](#)[* Plugins](#)[* About](#)[* Tags](#)[* Add Your Plugin](#)[* Ideas](#)[* Kvetch!](#)MOST POPULAR TAGS [MORE »](#)[* post \(15\)](#)[* comments \(10\)](#)[* widget \(9\)](#)[* ajax \(8\)](#)[* admin \(7\)](#)[* images \(7\)](#)[* category \(6\)](#)[* Technorati \(6\)](#)[* stats \(5\)](#)[* spam \(5\)](#)[* google \(5\)](#)[* del.icio.us \(5\)](#)[* statistics \(4\)](#)

SEARCH PLUGINS

[Go »](#)

Plugins

Featured Plugin

Subscribe to Comments

Subscribe to Comments is a robust plugin that enables commenters to sign up for e-mail notification of subsequent entries. The plugin includes a full-featured subscription manager that your commenters can use to unsubscribe to certain posts, block all notifications, or even change their notification e-mail address!

[More Info](#)[Download](#)

Most Popular

Akismet

Downloaded 134,024 times

Sidebar Widgets

Downloaded 97,371 times

Subscribe to Comments

Downloaded 12,225 times

Page Links To

Downloaded 2,928 times

Kramer

Downloaded 2,091 times

No WWW

Downloaded 1,662 times

Newest Plugins

Associative Dictionary

Added March 19

Full Text Feed

Added March 19

Run for Cover

Added March 19

Spam Viewer

Added March 18

FeedStats

Added March 18

©Feed

Added March 18

Recently Updated

Upload+

Version 0.1d

Guestbook Generator

Version 0.8

podPress

Version 7.6

WP-Print

Version 2.10

Running Time

Version 1.1 b2

JunxterAds

Version 0.1

Get Hosted

Want to see your WordPress plugin here? Go [add it!](#) All we require is that it be **GPL Compatible**.

Not convinced? We've got [more info](#) that should persuade you.

+∞+

ANNOUNCEMENT LIST: [Join](#)[REPORT A SITE BUG](#)

CODE IS POETRY

The WordPress Plugin Directory in 2007

WordPress 2.1 launched in January 2007, after a release cycle of more than a year. The following cycle, WordPress 2.2, was the first to adopt the new 120-day release cycle. This new goal, which would later become “deadlines are not arbitrary” in

WordPress' [list of philosophies](#), was an ongoing challenge, tested even in WordPress 2.2, which featured a new taxonomy system -- the biggest database architecture changes to date. Developing in an open source environment means leaving time for every voice to be heard, waiting for volunteers with busy lives to get things done, and discussing new features and architecture changes. It's a challenge that WordPress would have to address release cycle after release cycle.

In the early 2000s, the internet abounded with discussions about the best way to organize information. Content has metadata assigned to it, which can be used to organize and display information. Traditional web classification methods imposed a top-down categorization structure. A website created a category structure and users placed user-generated content in the correct category. These structures were often rigid, forcing users to shoehorn their content into something that didn't necessarily fit. A new method of classification emerged -- tags, a bottom-up classification form in which an index or a cloud can be generated based on keywords that the content creator applies to the content.

Social bookmarking site [Del.icio.us](#) was the first system to use tags. While Del.icio.us wasn't a bookmarks management pioneer, its tagging system set it apart. Users tag saved links, which are used to relate links to each other, both in a user's own collection and across the entire social network. Visiting the link <http://delicious.com/tag/php> displays all of the links tagged PHP.

Controlled, top-down content classification methods failed to scale on large-scale Web 2.0 applications in which millions of users created and organized their content. Users couldn't be forced to use controlled vocabularies. [Clay Shirky discussed the problem:](#)

users pollute controlled vocabularies, either because they misapply the words, or stretch them to uses the designers never imagined, or because the designers say "Oh, let's throw in an 'Other' category, as a fail-safe" which then balloons so far out of control that most of what gets filed gets filed in the junk drawer. Usenet blew up in exactly this fashion, where the 7 top-level controlled categories were extended to include an 8th, the 'alt.' hierarchy, which exploded and came to dwarf the entire, sanctioned corpus of groups.

The classification system that used mass tagging systems became known as a [folksonomy](#). This is a classification system in which users classify content themselves.

In 2005, Technorati, the blog search engine, [launched its own tagging system](#). This created a tag search across major platforms such as Blogger and Typepad, CMSs like Drupal, and other services such as Flickr, Del.icio.us, and Socialtext.

WordPress lagged behind, and there was pressure from both the free software and WordPress.com communities to add tags to the platform. WordPress' native classification form is hierarchical, top-down categories. To interact with WordPress, [Technorati picked up the "tag" from the WordPress post's category](#). This was unsatisfactory to many users as categories and tags are two different types of classification.

On his blog, [Carthik Sharma wrote](#):

categories can be tags but tags cannot be categories. Categories are like the huge signs you see on aisles in supermarkets – “Food”, “Hygiene”, “Frozen” etc, they guide you to sections where you can find what you are looking for. Tags are like the labels on the products themselves.

Categories and tags have two distinct use-cases. Categories are more rigid, whereas tags are a lightweight way of classifying content. There was, of course, a plugin that did the job: WordPress users installed the [Ultimate Tag Warrior](#) plugin (though WordPress.com users could not).

In 2007, [Ryan Boren opened a ticket to add tagging support to WordPress](#). Finally, WordPress would have tags. The next step was to identify the database schema.

A database stores all of a WordPress user's content and data. The database contains different tables from which data is retrieved. There is a post table, for example, which stores post-related data. The user table stores user data. Making changes to the database is not trivial. Any changes have to be done correctly because undoing them in the future can be difficult. Changes need to be performant. In a PHP/MySQL setup, increasing the number of database queries can slow the site down. The question arose: where should we store tagging data? Should it be in a new table? Or should it be stored in an existing table?

Ryan proposed two database schemas. One of these [created a new table for tags](#). Matt was keen on the second proposal -- putting [tags in the categories database table](#). He believed it didn't make sense to create another table identical to the categories table. In a comment on Trac he wrote:

We already have a ton of rewrite, API, etc infrastructure around categories. Mostly I see tagging as a new interface to the data. On the display side, people want their tags listed separately from their categories, and probably something like a tag cloud function.

WordPress had already successfully reused tables in the past; for example, posts, pages, and attachments are all stored in the same table, and at that time the categories table also contained link categories. Tagging, from a user's perspective, enabled them to tag posts, display a list of tags, and display posts with the same tag. What was the point in duplicating the infrastructure when it could be achieved within the current table system?

[Few developers supported](#) putting tags in the categories table, because categories and tags are two different ways to organize data. Some believed the category table would become bloated and argued that putting tags in the category table meant including additional code to keep the two taxonomy types separate. This additional code could introduce bugs, make it harder to maintain in the future, and make it difficult for developers to extend.

On [wp-hackers](#), April 2007 was spent discussing the new database schema for taxonomies. A suggestion that got considerable community traction was splitting categories, link categories, and tags into their own individual tables -- but no one could come to a consensus.

After checking in the single-table taxonomy structure (changesets [#5110](#) and [#5111](#)), Matt posted a new thread on wp-hackers [stating the case for using the categories table](#). He argued that:

1. It would perform faster as no additional queries would need to be carried out to support tags. A separate tag table would require at least two extra queries on the front end.
2. That it would provide a better long-term foundation. Tags and categories would be able to share terms (for example the category "dogs" and the tag "dogs").
3. That there were no user-facing or plugin-facing problems.

In the thread, he also proposed an alternative; creating a new table, inspired by Drupal's taxonomy system, specifically for terms within a specific taxonomy. Terms are the items within a category or tag, or other taxonomy; dog, cat, and chicken are all terms within an "animal" taxonomy. This additional table allowed terms to be shared among taxonomies while having the same ID (the ID is what identifies an item in the database). Just one term, "dog" for example, would be saved in the database, and this term could be used in any taxonomy.

Ryan Boren [proposed a compromise](#), one which enabled individual terms to be part of any taxonomy while keeping the same ID. It was a three-table solution, with a table for terms, taxonomies, and objects. Discussion ensued, a [new Trac ticket was opened up](#), and a new structure was created based on this proposal. The first table `wp_terms`, holds basic information about single terms. The `wp_term_taxonomy` places the term in a taxonomy. The final table, `term_relationships` relates objects (such as posts or links) to a `term_taxonomy_id` from the `term_taxonomy` table.



This approach has the advantage of assigning one ID to a term name while using another table to relate it to a specific taxonomy. It's extensible for plugin developers who can create their own taxonomies. It also enables large multisite networks, such as WordPress.com, to create global taxonomies -- unified tagging systems in which users of different blogs can share terms within a taxonomy.

As with many WordPress features, tags landed on WordPress.com before they shipped in WordPress. From the beginning, the new structure caused huge technical problems. The increase in tables meant that WordPress.com needed more servers to deal with the additional queries. "It was slower to be completely honest," [says Matt](#), "that was a cost that we saw in a very real way on WordPress.com, but also a hidden cost that we did impose on everyone who was doing WordPress in the world."

As well as being a challenging code problem, the taxonomy implementation activity highlighted development process problems. Heavy discussion meant development dragged on. Some wanted to delay the release, others wanted to pull the feature, others wanted to revise the scheme in a subsequent version. Andy Skelton [responded to the wp-hackers discussion](#):

To include a premature feature in an on-time release degrades the quality of the product. I refer to not only the code but the state of the community. Increments are supposed to be in the direction of better, not merely more. Better would be to release on time with a modest set of stable upgrades.

To block release while the one new feature gets sorted out would be a maladjustment of priorities. If 2.2 seems light on sex appeal, so be it. Better to keep the release date as promised.

The 120-day release schedule was in danger from just one issue. Making major architectural changes to core just weeks before a release was contrary to the new development process' aims. A thread suggested [delaying 2.2's release](#). The architectural changes were too important to be done in an unsatisfactory way.

Eventually, they decided to [delay the 2.2 release](#), pull tags out of core, and bring in widgets as a user-facing feature to encourage users to upgrade.

[Andy Skelton developed widgets](#) for WordPress.com users, built as a way to give people more flexibility with their blog's layout. Widgets are code blocks users can drag and drop into place through the user interface. Users can add a calendar, a search bar, or a piece of text, for example, to their sidebar and put them in any order they wish. It was hugely popular among WordPress.com users and the plugin for WordPress users proved popular too. Launching a feature like this on WordPress.com first meant it could be tested with lots of different users before making it into core as a feature.

The new tagging feature finally shipped in WordPress 2.3 -- a structure which was much compromised and haggled over. The wrangling over the taxonomy structure has had consequences for WordPress developers ever since. Shared terms have had lasting implications for core developers, plugin developers, and users. The problem with shared terms is that an item could be conceptually different but it is treated the same in the database. For example, the word apple. A developer creates the taxonomy "Companies" and the taxonomy "Fruit" and the user places the term "apple" in both. Conceptually, this is a different -- a company and a fruit -- and they appear in the user interface as different concepts, but the database treats them as the same thing. So if a user makes changes to one, like capitalizing it, for example, changes are made to both.

In a post in 2013, [Andrew Nacin wrote that](#) "Hindsight is 20/20, and shared terms are the bane of taxonomies in WordPress." Each term is represented in two different ways. Since an individual term can appear in multiple taxonomies, it's not straightforward to identify the actual term in the actual taxonomy that you want. Every term has two ways to identify it. This has caused challenges for building new features that require using one identification method when there are parts of WordPress that use the other.

To attach metadata to a term, for example, there must be one object identifier, but the public ID uses a different identifier. Consistently targeting data which is identified in multiple ways is extremely difficult. WordPress has a long-term commitment to maintaining backward compatibility so just simply creating a new schema isn't possible. The effort to get rid of shared terms must be done step by step, over a number of major releases in which shared terms stop being created, get split, slugs and names get moved from `wp_terms` to `wp_term_taxonomy`, and the `wp_terms` table gets dropped (this is being carried out over three releases in the 4.x series).

The over-engineered taxonomy system came via arguments and compromise. When the bazaar model works, it can produce software that people love to use. The bazaar model fails in intractable arguments that result in a compromise that no one is 100% happy with. It did, however, have one positive benefit. Neither of the original proposals for the taxonomy schema (putting tags into the category table, or creating a new table for just tags) would have allowed developers to create custom taxonomies, which was part of WordPress' transition from a straightforward blogging platform to a content management system (CMS).

Even with these arguments, software development continued mostly on schedule. Since tags were pulled from 2.2, there were only 114 days between WordPress 2.1 and 2.2, and then 129 days between the release of 2.2 and 2.3. While delays happened in the future, there was nothing like the long dark year between 2.0 and 2.1.

26. Sponsored Themes

For a long time, a community of hobbyists drove WordPress, but eventually the hobbyists wanted to support their hobby. Developers charged for customizations and Automattic was building a blog network and related products such as Akismet. Theme designers, too, wanted to make money alongside a free software project for the time and effort they put into developing themes.

It isn't immediately obvious how someone can make money and uphold a free software project's ethos. How can the two goals co-exist? In the early days of the WordPress economy the distinction between freedom and free beer was blurry. Community members fumbled for answers: is it acceptable to make money with a GPL project? Who has the right to make money out of a project that belongs to its volunteers? How can one run a theme business when the core product is free?

The period between 2006 and 2009 was one of experimentation and discovery for businesses in the WordPress ecosystem. During these tumultuous years, the community wrestled with commercialization and the theme marketplace grew. WordPress users -- web users -- have always been concerned with their websites' look. As WordPress became more popular, many bloggers built their own themes and realized that they could make money from them.

Themes lagged behind plugins for an official repository. For a long time, themes were hosted on themes.wordpress.net, an unofficial theme directory. Theme developers could simply upload their theme and users could browse the directory.



NOVEMBER 30, 2006

USER RIGHTS UPDATE

SORTING OPTIONS

- 1 column
- 2 columns
- 3 columns
- 4 columns
- Red
- Green
- Blue
- Violet
- Orange
- Brown
- Pink
- Salmon
- Gray
- Black
- White

Due to massive amounts of work I haven't been able to update user rights before now - so everyone earlier have requested rights to upload themes, you should be able to now.

68 RESPONSES



[Download \(26\)](#)

[Test run](#)

Rating: Not rated yet



[Download \(66\)](#)

[Test run](#)

Rating: 5



[Download \(65\)](#)

[Test run](#)

Rating: 5

The WordPress theme viewer in 2007

However, the system was susceptible to spam and duplicate themes. Some theme developers abused the system by [downloading their theme multiple times](#), so that they would appear at the top of the most popular results. The theme viewer was at the centre of the first big debate about themes.

Theme designers tried theme sponsorship as a way to make money from their designs. Designers often link to their website in their theme as a credit. Kubrick, for example, links to Michael Heilemann's website. Every site that installs the theme links to the designer's website. The number of incoming links to a website is a variable in Google's PageRank algorithm: the more incoming links, the higher the PageRank, the further up in Google's search engine results. If hundreds, or even thousands, of people install a theme, the designer can watch themselves soar up Google's search results. Even better if that link comes from a high authority website.

Designers realized the link doesn't have to be to their own website -- it can be to anything at all. Links have an intrinsic value, particularly to internet marketers. With a link in a WordPress theme, marketers don't have to approach individual sites to ask them for a link. All they have to do is pay a web designer to include it in their theme, release the theme for free, and soon hundreds of sites are linking back.

Theme sponsorship approaches vary. Sometimes companies contact well-known theme authors to sell links in their themes. Theme authors are also proactive. Authors [advertize and sell their themes on websites](#), others [auction the theme at Digital Point](#), or they simply auction links. In those early days, once the sale was made, designers would promote the theme on different WordPress theme sites, including official sites such as the WordPress Codex, community resources like the WordPress theme viewer, and reputable blogs such as Weblog Tools Collection. These themes were often distributed with a [Creative Commons 3.0 license](#), which permits free sharing and theme adaptation, provided the credit link remains.

Websites focused on making monkey online became aware of theme sponsorship. [Tutorials and articles about theme sponsorship proliferated](#), and sponsorship became part of an acceptable "link building strategy."

Theme authors took different approaches: some published themes with visible text links but didn't tell their users about the link, others used PHP or CSS to hide the links, others still made it very clear that the theme had been sponsored.

For those in favour of theme sponsorship, the matter was simply about being paid for their work. Why should they work for free? Why shouldn't they be compensated for their work? Selling sponsored links ensured they could create and distribute themes for free, which benefited the whole community. They argued that designing a good theme takes time and that free themes were inevitably of poorer quality than sponsored themes.

Sponsored themes quickly became prevalent, with even respected sources [selling links in their themes](#). It became a "great business model"; finally a way to make money from WordPress! Besides, those who supported the model argued that the default WordPress blogroll contained links to all of the original developers of WordPress -- Alex, Donncha, Dougal, Michel, Matt, Mike, and Ryan -- all of whom were benefiting in Google's search results.

Others felt that themes.wordpress.net was becoming a [spam repository](#). More than 50% of themes on the WordPress Theme Viewer were sponsored themes. These contained links to everything from iPhone repairs to gambling websites, to web hosting and flower delivery. Some themes were uploaded multiple times with only minor changes, thereby increasing the number of links on the theme viewer. Critics of theme sponsorship -- many of whom were designers themselves -- said that the themes polluted the community. They weren't against theme designers making money, but they didn't want to see the WordPress community become a hive of spam and SEO. Theme sponsorship had opened the floodgates to SEO and internet marketers.

Buying and selling links went beyond the WordPress community. A few years earlier, Matt Cutts, the head of web spam at Google, explained how [link selling affected PageRank](#). Google's algorithm detected paid links, and while it wasn't foolproof it worked pretty well. Paid links made it harder for Google to gauge a website's trustworthiness. Google, as a result, took away that site's ability to give reputation:

Reputable sites that sell links won't have their search engine rankings or PageRank penalized – a search for [daily cal] would still return [dailycal.org](#). However, link-selling sites can lose their ability to give reputation (e.g. PageRank and anchor text).

WordPress users who installed sponsored themes could be penalized for links that they weren't even aware they were hosting. And, hidden links could further reduce a user's PageRank. In a post in April 2007, Matt Cutts condemned [hiding links in a website](#) and asked web masters to disclose paid links.

There were plenty of themes that contained links that weren't sponsored. Designers [included a credit links in their footer](#), linking to their website. Designing and releasing a WordPress theme allowed designers to increase their profile on the web. Like the internet marketers, their websites benefited from higher search engine results, and website visitors might click on the link, sending business to the designer.

It was common at that time for a designer to release their themes under a Creative Commons license, which asks users to leave the credit link intact. In the middle of the sponsored link furor, one designer took the next step. Tung Do ([tungdo](#)) authored the popular WordPress resource WPDesigner.com, along with a number of WordPress themes. In April 2007, he announced he would [release his themes under the GPL](#). "Despite that I'm just ONE theme designer and despite that I don't contribute directly to the WordPress code (sic)," he wrote, "I believe that switching to GPL is a step in the direction to support the WordPress team and to help the WordPress theme community grow (positively)."

Weblog Tools Collection (WLTC) was the first website to take direct action against sponsored themes. At WLTC, theme authors submitted themes and Mark Ghosh, who ran the site, regularly wrote about [theme releases](#). In April, Mark weighed in on sponsored themes. While he didn't condemn them outright, he did instate a new policy:

all themes with sponsorship links will be labelled as such when they are published, non-sponsored themes will be published first and we require sponsorship disclosure to be made to us when authors make us aware of their new themes. If this disclosure is not provided and the theme has sponsored links, the author will be barred from being able to post their new themes on [weblogtoolscollection.com](#) until further notice.

The [166 comments in the comment thread](#) highlight just how divisive the issue was. Viewpoints ran to both extremes. Many users were unhappy about links being placed in their websites -- this was particularly concerning to people whose websites had a moral or religious bent. While users supported linking to theme authors, they weren't happy that links for credit cards or flower delivery were being displayed on their website. Theme developers said that it was a way for them to be able to fund themselves

creating free themes and were sad to see it being abused.

A few days later, Matt followed Mark's lead and [posted on WLTC about sponsored themes](#). He had [become aware of the trend of including advertising links in themes](#) back in September 2006 when he had downloaded the Barthelme theme from plaintext.org and discovered a link to a New York flower delivery service. For him, there were three main issues:

- that sponsored links negatively impact a user's Trustrank and that the user hadn't made this decision themselves;
- that sponsored themes are adware;
- that theme authors who sell links and release it under a creative commons license contravene the GPL.

All of these factors meant a negative experience for WordPress users. While the project supported people making money from WordPress-related products and services, it didn't support methods detrimental to users. Whatever the original intentions of sponsored links, themes had become so polluted that they undermined the trust that a user had in the software and the community. As a user-focused community, the project needed to regain that trust.

The argument that theme authors deserved to be compensated for their work held little weight when WordPress itself had been built by volunteers. There was no opposition from the project to people making money from WordPress, but official WordPress resources should only promote companies and individuals in line with the core project ethos.

Matt closed the post, [linking to a vote](#) on a proposal to remove sponsored themes from WordPress.org. The discussion on the thread has arguments for and against theme sponsorship; some vote for a complete ban, others for sponsored theme disclosure, while others feel theme designers should be allowed to include any links they want in their theme.

Whatever the results of the vote, the tide was turning on sponsored themes. There were not looked upon favorably at WordPress.org, with sites and people who had promoted sponsored themes already [banned from the forums](#). Even Matt Cutts weighed in, saying that he [agreed 100% with Matt's position on sponsored themes](#).

In July, Mark announced on WLTC that he [would no longer post about sponsored themes](#) and shortly after [Matt announced that all sponsored themes would be removed from themes.WordPress.net](#). Despite a positive reaction from much of the community, there was a backlash, directed particularly toward [Mark](#) and [Matt](#).

Some theme developers saw theme sponsorship as a valid way of making money, and [were angered at being branded as "unethical,"](#) particularly when they saw other theme developers behaving unethically; downvoting other developer's themes, using sock puppetry to upvote their own for example, or stuffing themes with copious links to their website.

Despite the sponsored theme ban on official WordPress resources, link sales continue today. The Digital Point forums, for example, are filled with themes available for sponsorship. Theme sponsorship is not without its dangers. In 2012, a former theme sponsor [posted on the Webmaster world forums](#) about Google penalization for "inorganic" incoming links:

Some 2+ years ago in throws (sic) of questionable wisdom I sponsored about 5 or 6 WordPress themes where the "Designed by" link in the footer gets replaced by a link to your site. They were nice looking and "relevant" themes, at least as far as the name and pictures used in design suggest. They were not used much initially and I did not think much of them until these "unnatural links" notices started flying a month ago. Google confirmed that these links were the issue, but with the themes in the wild there wasn't a whole lot that the sponsor could do about it other than contact the websites using the theme and asking them to remove the link.

Sponsored themes was the first approach that people, en masse, took to make money from WordPress themes. There were people doing custom design and development, but link sales appeared to be a valid way of making money, particularly on an internet that abounded with SEO experts and internet marketers on an unremitting search for ways to climb Google's search results. Sponsored themes brought WordPress, not for the first time, into proximity with SEO, both white hat and black hat. Selling links in a theme slides easily into questionable SEO practices. Besides, it just wasn't a business model that was sustainable. With Google as the rule-maker, a simple rule change could wipe out an entire market. And as sponsored themes started to disappear from the community, theme designers and developers looked for new ways to support their hobby.

27. Update notifications

In July 2007, Peter Westwood became a core committer, bringing the number of people who could commit code to four. Westi got involved in the project in 2004, when, while working as a software engineer, he had written a script that downloaded WordPress and installed it on a server. He helped with the Codex and finding and fixing bugs. Like many other developers, he had little PHP experience when he came to the project.

The wp-hackers mailing list was in to its third year. Over time, productive discussion on the mailing list diminished. More decision making happened elsewhere. For example, in February 2007 [Ryan integrated phpmailer with WordPress](#). After only a short discussion on Trac, the code was committed. It wasn't until September that a [phpmailer discussion happened on wp-hackers](#) -- a discussion that few committers participated in.

Less development discussion happened on the mailing list. It inclined toward [meta-discussion about the mailing list](#), about the high signal to noise ratio, and about how to [improve mailing list etiquette](#).

Andy Skelton wrote about the problem with wp-hackers:

There is just one thing I want to make clear about wp-hackers: a hacker is not someone who discusses or pays lip service or dissents or casts a vote or says what can or should be done. Hackers aren't committee members. Hackers are more interested in proving what can be done than arguing about it.

On wp-hackers there was too much talking, not enough hacking, too much opinion, and not enough fact. During its life, a mailing list can become dominated by people who have time to comment, rather than those doing the work. Those doing the work have the most valuable feedback but they're often too busy to keep up with a high-traffic mailing list.

WordPress isn't alone in this phenomenon. A [2013 study of the Apache Lucene mailing list](#) found that "although the declared intent of development mailing list communication is to discuss project internals and code changes/additions, only 35% of the email threads regard the implementation of code artefacts." As well as discussing development, mailing list participants discussed social norms and behavior. The study also found that project developers participate in less than 75% and start only half of the discussions, and that developers prefer to communicate on the issue tracking software -- a finding resonant with WordPress, when developers switched from the mailing list to WordPress Trac.

Mailing lists tend toward bike shed discussions. [^fn-1] The idea of bike shed painting is derived from [Parkinson's Law of triviality](#):

Parkinson shows how you can go in to the board of directors and get approval for building a multi-million or even billion dollar atomic power plant, but if you want to build a bike shed you will be tangled up in endless discussions.

An atomic plant is so vast that only very few people can grasp it. The board of directors assume that somebody has the knowledge and has done the groundwork. In contrast, everyone can build a bike shed, so everyone has an opinion on how it should be built, and, more specifically, what color it should be.

In his book, *Producing Open Source Software*, Karl Fogel notes that as the technical difficulty of an issues goes down, the "probability of meandering goes up." If a discussion is highly technical it can be difficult for everyone to follow, but there are other discussions which everyone can follow and engage with:

consensus is hardest to achieve in technical questions that are simple to understand and easy to have an opinion about, and in "soft" topics such as organization, publicity, funding, etc. People can participate in those arguments forever, because there are no qualifications necessary for doing so, no clear ways to decide (even afterward) if a decision was right or wrong, and because simply outwaiting other discussants is sometimes a successful tactic.

With its community growth and low barrier to entry, the WordPress project has been susceptible to bike shed discussions like

any other free software project. A Google search for “bikeshed” on the wp-hackers mailing list displays the discussions in which someone has cried “bike shed,” as does a similar search on Trac.

In 2007, for example, discussion about what to call links got heated. A member of wp-hackers asked, “Anyone know why it’s still called Blogroll in admin, when it’s called Bookmarks in functions (wp_list_bookmarks) and yet displays by default as a list of “Links” in the sidebar?” The original post, *Blogroll, Bookmarks, Links*, generated a total of 79 emails on the mailing list alone, and discussion spilled on to a Trac ticket.

Of course, one person’s bike shed is another person’s bête noire, and there were times that wp-hackers was alright with issues that the community felt ought to be taken seriously. One such instance was in early September 2007, before the release of WordPress 2.3, which contained the update notification system. This notifies users when a new version of WordPress or a plugin is available for install. The system collects the WordPress version, PHP version, locale setting, and the website’s URL from a user’s site and sends it back to <http://api.wordpress.org>. A further piece of code carries a plugin update check, which sends the website’s URL, WordPress version, and plugin info (including inactive plugins) to api.wordpress.org.

For some, this data collection is contrary to the free software project ethos. They thought that collecting the blog URL was unnecessary. A ticket on WordPress Trac requested that update checking be anonymized. Others had no problem with WordPress collecting the data, but were unhappy that WordPress did not disclose it. A number of people who didn’t oppose the changes still felt that there should be an opt-out option so that users who required complete privacy would be able to turn off data collection.

This debate goes to the heart of one WordPress’ design philosophies: “decisions, not options.” This idea is heavily influenced by a 2002 article written by GNOME contributor Havoc Pennington. Many free software user interfaces cram in options so that they can be configured in multiple ways. If an argument ensues in a software project about whether something should or shouldn’t be added, a superficial solution is to add an option. The more options one adds, the more unwieldy a user interface becomes. In his article, Pennington writes “preferences have a cost [..], each one has a price and you have to consider its value.” He outlines the problem with too many options:

- When there are too many preferences it’s difficult for a user to find them.
- They can damage QA and testing. Some bugs only happen when a certain configuration of options are selected.
- They make creating a good UI difficult.
- They keep people from fixing real bugs -- some software projects add options that are just there to fix bugs.
- They can confuse users.
- The space that you have for preferences is finite so fill it wisely.

WordPress developers carefully consider introducing any new option. Over time they have become better about making decisions about options. By the time data collection question arose, they were extremely reluctant to introduce new options. Westi posted to wp-hackers:

One of the core design ideas for WordPress is that we don’t introduce options lightly. The moment you think of making a feature optional you challenge the argument for introducing the feature in the beginning.

What benefit would an opt-out button for update notifications provide to users? The purpose of the notifications is to inform a user that a new version of WordPress, or a plugin, is available for them to install. This helps users to stay up-to-date and secure. Adding an option to opt-out of update notifications would only reduce the number of people updating their sites, and increase the number of insecure instances of WordPress. The benefit to adding the option doesn’t outweigh the cost.

Therefore, despite the extensive discussion on wp-hackers, WordPress 2.3 was launched as planned, with the following note in the announcement post:

Our new update notification lets you know when there is a new release of WordPress or when any of the plugins you use has an update available. It works by sending your blog URL, plugins, and version information to our new api.wordpress.org service which then compares it to the plugin database and tells you what’s (sic) the latest and greatest you can use.

The project also published the first version of a privacy policy on WordPress.org.

Amid these discussions, the project structure changed when Peter Westwood and Mark Jaquith became lead developers. The announcement post describes them as “a few old faces who are taking on bigger roles in the community.” Both were active committers who had taken on greater leadership roles. Both had participated in many of the more challenging discussions in the community, and they didn’t always agree with current project leadership. They both, for example, opposed Matt’s proposal for the taxonomy structure, and Mark was one of the people to voice concerns over data collection in WordPress 2.3. This funnel broadening brought new perspectives to the project leadership. For the first time since they early days of the project, there were people other than Matt and Ryan to help guide and shape development of both the software and the project.

[^Fn-1] The notion of a bike shed was popularized in FOSS projects by Karl Fogel. In his book, [Producing Open Source Software](#), he reprints [an email from Poul-Henning Kamp](#) to the FreeBSD mailing list with the title “A bike shed (any color will do) on greener grass...,” in which Kamp uses the “painting the bikeshed” analogy to describe discussions on the mailing list.

28. Happy Cog Redesign

Shuttle had failed and WordPress’ admin needed a redesign. Matt went to [design studio Happy Cog](#). Jeffrey Zeldman, Happy Cog founder, led the project with WordPress’s logo designer Jason Santa Maria, and UX designer Liz Danzico.

Whereas Shuttle focused on aesthetics, Happy Cog identified and corrected information architecture problems, and updated and improved WordPress’ look and feel. Despite the project’s user-first ethos, the admin screens had become cluttered over time, as new features were added in a relatively haphazard way. The change between WordPress 1.5 and WordPress 2.3 shows the features stuffed into the Pods Shuttle had introduced.

The screenshot shows the WordPress 1.5 'Write' screen. At the top, a navigation bar includes 'Write' (highlighted in blue), 'Edit', 'Categories', 'Links', 'Users', 'Options', 'Plugins', 'Templates', 'Profile', 'View site >', and 'Logout (site admin)'. The main area has a 'Title' input field, a large 'Post' text area with a WYSIWYG editor, and a 'Quicktags' toolbar with various HTML tags. A 'PingBack' checkbox is checked. Below the post area is a 'TrackBack' input field. At the bottom right are buttons for 'Save as Draft', 'Save as Private', 'Publish', and 'Advanced Editing >'. To the right of the post area is a sidebar titled 'Categories' with a 'General' option selected. At the very bottom, a link to 'WordPress 1.2 – Support Forums' and a note about the page load time ('0.02 seconds') are visible.

The Write screen in WordPress 1.5

The screenshot shows the 'Write' screen in WordPress 2.3. At the top, there's a blue header bar with the 'Blog' logo and a 'View site »' link. To the right, it says 'Howdy, admin.' with links for 'Sign Out' and 'My Profile'. Below the header is a dark blue navigation bar with links for 'Dashboard', 'Write' (which is highlighted), 'Manage', 'Comments', 'Blogroll', 'Presentation', 'Plugins', 'Users', and 'Options'. Underneath this is another dark bar with 'Write Post' and 'Write Page' buttons.

The main area is the 'Post' editor. It includes a 'Title' field, a rich text editor toolbar with various icons, and a 'Path:' field containing 'Tags (separate multiple tags with commas: cats, pet food, dogs)'. At the bottom of the editor are 'Save and Continue Editing', 'Save', and 'Publish' buttons.

To the right of the editor is a sidebar with several sections:

- Categories**: A dropdown menu with an 'Add' button. It also has a checkbox for 'Uncategorized'.
- Discussion**: A section with a '+' button.
- Post Password**: A section with a '+' button.
- Post Slug**: A section with a '+' button.
- Post Status**: A section with a '+' button.
- Post Timestamp**: A section with a '+' button.

Below the sidebar are three expandable sections with '+' buttons: 'Optional Excerpt', 'Trackbacks', and 'Custom Fields'.

At the bottom left of the main area is a 'WordPress Bookmarklet' section with instructions and a 'Press It - Blog' link. At the very bottom, there's a footer message: 'Thank you for creating with WordPress | Documentation | Feedback | Your WordPress 2.3 is out of date. Please update.'

The Write screen in WordPress 2.3

Happy Cog produced designs and WordPress developers coded them. The project included user research and an interface audit to identify WordPress' strengths and weaknesses which informed new structural and interface designs.

WordPress, as a free software project, was an unusual client for a traditional design agency. The Happy Cog team was mostly sheltered from the community, by both Matt and Jeffery, but they knew this was an entirely different type of client. [Jason Santa Maria says now:](#)

Any other client will have customers and their own community, but you really have to just manage the people inside of a company, whereas when you're dealing with an open source project, you deal with the people that you're talking with, but there's this whole gamut of other people that you will only ever get to talk to a small portion of. I think that that's really difficult.

Plus, I think that on an open source project like this, it's inherently different, not just because it's more of a CMS than an informational website -- the design needs are different -- but it's just a different kind of way to work, knowing that whatever you do probably isn't going to stick around for very long. It's going to continue to evolve and continue to be adapted.

Usually, in the very near-term as well, not even three-four months from now, but next week.

An audit and usability review were among the first steps. Liz Danzico researched and produced a 25-page research audit on WordPress. WordPress needed an admin that didn't intrude on the user. In the audit, she quotes [Mark Jaquith](#): "That's when I know WordPress is doing its job: when people aren't even aware they're using it because they're so busy using it!"

Liz spoke to Mark Riley, whose support forum experience gave him direct access to users' complaints. One of the major problems he highlighted was the clutter that had appeared in the write screen between WordPress 1.2 and WordPress 2.0. Features had been added haphazardly and then tucked into modules using the pods introduced by the Shuttle project. There were many actions a user could take on the Write screen -- many of which were confusing or hidden.

Discussion focused on navigation structure, including labelling, position, and functionality grouping. Should they go with an object-oriented navigation (Posts, Pages, Comments, etc.,) or an action-oriented navigation (Write, Manage, etc.,)? Liz's first hunch -- supported by the newly-launched Tumblr -- was that users preferred navigating by nouns. She felt WordPress' verb structure was confusing. In the first navigational structure iterations, both a noun version and a verb version were produced. In the end, however, and after limited user testing, the team went with a mixture of nouns and verbs: Write, Manage, Design, Comments. This meant that the functionality for different content types was scattered over different menu items -- to write a post, for example, the user would go to "Write" but to manage a post they would navigate to "Manage."

Happy Cog provided extensive and detailed proposals and research for WordPress, all of which was fed back to Matt as the client. Their reports display a sensitivity to web users, and an appreciation for simplifying and streamlining WordPress.

When it came to the design stage, Jason Santa Maria created three designs to present to Matt, who would choose one to move forward.

He designed three early mockups:

Blog Name

[Visit Site](#)[Write](#) | [Manage](#) | [Design](#) | [Comments \(36\)](#)
[Post](#) | [Page](#) | [Link](#) | [Category](#)

WRITE POST

Title

Permalink: <http://www.myblogname.com/archive/best-post-ever/> | [Edit](#) ?

Add Media ?

[Image](#) | [Video](#) | [Audio](#)

Post

[Visual](#) | [HTML](#) | [Preview](#)[**b**](#) | [*i*](#) | [link](#) | [b-quote](#) | [del](#) | [ins](#) | [img](#) | [ul](#) | [ol](#) | [li](#) | [code](#) | [more](#) | [lookup](#) | [Close Tags](#)

Publish Status ?

[Scheduled](#)Scheduled for:
06/2/07 at 12:30 PM [Change](#)[Save](#) | [Publish](#)Last saved as **Scheduled**
on 06/2/07, 12:30 PM

Categories ?

[All Categories](#) | [Most Used](#) | [+ Add New](#)

- Category 1
- Category 2
- Category 3
 - Sub-category 1
 - Sub-category 2

Tags ?

[Add new tag](#) | [Add](#)

Separate tags with commas

Recently Used Tags:

[Tag N](#), [Tag N](#)

Tags Used:

 Tag N Tag N

Mockup Number 1

Blog Name

Settings | Plugins | Users

Write

Write Post

Manage

Design

Comments 36

Title

Preview this Post

Permalink: http://www.myblogname.com/archive/best-post-ever/ [Edit](#) ?

Add Media ?

[Image](#) | [Video](#) | [Audio](#)

Post

[Visual](#) | [HTML](#)[b](#) [i](#) [link](#) [b-quote](#) [del](#) [ins](#) [img](#) [ul](#) [ol](#) [li](#) [code](#) [more](#) [lookup](#) [Close Tags](#)

Publish Status

Scheduled

Scheduled for:
06/2/07 at 12:30 PM [Change](#)[Save](#) [Publish](#)Last saved as Scheduled
on 06/2/07, 12:30 PM

Categories ?

[All Categories](#) | [Most Used](#) | [+ Add New](#)

- Category 1
- Category 2
- Category 3
 - Sub-category 1
 - Sub-category 2

Tags ?

 [Add](#)

Separate tags with commas

Recently Used:

[Tag N](#), [Tag N](#)

Tags Used:

 Tag N Tag N Tag N Tag N

Suggested Tags:

[Tag N](#), [Tag N](#), [Tag N](#)[Tag N](#), [Tag N](#), [Tag N](#)[See More Tags](#)

Advanced Options

[+ Short Excerpt](#) ?[+ Long Excerpt](#) ?[+ Comment & Pings](#) ?[+ Trackbacks](#) ?[+ Password Protect This Post](#) ?[+ Custom Fields](#) ?

Mockup Number 2

Blog Name

[Visit Site](#)[Settings](#) | [Plugins](#) | [Users](#)[Write](#) [Manage](#) [Design](#) [Comments 36](#)[Post](#) [Page](#) [Link](#) [Category](#)**WRITE POST****Title**Permalink: <http://www.myblogname.com/archive/best-post-ever/> [Edit](#) ?**Add Media** ?[Image](#) | [Video](#) | [Audio](#)[Post](#) [Preview](#)[Visual](#) [HTML](#)[**b**](#) [*i*](#) [link](#) [b-quote](#) [del](#) [ins](#) [img](#) [ul](#) [ol](#) [li](#) [code](#) [more](#) [lookup](#) [Close Tags](#)**Categories** ?[All Categories](#) | [Most Used](#) | [+ Add New](#)

- Category 1
- Category 2
- Category 3
 - Sub-category 1
 - Sub-category 2

Tags ?[Add new tag](#) [Add](#)

Separate tags with commas

Recently Used Tags:[Tag N](#), [Tag N](#)**Suggested Tags:**[Tag N](#), [Tag N](#), [Tag N](#)[Tag N](#), [Tag N](#), [Tag N](#)[See More Tags](#)**Tags Used:** Tag N Tag N Tag N Tag N Tag N Tag N**PUBLISH STATUS**[Scheduled](#)Scheduled for: 06/2/07 at 12:30 PM [Change](#) ?[Save](#) [Publish](#)

Last saved as Scheduled on 06/2/07, 12:30 PM

ADVANCED OPTIONS[Short Excerpt](#) ?[Long Excerpt](#) ?[Comment & Pings](#) ?[Trackbacks](#) ?[Password Protect This Post](#) ?[Custom Fields](#) ?Thank you for using [WordPress](#) | [Feedback](#) | [Get Version 2.2](#) | © Copyright 2007

Mockup Number 3

Matt chose was the visually lighter design. Jason had taken Shuttle's heavy blue palette and lightened the interface. An orange accent color complemented the lighter blues.

When the design was finally signed off, WordPress trunk started to transform. As change happened [community feedback](#) was posted to [wp-hackers](#). Was the admin actually more usable? Major changes broke familiar patterns. Some comments make comparisons with Shuttle; a few community members [want to implement Shuttle's interface](#).

Just before release, a [sneak peek](#) was posted to the development blog, and WordPress users installed the release candidate to get a better look. The response was similar to that of the rest of the community. Regular WordPress users were surprised by the

functionality changes to the Write screen. For some, it was a step backward for usability.

The screenshot shows the 'Write Post' screen in the WordPress 2.5 admin. At the top, there's a navigation bar with 'Dashboard', 'Blog' (with a 'Visit Site' button), 'Howdy, admin! | Log Out | Help | Forums', and links for 'Write', 'Manage', 'Design', 'Comments', 'Post', 'Page', and 'Link'. On the right, there are 'Settings', 'Plugins', and 'Users' links. The main area is titled 'Write Post'. It includes a 'Title' input field, a rich text editor toolbar with 'Visual' and 'HTML' tabs, and a media library 'Add media' button. Below the editor is a text area with a placeholder 'Path: p'. To the right of the editor is a 'Publish Status' section with 'Unpublished' selected, an unchecked 'Keep this post private' checkbox, and a 'Publish Immediately' button. Below this are 'Save' and 'Publish' buttons. A 'Related' sidebar lists links like 'Manage All Comments', 'Manage All Posts', etc. On the left, there are sections for 'Tags' (with an 'Add new tag' input and 'Add' button) and 'Categories' (with 'All Categories' selected, showing 'Uncategorized' and 'Most Used'). At the bottom, there's an 'Advanced Options' sidebar with expandable sections for 'Excerpt', 'Trackbacks', 'Custom Fields', 'Comments & Pings', and 'Password Protect This Post'. The footer contains the WordPress logo and the text 'Thank you for creating with WordPress | Documentation | Feedback | Get Version 3.8.1'.

The Write screen in the WordPRess 2.5 admin

WordPress 2.4 was the first version to fall significantly behind the 120-day release cycle. Versions 2.2 and 2.3 were mostly on schedule, but the Happy Cog redesign brought in major changes to the codebase that meant delaying the release. Rather than just delaying release, a version was skipped, moving straight from WordPress 2.3 to WordPress 2.5, which was scheduled for release in March.

But there were bigger problems with the release cycle than the delay. The community felt that they had not been consulted; they felt disenfranchised from the process. In some ways, the redesign was destined to fail before it even began. With no participation, no process, there was no way to assure community buy-in. Jeffrey Zeldman, reflecting on the process now says:

It worked for us, from our perspective, that we were a small team reporting to a single client who had life or death approval over everything. We had to please one user: Matt. He could say yes or no. We completely bypassed the community. That enabled us to get a design done that we felt was crisp and focused and achieved certain goals. And that sounds great. Except that, because the community wasn't involved, inevitably, the design then became unpopular because nobody got

their say in it. And if I could go back and do it again, I would involve them up front, and find ways to get my feedback without it turning into a committee clustercuss.

As with Shuttle before it, problems surfaced in the free software project's design process. Free software projects are collaborative enterprises; when part of that process moves behind closed doors to avoid the problems of working by committee, new problems arise. However much a project tries to avoid it, that committee exists and must have its say.

29. Premium Themes

In the wake of sponsored themes, theme developers were looking for ways to make money. Some made money from customizations, but increasingly people wanted to make money with WordPress-products rather than with services. Why build one site when you can build one theme and sell it multiple times? If theme developers couldn't sell links, they could do the next best thing -- sell themes directly to WordPress users. There was, however, some hesitation around this. When the project removed sponsored themes from official resources, some interpreted this as "WordPress doesn't support sponsored themes," some as "WordPress doesn't want me to make money," or "Automattic doesn't want me to make money."

Some people felt that Matt only wanted Automattic to make money from WordPress, a situation that wasn't helped when, after the sponsored themes debate, [Matt announced a WordPress.com theme marketplace](#), resulting in [cries of hypocrisy](#). Despite putting in the initial groundwork, Automattic decided not to go ahead. "It didn't seem right to go into it when we hadn't really worked out all the licensing issues yet," [says Matt now](#). [^Fn-1]

While the plan for the WordPress.com theme marketplace was set aside, employees inside Automattic were still interested in whether people would pay for a blog design. Noël Jackson ([noel](#)), a former Automattic employee, [recalls](#) the time when premium WordPress themes started appearing:

It was interesting, especially in the beginning when it started happening. It was interesting to see revenue figures for themes that were released as a premium theme -- not a WordPress.com premium theme -- but someone selling their theme and keeping the GPL license and thinking, "Well, people are actually buying this." There wasn't an issue with it. I think that made everybody happy.

Bloggers who used WordPress dabbled with making themes. They released them for free. "A funny dynamic happened," [recalls Cory Miller](#) ([corymiller303](#)), the founder of iThemes, "people started hitting my contact form and saying will you customize this, or will you do this theme, or will you do all that stuff, and I was like I'm, I'm kind of a fraud here. And so I started charging for my work." This is familiar tale some of the most prominent WordPress business founders tell today. Many of the people who run those businesses started out simply making themes. Often these were accidental businesses, with accidental business-people -- people who came to the community to scratch their own itch and found themselves inundated with requests for help, support, and customizations.

Although sites like Template Monster sold WordPress themes as early as 2006, it was the last quarter of 2007 that saw the first surge in WordPress premium theme releases from within the WordPress community. Brian Gardner's [Revolution](#) -- one of the first premium WordPress themes -- launched in August 2007. A few weeks later Tung Do released [Showcase](#). Later that year saw [Proximity](#) from Nathan Rice, [Solostream](#) from Michael Pollock, Charity Ondriezek's [Cornerstone Theme](#), Adii Pienaar's [Premium News Theme](#), Magnus Jepson's [PortfolioPress](#) and early the following year [Thesis](#), from Chris Pearson.

Most of the theme vendors followed a similar path to one another: blogger to hobbyist themer. From releasing a free theme to testing the brand new market. Few had career aspirations, it was just a side project to make money in their spare time. Some also did independent development for clients. The idea that selling WordPress themes could be big business didn't cross their minds. "Even when I quit my job to spend more time working on these themes," [says Adii Pienaar](#) ([adii](#)), co-founder of WooThemes, "and eventually rebranding and calling that effort WooThemes, I still thought that I was going to build a business

around the custom design/development stuff, around consulting."

Theme sellers weren't sure how to price and license their themes. There was no community consensus about licensing and nothing clear from the project. Theme authors created their own license. A common model was offering a single use license to use a theme on one site, and a developer or bulk license, to use a theme on more than one site. These licenses restricted the number of sites on which a theme could be installed, and what the user could do with a theme. Some, for example, required that the user retain the author credit in the theme, which enabled the authors to charge users to remove the attribution link.

Users download a free theme at no cost, or pay for a premium one. The label "premium," however, implies that paid themes are *intrinsically better*, as though attaching a price tag somehow elevates premium themes above other, free (and theoretically inferior) themes.

These commercial themes were often branded as "premium themes" but there was little consensus about what made a theme "premium." When WLTC asked, "What makes a WordPress Theme Premium," responses included: better documentation and support, less bloggy, better design, better features, better code, more functionality. Or, at least, that's what community members felt premium themes should include. Several community members argued that money was the only difference between free and premium themes.

For designers and developers making free and premium themes, what distinguished them was the time and effort they put into them, and the level of effort required to set them up. In [an interview in June 2008](#), premium theme developer Darren Hoyt talks about the differences between creating a free and a premium theme. He outlines his considerations:

- That hand-coding and tweaks to the files should be minimized with the inclusion of a custom control panel.
- That users shouldn't have to download plugins so all theme functionality was bundled with it. "We tried to make Mimbo Pro more than just a theme or "skin" and more like an application unto itself."
- That best practices for commenting code, valid markup, and well-written CSS are followed.
- That documentation be provided and evolve.
- That buyers get ongoing support and interactions with the developer.

Brian Gardner, designer of Revolution, [said in a March 2008 interview](#) that for him, a premium theme differs in how the site looks and acts. "If people respond to a site and say 'wow, I can't believe WordPress is behind that,' that's what premium themes are all about."

Attitudes varied widely between users, developers, and other community members. The [user who interviewed Brian Gardner](#) was a keen Revolution advocate. He commends the guaranteed support, comfort, safety, and the feeling that he's being looked after. A price tag doesn't necessarily equate to these things, but some premium theme developers make looking after their customers a priority and their theme sales rise. WordPress was no longer confined to the world of personal blogging. Businesses, particularly small businesses, used WordPress to build their websites. Paying a small fee for peace of mind is a worthwhile trade-off; it brings the promise of ongoing support. A free theme comes with no guarantees.

Some theme developers were frustrated by premium theme sellers. "'Premium,' when it comes to WordPress themes," [writes Ian Stewart \(iandstewart\)](#) in 2008, "simply means 'it costs money' and not 'of superior quality.'" The false dichotomy set up by the "premium" label undervalues themes released for free. Justin Tadlock ([greenshady](#)) released the Options theme to prove that there was [nothing to stop developers releasing a free theme fully-featured with bells and whistles](#). He was one of the first to explore an alternative business model, setting up [his own theme club](#). In contrast to theme sales, he wanted to create a community around theme releases and support. Later that year he [launched Theme Hybrid](#).

The issue of whether someone *should* be selling themes wasn't clear-cut. Some thought that they could either be involved with the community, or make money selling themes or plugins. A wp-hackers thread from March 2008 [asks for feedback on the premium theme concept](#). The author concludes that either a person can focus on making money or on contributing to the community. Mark Jaquith [points out](#) that the two aren't mutually exclusive. It's possible to make money with WordPress while still making a positive contribution to the project.

In the same thread, Matt outlines how he sees businesses best interacting with the WordPress community:

A GPL software community is like the environment. Yes you could make profit building widgets in your widget factory and not pay any heed to the forests you're clearing or the rivers your (sic) polluting, but in a few years you're going to start to deplete the commons, and your business and public perception will be in peril. [...] Contrast that with sustainable development, with giving back to the community that sustains (and ultimately consumes) your services and you have the makings of something that could be around for generations to come.

Matt didn't see the premium theme market as in step with the community. He [proclaimed premium theme sellers were in "murky waters,"](#) pointing out that there wasn't much money in the enterprise. In a [2009 conversation with Ptah Dunbar](#), Matt says that he doesn't like where premium WordPress themes are going, and that the premium theme market isn't helping the community to grow.

Whatever the community's perception, users pay for themes, and companies that sell themes make money. Take ThemeForest as an example. The marketplace sells themes and templates for WordPress and different CMSs, as well as HTML templates. Upon launch in September 2008, ThemeForest sold mostly HTML templates; WordPress themes were 25% of sales. By September 2009, WordPress themes were 40% of all ThemeForest sales. In 2013, WordPress themes were 60% of all sales.

The discussions weren't just about whether people should make money from WordPress. A bigger question was, how should themes be licensed? WordPress itself is licensed under the GPL. The GPL's copyleft distribution terms ensure that distributions and modifications of the software carry the same license. Derivative works also carry the license. Is a theme derivative of WordPress? There were arguments for and against, speculation and opinion -- much of which was based on interpretations (or second-hand interpretations) of the license. Court cases are rarely fought around the GPL. Those that come close are often settled out of court, as with the [free software Foundation vs Cisco lawsuit](#) which happened at the same as the WordPress community's debate about WordPress themes' status. The community was divided into two camps -- those who thought themes didn't have to be GPL, and those who thought they did.

There are a number of different factors. One is whether a WordPress install and its theme is an "aggregation" or a "combination." The GPL FAQ [distinguishes between the two:](#) an aggregate means putting two programs side by side (like on a hard disc or a CD-ROM). Some who argued that themes don't have to be GPL, said that WordPress with a theme is an aggregate. They believed that theme modifies the output of WordPress, but doesn't necessarily engage with its internals. WordPress core developers, however, stated [that themes use WordPress core internals.](#) For a theme to be completely independent it would need to use none of WordPress' internal functions.

For those who accepted that PHP was linked with WordPress in such a way that the theme inherited the GPL, there was still another possibility: distributing with two licenses, what people often refer to as a "split license" or, [what Matt described as, "packaging gymnastics."](#) While the PHP in a WordPress theme works with WordPress internal functions, the CSS, Javascript, and images, do not. Therefore, a theme developer can package their theme with two licenses, one for the PHP and one for the other files. For theme authors concerned about piracy and copyright this protected their intellectual property while ensuring they followed the letter of the GPL. For the project, however, this was simply an evasive game that did an injustice to the spirit of the GPL, i.e., that users be free to do what they wish with the software.

Piracy was a major concern for many theme authors. They worried that anyone could buy their theme, legally distribute it, and even sell it. Putting a proprietary license on a theme, however, doesn't stop people from distributing them. Themes from commercial sellers cropped up on torrent sites all over the web and [were sold on forums such as Digital Point for a cut price.](#) It was easy, if a user wanted, to download a commercial theme for free (though often it came loaded with additional links and malware).

Other fears were around making money. How could a business make money selling a product that gave the buyer the right to give it away, or even sell it, themselves? Designers and developers were reluctant to distribute with a license that they thought would damage their business. While Matt suggested that they look into doing custom development, theme sellers were interested in selling products. It's more immediately obvious how you can scale a business selling products than with services. After all, if you don't factor in support and upgrades, it's possible to sell a digital product ad infinitum, creating a relatively passive income for yourself. Time and resources constrain doing custom design or development for clients.

Revolution was the first theme to be distributed with a [100% GPL license.](#) Brian Gardner, Revolution's developer, has a typical

theme developer story -- he started as a tinkerer, did some contract work, started to sell themes, and realized that he could make a business out of it.

Developers often came to the project with an implicit awareness of the free software and hacker ethos; they were comfortable with sharing code. Theme sellers came through another route: first they were bloggers, then tinkerers, then designers. As opposed to developers, theme sellers believed they had to protect their designs. "I and others that were starting to sell themes at that point wanted to protect our work," [says Brian now](#), "we were almost scared of the open source concept where the design and images and the code we felt was all ours and really wasn't derivative of WordPress itself so we sold things on a proprietary level."

[Matt's Theme Shaper blog comment](#) made Brian reconsider his stance. Matt wrote:

I'm happy to give significant promotion to theme designers who stop fighting the license of the platform which enabled their market to exist in the first place, just email me.

This was the encouragement that Brian needed. He emailed Matt to tell him he was interested. Before making the switch, Brian [flew to San Francisco](#) with [fellow theme author Jason Schuller](#), to talk to Toni and Matt. They discussed why theme sellers were using proprietary licenses and their fears around piracy. Matt and Toni made it plain: they were happy for theme sellers to make money, but themes needed to be GPL. In all of the confusion around the GPL, the message had become mixed. Theme sellers thought that they were expected to give themes away, not just that they were expected to distribute themes matching WordPress' principles of freedom.

A few days after the meeting, [Brian announced that in the future, Revolution would be GPL](#). Jason Schuller followed suit.

In the midst of these debates, themes found a new home on WordPress.org. The theme viewer that had been hosted on [themes.wordpress.net](#) [was riddled with problems](#): themes had security holes, many of them had obfuscated code. The system was continually being gamed, there were duplicate themes, and many contained spam. The theme directory on WordPress.org was set up to rectify that; it was a place developers could host their themes, and where users could find quality WordPress themes. In July 2008, [the Theme Directory launched on WordPress.org](#), using bbPress (which, at that time, was not a plugin), making it easier than ever to distribute free themes.

Theme Directory

 Username * Password * [Log in](#) or [Register](#)
[Extend Home](#)
[Plugins](#)
[Themes](#)

- [Upload Theme](#)
- [My Themes](#)
- [More Info](#)
- [Contact Us](#)

[Ideas](#)
[Kvetch!](#)
[Popular Tags](#) [More »](#)
[fixed width \(8\)](#)
[two columns \(6\)](#)
[widgets \(4\)](#)
[custom header \(4\)](#)
[options page \(2\)](#)
[simple \(2\)](#)
[easy customization \(2\)](#)
[widget ready \(2\)](#)
[valid CSS \(2\)](#)
[valid XHTML \(2\)](#)
[white \(2\)](#)
[blue \(2\)](#)
[three column \(1\)](#)

Looking for the awesome WordPress themes? Here's the place to find them!

9 THEMES, 12,769 DOWNLOADS, AND COUNTING

[Search Themes](#)

Featured Themes



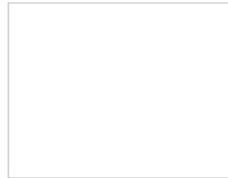
Tarski

An elegant, flexible theme developed by [Ben Eastaugh](#) and [Chris Sternal-Johnson](#).

[Download](#)


Dum-Dum

Candy looking theme.

[Download](#)


Monotone

A photo blogging theme. Colors change to match the photo.

[Download](#)

Get Hosted

Want to see your WordPress theme here? Go [add it!](#) All we require is that it be [GPL Compatible](#).

Not convinced? We've got [more info](#) that should persuade you.

Most Popular »

- [Tarski](#) Downloaded 4,520 times
- [Monotone](#) Downloaded 2,589 times
- [Modmat](#) Downloaded 1,910 times
- [Dum-Dum](#) Downloaded 1,664 times
- [BlueSky](#) Downloaded 516 times
- [Prologue](#) Downloaded 413 times
- [Gone fishing](#) Downloaded 378 times
- [Thistle](#) Downloaded 374 times
- [Happy Cyclope](#) Downloaded 336 times

Newest Themes »

- [Tarski](#) Added July 19
- [Gone fishing](#) Added July 18
- [Happy Cyclope](#) Added July 18
- [BlueSky](#) Added July 18
- [Modmat](#) Added July 18
- [Thistle](#) Added July 18
- [Dum-Dum](#) Added July 18
- [Monotone](#) Added June 30
- [Prologue](#) Added June 13

Recently Updated »

- [Tarski](#) Version 2.2
- [Gone fishing](#) Version 2.0
- [Happy Cyclope](#) Version 1.0
- [BlueSky](#) Version 1.0
- [Modmat](#) Version 1.0.2
- [Thistle](#) Version 1.0.2
- [Dum-Dum](#) Version 1.0
- [Monotone](#) Version 1.1
- [Prologue](#) Version 1.3

[Report a Site Bug](#) | [Privacy](#) | [GPL](#) | [Browse Happy](#) | [Fan WP on Facebook](#) | [WordPress Updates RSS](#)

CODE IS POETRY

The WordPress Theme Directory in 2008

[^Fn-1] The theme marketplace didn't launch until 2011, four years after the initial announcement.

Endmatter

January 2007 - March 2008

Releases: WordPress 2.2, 2.3, 2.5

Jazzers Stan Getz Michael Brecker

Features

WordPress 2.2 Widgets

WordPress 2.3 Tagging Update notifications Pending review feature

WordPress 2.5 New design Dashboard widgets Search includes posts & pages Tag management Plugin upgrades from the admin Galleries Secure cookies Easier taxonomy registration Shortcode API

Community

- Sponsored themes
- Growth of premium themes

Companies launched

- Crowd Favorite
- The Theme Foundry

Contributors

PART 5

30. Riding the Crazyhorse

WordPress 2.7 brought in what Ryan Boren describes as “the modern era of WordPress.” It also brought a new face to the WordPress project. When the reaction to WordPress 2.5’s redesign was less than positive, it was unclear whether the problem was inherent to the user interface, or whether people didn’t like it because they felt cut out of the process. Jen Mylo ([jenmylo](#)) [^fn-1], an old friend of Matt, was running a usability testing and design center at a New York agency in conjunction with Ball State University. The center ran usability studies and made use of the latest eye-tracking technologies. The agency’s clients included big companies like television networks such as ABC, NBC, and MTV. When a TV network didn’t have a product ready in time for a usability test window, Jen offered the slot to WordPress at cost.

The usability review had three main stages done in two rounds. In Round 1, WordPress 2.5 was tested, and recommendations made on “low hanging fruit” improvements to the admin. Following that, the development team created a prototype (Test1515) based on the recommendations to see if they improved users’ experiences. Finally, based on the Test1515 findings, the Crazyhorse [^fn-2] prototype was created and tested in Round 2.

The research team used three main testing methods in Round 1:

1. Talk-aloud, in which the participants are asked to think aloud as they carry out tasks.
2. Morae Software screen activity and video recordings of participants, which enables researchers to watch remotely.
3. Eye-tracking technology to identify granular problems.

The [twelve participants](#) tested a range of activities within WordPress’ admin. In Round 1, despite finding WordPress generally easy-to-use, the researchers identified a number of problems, including:

1. Verbs vs nouns: users found it difficult to conceptualize tasks as action-oriented (Write/Manage). Instead, they perceived content in a more object-oriented way. (Posts, Pages, Comments, etc.)
2. Users didn’t spend time on the Dashboard. They used it as an entry point for other pages.
3. The write post screen caused problems for users. Tags and categories appeared below the fold, and some participants forgot to add categories and tags before publication, returning to the post screen to add them afterward.
4. The comments screen was confusing. Users weren’t aware that they had to click a commenter’s name to edit a comment, and when asked to move a comment to spam they looked in the wrong place.
5. The difference between uploading and embedding media in the media uploader confused users.

They also discovered minor issues around settings, the media library, link categories, and tag management. Users also demonstrated a desire to have more control over the modules on the dashboard and the post edit screen, for example.

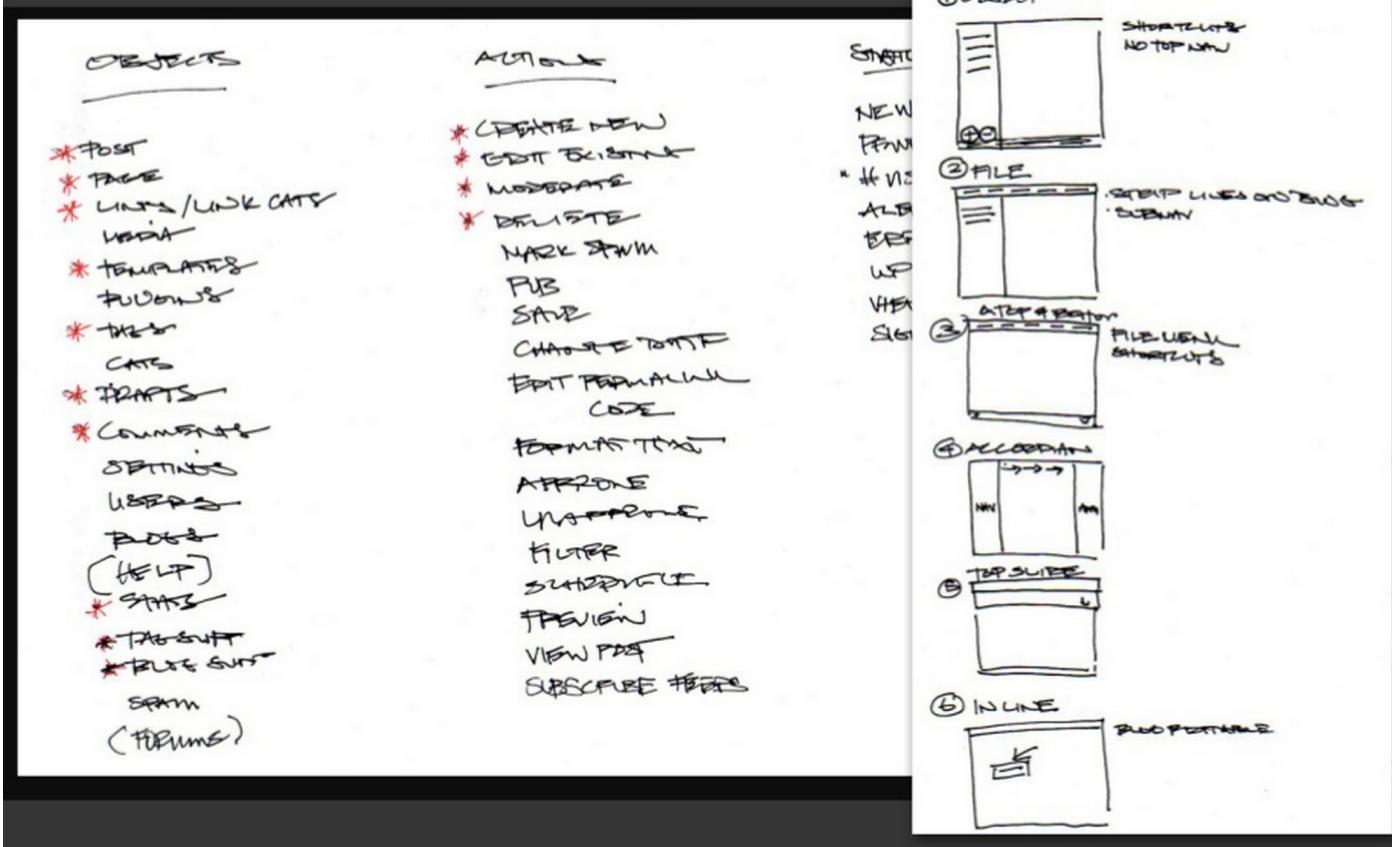
The [usability test report](#) says the following:

In addition to a laundry list of small interface issues that presented simple fixes, such as changing comment author links, we were faced with larger issues such as the desire for user-determined hierarchies on long/scrolling screens, ambiguity in the Write/Manage navigation paradigm, and a disconnect between the act of adding media to a post and the ability to manage it.

While minor changes were incorporated into the Test1515 prototype and tested, they were so minor that participants didn’t have a strong reaction either way. The team decided to create a more drastic prototype -- dubbed Crazyhorse -- to achieve the following:

- maximize vertical space,
- reduce scrolling,
- increase access to navigation to reduce unnecessary screen loads,
- enable drag and drop on screens that would most benefit from user control, and
- redesign management screens to take advantage of natural gaze paths.

Navigation groupings



Navigation sketches for Crazyhorse. Taken from Jen and Liz's WordCamp San Francisco 2008 presentation

Jen and Liz Danzico, who continued to work on WordPress' usability in the Crazyhorse project, created the design for the prototype. They sketched multiple ideas: front-end editing, accordion panes, and a top navigation. They chose the simplest prototype: a left hand navigation panel, similar to Google Analytics and other web apps.

WordPress developers built the Crazyhorse prototype in a Subversion branch, based on the prototype document which outlined the changes and their rationale. The project focused on user experience and functional development, so the prototype retained WordPress 2.5's visual styles. As in Round 1 testing, participants carried out tasks, and progress was assessed using talk-aloud, Morae, and eye-tracking.

Most responses demonstrated a preference for Crazyhorse over WordPress 2.5 and every new feature tested provided actionable information for the next version of WordPress.

Participants loved the navigation's new position, at the left-hand side of the screen, for example. They also preferred the object-oriented approach to organization. (Posts, Pages, Media, etc.)

WordPress Prototype 1.1

Dashboard, 3-column Alternative

The screenshot shows the WordPress Prototype 1.1 dashboard with a 3-column layout. The top navigation bar includes links for WP, My Account, My Dashboard, New Post, Comments (6), Help, Howdy, User Name! | Log Out, and < Browse WP Blogs >. The main content area is titled "Test Blog Name" with a "View Site" button. The left sidebar contains sections for CONTENT (Posts, Comments, Media Library, Links, Pages), TEMPLATES (Themes, Widgets, Edit CSS), and UTILITIES (Inbox, Stats, Tags, Categories, Users, Settings). The right sidebar features a "WordPress Community" section with developer blogs and a "Tag Surfer" section showing recent posts by various users. The central column displays three main widgets: "Inbox" (listing recent comments and posts), "QuickPress" (a form for creating new posts), and "Statistics" (an overview of site performance with a large red X placeholder for the chart).

The Dashboard in the Crazyhorse prototype

The new Dashboard was considered more useful, and people liked the ability to customize it. They liked QuickPress, though they weren't sure if they would use it. With action links now positioned under the comments, users found it easier to edit and act on comments.

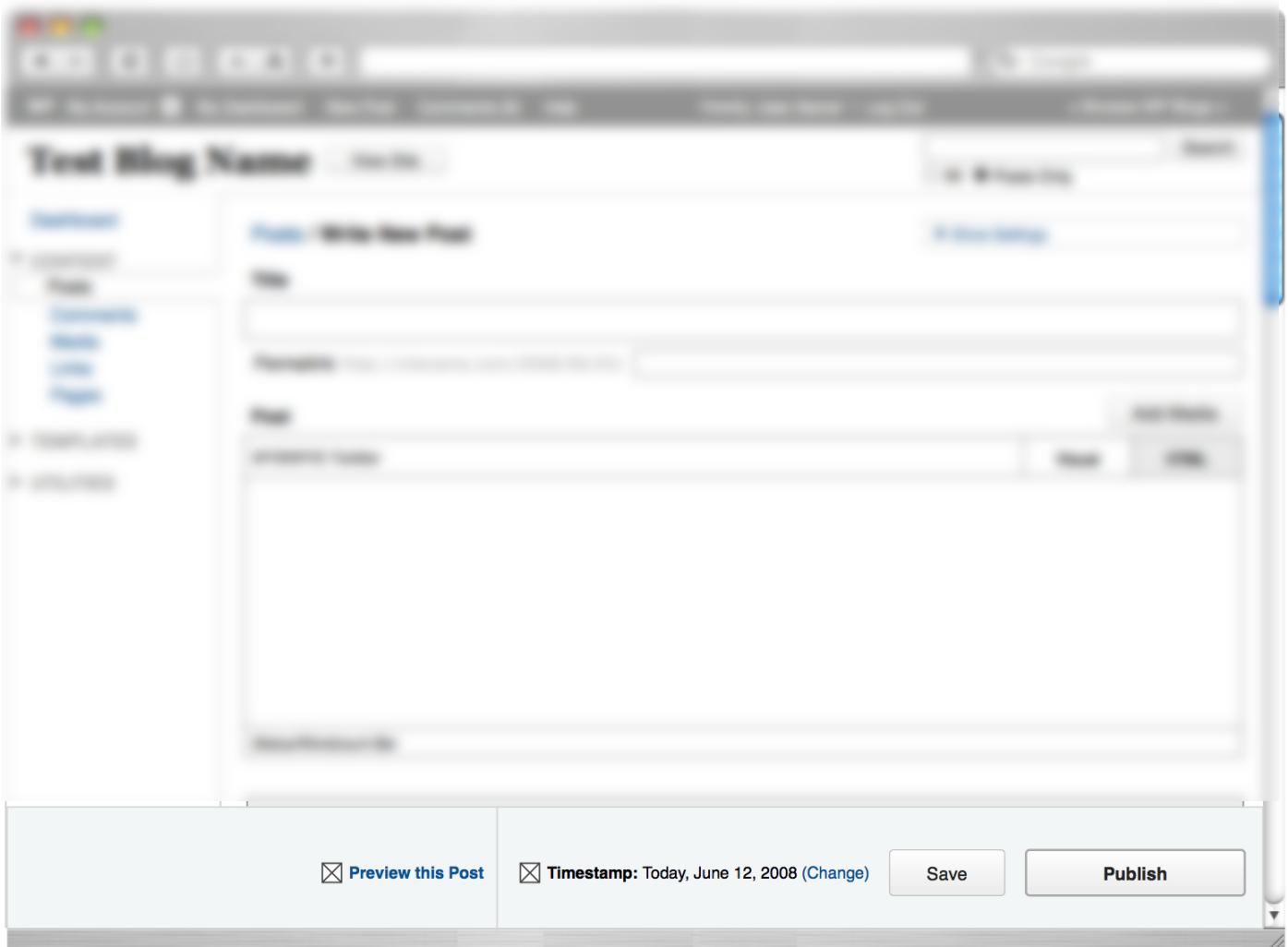
WordPress Prototype 1.1

Write New Post, Alternative 01

The screenshot displays the 'Write New Post' screen in the WordPress Prototype 1.1. The top navigation bar includes links for WP, My Account, My Dashboard, New Post, Comments (6), Help, and a user message 'Howdy, User Name! | Log Out'. On the far right, there's a search bar and a link to 'Browse WP Blogs'. The left sidebar contains sections for CONTENT (Posts, Comments, Media, Links, Pages), TEMPLATES, and UTILITIES. The main content area is titled 'Posts / Write New Post' and includes fields for 'Title', 'Permalink' (set to <http://sitename.com/2008/06/03/>), a large 'Post' text area with a 'WYSIWYG Toolbar' (with Visual and HTML tabs), and a 'Status/Wordcount Bar'. Below the post area are 'Save' and 'Publish' buttons. To the right, there are several panels: 'Preview Post', 'Tags' (with a 'Add new tag' button and a note to 'Separate tags with commas'), 'Categories' (listing five categories with an 'Add new category' link), 'Trackbacks & Pings', 'Post Author', and 'Privacy Options'. A 'Comments on this Post' panel is also present. The bottom left corner of the main content area is labeled 'WordPress Footer'.

The Write Screen in the Crazyhorse prototype

The new Write screen had a drag and drop feature, which allowed users to decide which elements got prime screen real estate. They also liked having access to post comments and felt that the new media uploader -- with its clear integration with the media library -- was a huge improvement.



The bottom publish bar in the Crazyhorse prototype

A publishing bar that floated at the bottom of the screen received the only major negative reactions. Users tended to look at the bar a few times before realizing it contained the Publish button. Some users compared it to a banner ad or thought it was part of their browser.

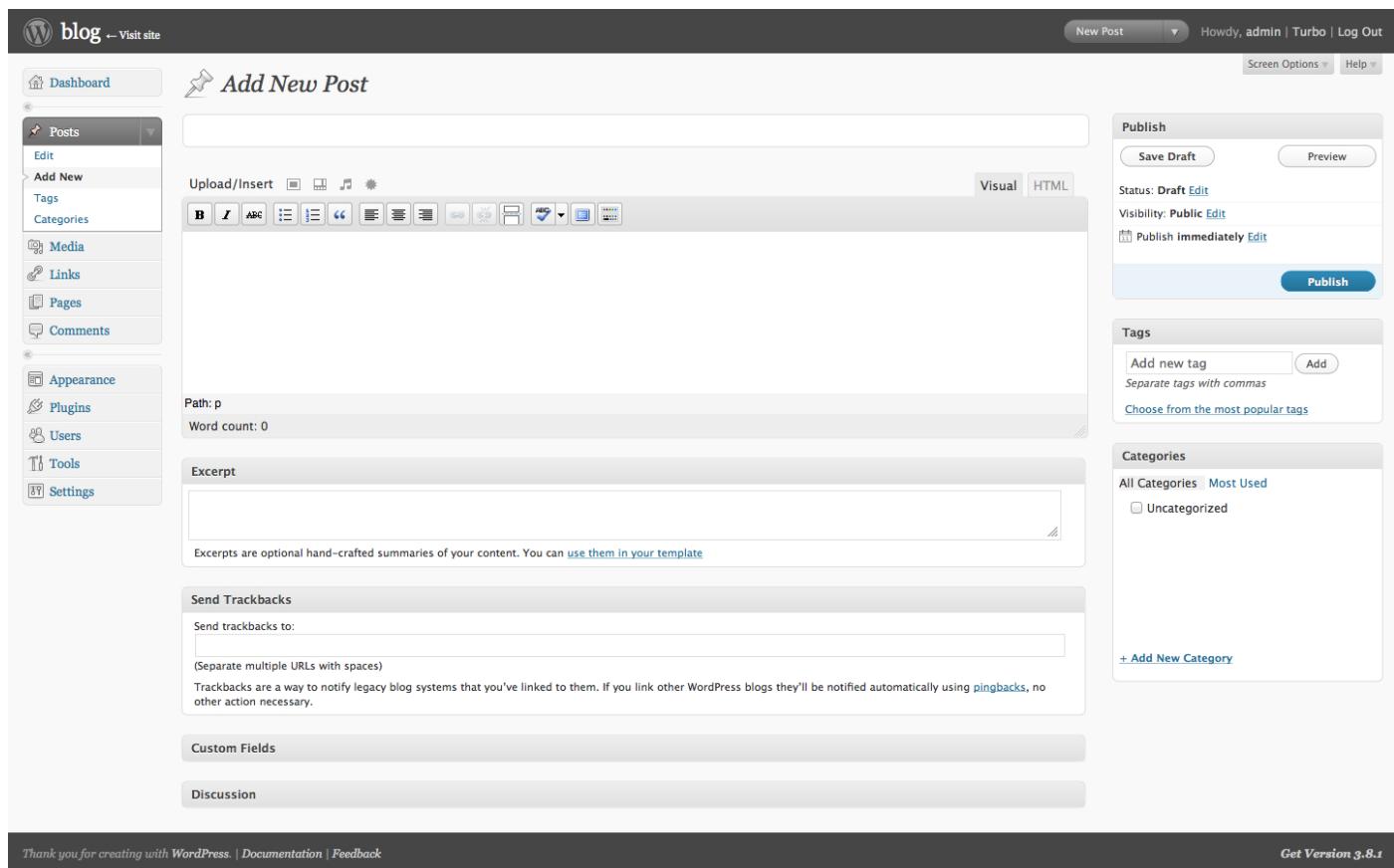
While Happy Cog and project Crazyhorse undertook user research, they ended up with quite different results. For Happy Cog, Liz conducted a series of interviews with community members. Later she conducted in-person user testing. The Crazyhorse project, however, used eye-tracking technology. This meant that the testers didn't have to rely solely on what participants said; instead they had insight into what the participants were looking at while they were making statements. Not only that, but gaze trails provided insight into how users navigated the screen with their eyes -- where were they drawn first? Were they missing important UI elements? Did they understand what they were seeing?

With the success of the prototypes, the design layer was the next step. When Crazyhorse was [first merged with trunk](#) it was just a set of live wireframes. No design changes were presented, which prevented participants from being swayed by color or typeface preference. By the Automattic meetup in Breckenridge, Colorado, in 2008, Crazyhorse was ready for some color.

At Automattic meetups, the company splits up into small groups to work on projects assigned at the beginning of the week. Everyone received assignments, and Matt Miklic, MT, ([iammatthomas](#)) [^fn-3] got Crazyhorse and free rein to give it the coat of paint it needed. Jen and MT spent the entire week redesigning the admin, producing many designs: a heavy blue reminiscent of Shuttle-inspired WordPress, and a version that used the light blue, grays, and orange of Happy Cog. In the end the two melded into a gray color scheme that became a feature of WordPress until 2013. Crazyhorse brought a huge number of changes to WordPress, with a completely transformed admin. To deal with all of these changes, Andrew Ozz ([azaozz](#)) was given commit access.

A marked distinction between the Happy Cog and Crazyhorse processes was Jen's connection with the community. She kept the community abreast of what was going on. The testing was carried out as an adjunct to the WordPress project, with little community involvement; the reason for testing was to verify actual usability flaws versus whether people simply didn't like the new design. If it had been just a matter of user color preferences, the Crazyhorse project would have come to nothing. But when it became obvious that the WordPress interface did need to change, a dialogue began in the community. The [designs](#) and the [usability report](#) were shared on the development blog. The team surveyed users on [navigation options](#), and the [community discussed issues on the wp-hackers mailing list](#). When WordPress 2.7 released, the [launch post](#) lists the posts Jen and the developers wrote about the process. Up to that point, it is the only iteration of the WordPress admin to have such an information trail.

With the Crazyhorse project, WordPress' admin changed drastically -- twice in 2008 alone. When [screenshots of the changes made their way onto community blogs](#), the inevitable question was "why are they changing it again?" WordPress 2.5's design hadn't settled in before another huge change came about in 2.7. A change in the UI meant that users of varying skill levels needed to completely relearn how to use WordPress. Not only that, with the growing community of people who had created tutorials with screenshots and videos on how to use WordPress, every screenshot would need to be retaken, every video re-shot. However, when WordPress users started upgrading the [feedback was positive](#). Users loved the new interface; they found it intuitive and easy to use, finally demonstrating that it wasn't change they had been unhappy with just nine months earlier, but the interface itself.



The Write Screen in the WordPress 2.7 admin

There were other major changes in WordPress 2.7. The WordPress Plugin Repository arrived on the admin screen. Users no longer had to download a plugin and upload it using FTP. They could search the plugin directory for the features they needed right from their admin, and install the plugin with just a few clicks. This made it much easier for WordPress users to quickly find and install plugins, removing the barrier to entry that came with FTP. There were also improvements to WordPress' documentation. In December 2006, Matt had posted to wp-hackers, [apologizing for his stance on inline documentation](#). [Inline documentation patches](#) were committed to core. In WordPress 2.7 [PHPDocumentor was added](#), through a big push by Jacob Santos ([jacobsantos](#)) and Jennifer Hodgdon ([jhodgdon](#)) to get WordPress functions documented in the code. There was also the beginning of an [official user handbook](#).

After completing the Crazyhorse redesign Jen joined Automattic to work on WordPress. At that time, the project was still small -- bigger than the early days of the cluster of bedroom hackers who built the platform, but not as big nor as diverse as it is today. There was a close group of core developers who led the project working in IRC and a Skype chat room. In [October 2008](#), Jen first appeared on [wp-hackers](#). She brought fresh eyes and a completely new perspective to the project. So far the project had been run by a group of hackers, many of whom were sympathetic to user needs, but who weren't always able to put themselves in users' shoes. Jen had a background in user testing and user experience -- something largely absent in the community until she joined the project. "Having someone with an actual formal training in testing and user experience, it was just useful," says [Mark Jaquith now](#)¹. "It was not just useful then, but it also changed us. Or at least it changed me. Where I started thinking in these ways as well and becoming better at stepping out of my own head."

This new focus on user testing and user experience meant that the project was able to hone its focus on users. With the number of WordPress users growing, this meant ensuring that the software didn't just work for a small group of bloggers but for everyone who wanted to install it.

[^fn-1] At that time Jen Mylo's name was Jane Wells. Her name changed in 2014. [^fn-2] Unlike WordPress releases, smaller projects, like Crazyhorse, don't always follow the jazz-musician naming convention. [^fn-3] At that time Matt Miklic's name was Matt Thomas. His name changed in 2014.

31. Themes are GPL too

As the project transformed, so did the wider economy around WordPress. When Brian Gardner released the Revolution theme under the GPL in late 2008, theme developers watched with curiosity to see how that would influence his business. It seemed counterintuitive to give a product away and run a business. What better way to test the waters than to see how someone else's GPL experiment would play out?

As theme authors watched and waited, commercial theme licensing came to a head. Just before Christmas 2008, 200 themes were pulled from the [WordPress theme repository](#). A statement was added to the [About page](#) of the theme directory that said "All themes are subject to review. Themes for sites that support "premium" (non-GPL or compatible) themes will not be approved." This meant that the Theme Directory no longer allowed free GPL themes that linked back to the site of a premium theme seller, whether the themes they sold were GPL or not.

Premium theme developers and [the wider community](#) were annoyed. What was the problem with theme authors selling themes? The themes hosted on the theme directory were free and complied with the GPL. As so often in the past, people conflated the actions of WordPress.org with Automattic and the company came under fire. Some perceived theme removal as a sign that Automattic didn't want anyone making money.

The theme sellers' rhetoric may have been hyperbolic, but theme sellers were genuinely unhappy. Their themes were pulled without warning. Many theme sellers saw their free theme releases as a way of supporting the free software community, and spent considerable time ensuring that they met the theme repository's standards. They felt as though the rug had been pulled from underneath them: they'd done their best to comply with WordPress.org standards but suddenly it wasn't enough.

An email from Matt made rounds in the community:

Thanks for emailing me about the theme directory. The other day I noticed a ton of bad stuff had snuck in like lots of spammy SEO links, themes whose sites said you couldn't modify them (which is a violation of the GPL), etc. Exactly the sort of stuff the theme directory was meant to avoid.

There were also a few that violated WP community guidelines, like the domain policy. So since Monday we've been clearing stuff out en mass. If you're kosher with the GPL and don't claim or promote otherwise on your site and your theme was

removed, it was probably a mistake. Give us a week to catch up with the bad stuff and then drop a note.

In a [podcast on Weblog Tools Collection](#), Matt discusses his position on the premium theme market ([transcription](#)). In the interview, he describes how, while updating a friend's website, he was looking for WordPress themes in the directory. He found themes with SEO links or linked to SEO sites -- behavior that the theme directory had been set up to avoid. The WordPress.org team questioned whether they wanted to allow GPL themes that only served as advertisements for non-GPL themes elsewhere.

In the interview, Matt discusses the distinction between "premium" and "free" themes, and the importance of correct labelling. When it comes to "premium" themes, Matt argues that word "proprietary" makes more sense. It doesn't make a value judgement that themes that cost money are better than those that don't. "Proprietary" is also more accurate than "commercial" or "premium" to distinguish a theme from a GPL theme, because GPL themes, such as Brian Gardner's Revolution, can be commercial.

I love what Revolution has done, where they say 'Ok, so we still sold the theme, and we still bundle the support and everything like that with it, but it's also available as GPL.' So they're able to, within the GPL framework, create a business and respect the underlying license of the community that they are building on top of.

Most importantly, Matt makes it clear that theme developers are free to do what they want on their site, but that the WordPress project is equally free to do what it wants on WordPress.org, and that includes whether it should or should not promote businesses that sell non-GPL WordPress products. Part of the aim of the theme and plugin repositories is to promote theme and plugin developers. The project doesn't want to promote theme developers who follow the license just to get on WordPress.org but who then violate WordPress' license elsewhere. Therefore, theme sellers who sell non-GPL products outside of WordPress.org are not promoted on the site.

The interview delineated what WordPress.org would and would not support. Even if people didn't like it they had their answer. The storm calmed and cleared the way for 2009, when businesses started to embrace the GPL.

In the months following the debate, discussions happened behind closed doors about how to sustain a business under the GPL. As the first to embrace the GPL, Brian Gardner advised other premium theme sellers. In April 2009, [Spectacu.la](#), the theme shop that first posted about themes being removed from the repo, announced that it was going fully GPL. It was followed in June by [iThemes](#) and [WooThemes](#).

In July, Matt announced that he had contacted the Software Freedom Law Centre. They [provided an opinion on theme licensing](#):
the WordPress themes supplied contain elements that are derivative of WordPress's copyrighted code. These themes, being collections of distinct works (images, CSS files, PHP files), need not be GPL-licensed as a whole. Rather, the PHP files are subject to the requirements of the GPL while the images and CSS are not. Third-party developers of such themes may apply restrictive copyrights to these elements if they wish.

WordPress themes are not a separate entity from WordPress itself. As Mark Jaquith [wrote later](#), "As far as the code is concerned, they form one functional unit. The theme code doesn't sit "on top of" WordPress. It is within it, in multiple different places, with multiple interdependencies. This forms a web of shared data structures and code all contained within a shared memory space."

Following that announcement, more theme sellers adopted the GPL license, though not all went 100% GPL (i.e., including CSS, images, and JavaScript). Envato, for example, whose marketplace, ThemeForest, was growing in popularity, opted [for GPL compliance -- with two licenses](#) in which the PHP was GPL but the additional files were proprietary.

Themes that are packaged using this split license follow the GPL. The PHP carries a GPL license and other assets do not. The other elements, (CSS, images, JavaScript, etc.) usually have some sort of proprietary license. This ensures legal compliance. It aims at protecting author rights by removing the freedoms guaranteed by the GPL -- the freedom to use, modify, distribute, and distribute modifications. Users are not free to do what they want with a theme licensed in such a way because the CSS and JavaScript are just as important to a theme as the PHP that interacts with WordPress' internals. But it would be a number of years before this debate about licensing would be had.

32. Improving Infrastructure

Jen brought a bigger emphasis on usability but she also encouraged more people to contribute to WordPress. While conversations happened on wp-hackers and Trac, it wasn't always obvious how new people could get involved with the project. Project growth compounded the problem; it wasn't always easy for someone new to understand project structure, what was going on, and how to contribute. Jen [wrote about how to get involved](#), and the development blog announced [patch marathons](#) and [bug hunts](#).

Jen brought more structure; she reinstated weekly development chats with agendas, and prevented discussions from disappearing down rabbit holes. When she joined the project, core developers communicated in two main ways: via the IRC chat room and by private Skype channel. Skype's drawback is that it's closed -- it doesn't create opportunities for other people to get involved. Because the main blog at the time, wpdevel.wordpress.com was on WordPress.com and not WordPress.org it didn't feel official. There had been little project management; developers wrote code and pushed it out when it was ready.

This was at a time when Matt was stepping back. Ryan Boren led development, while Jen stepped up to project manage the software and took a leadership role within the community.

She tried to address communication fragmentation within the project. By 2009, project communication was happening in different places: Trac, wp-hackers, the #wordpress-dev IRC chat room, wpdevel.wordpress.com, and the WordPress.org development blog. Jen highlights the state of each of WordPress' primary communication channels: the #wordpress-dev channel had become mostly inactive, wp-hackers had thousands of subscribers but was often just a troubleshooting forum, the development blog was just used for official announcements, the wpdevel.wordpress.com blog (directed now to <http://make.wordpress.org/core>) housed team progress updates, Trac had become an unworkable mess with hundreds of irrelevant tickets, and the ideas forum contained many highly-voted but irrelevant threads.

A [forum thread discussed communication improvements](#). Suggestions include: a place other than Trac for people to raise things, a way to make it easier for people to write automated unit tests, allowing the community to vote on ideas, greater documentation integration for WordPress and Trac, adopting P2 for discussion, and using BuddyPress (a social-networking plugin that is a sister-project to WordPress) on WordPress.org.

Amid suggestions there are complaints: that no one official reads the ideas forum, that timezone differences make IRC meetups difficult to attend, that too many communication channels exist -- far too many for people to keep up with. Some complained about community governance and a lack of transparency. As Jen observed, "Your post just proves the point that communication is an issue. I would not say that WP lacks a clear direction, I would say that it simply hasn't been communicated properly."

Jen wanted to clarify how project decisions were made. It wasn't uncommon for people to toss feature requests into IRC chat. People needed to know what each communication channel was for and understand the correct channels to ask questions and request features.

Much development discussion had shifted from wp-hackers to Trac. Trac's main benefit is that it focuses discussion directly on the bug, feature, or enhancement at hand. That said, as more and more people started using it, it became -- just like wp-hackers -- susceptible to intractable, bikeshed discussions.

Smilies were at the heart of one such recurring discussion. In 2009, a [ticket](#) requested replacing WordPress' smilies with Tango/Gnome smilies. Ryan Boren committed the patch.

The smilies landed first on WordPress.com, which receives daily codebase updates. The [feedback was negative](#) and reaction on the Trac ticket spiralled; contributors were unhappy smilies had been changed without discussion. They argued that WordPress had [changed users' content](#), without giving them any say in it. The discussion spread from Trac tickets to [community blogs](#). Some wanted a public poll to aid the decision.

Ryan Boren eventually [reverted the change](#), saying: “Back to the prehistoric smilies that everyone complains about but evidently likes better. :-) I was a fool for not appreciating the explosive topic that is smilies, my bad.” [^Fn-1]

As well as trying to fix communication problems, some worked on improving documentation. To commit a patch in WordPress core, there is a review process. The WordPress Codex, on the other hand, is the opposite: any contributor can publish documentation immediately. This is a low-friction way to create documentation, but lacking rules and structure it became difficult to navigate, pages fell out of date, and, despite the efforts of the documentation team, it became a mess. User documentation is packaged up with developer documentation, often on the same page, and for many WordPress users and developers the Codex became less useful. In the meantime, WordPress tutorial blogs proliferated. In the absence of good, official documentation, people went elsewhere.

The Handbook Project sought to create references for theme development and plugin development. The team launched a [Trac instance](#) and [wordpress.com blog to manage the handbooks](#). It can be difficult to recruit long-term help on documentation. With the low entry barrier, many people make their first WordPress contribution through documentation, but as they figure out the project's contours they move on to contributing to core. The original handbook project was passed between different people, and by 2014 they were near completion.

In mid-2009, WordPress dropped the long-term security branch, something Mark Jaquith had maintained since 2006. The plan was to maintain it until 2010. However, during that period there were major changes to WordPress' security. Backporting those changes into the 2.0.x branch meant complex rewriting that could have introduced new bugs or instability. The developers found that few people were on 2.0; new feature proliferation meant that people upgraded more readily. The legacy branch continued until just six months shy of its 2010 target, and [was deprecated in July 2009](#).

[Long-term support branch \(LTS\) requests remain](#) -- specifically from big companies that use WordPress as a CMS -- though it's unlikely that WordPress will ever start a new branch. Matt [wrote to wp-hackers](#):

Not backporting is a conscious decision. I would rather invest 100 hours in backward compatibility in a new version than 2 hours in backporting a fix to an obsolete version of WordPress. Plus, as noted by everyone else, backporting was often impossible because it wasn't one or two line fixes, it was architecture changes that would touch dozens of files. The LTS was actually *less* stable with these "fixes" backported because it had almost no one using it.

Rather than supporting an LTS branch, the project focuses on easy updates and compelling features that entice users to upgrade. This iterative process has been constantly improved through the project's history, from Matt's first mention of upgrades in his 2006 State of the Word address to automatic updates introduced in WordPress 3.7. This approach focuses development on keeping users secure, rather than trying to maintain an older software branches.

[Fn^1] The smiley debate was reignited in 2014, when WordPress.com updated its emoticons and a proposal was made to produce [high-dpi smilies for use on retina displays](#). Yet again, the arguments were brought out, and [spread once again to community blogs](#). As of 2015, the smilies are still not updated.

33. Meeting in Person

In December 2009, the core team -- Matt, Mark, Ryan, Westi, Andrew, and Jen -- met in Orlando. Despite having worked together on the project for years, it was the first time that some of them had met in person. They [discussed live project issues](#), including “the merge, canonical plugins, the WordPress.org site, community stuff, and all the other things that are important but that we never seem to have time to address.” For the duration of the meetup, they worked from the Bohemian Hotel lobby in Orlando. As well as discussing WordPress' vision and goals, they had a Trac sprint that edged them closer to shipping WordPress 2.9.



From left to right: Mark Jaquith, Jen Mylo, Andrew Ozz, Peter Westwood, Matt Mullenweg, Ryan Boren

They [posted results to the WordPress news blog](#), highlighting the breadth of the discussion:

Direction for the coming year(s), canonical plugins, social i18n for plugins, plugin salvage (like UDRP for abandoned plugins), WordPress/MU merge, default themes, CMS functionality (custom taxonomies, types, statuses, queries), cross-content taxonomy, media functions and UI, community “levels” based on activity, defining scope of releases, site menu management, communications within the community, lessons learned from past releases, mentorship programs, Trac issues, wordpress.org redesign, documentation, community code of conduct.

Meeting up in person allowed them to bounce ideas around on issues and generally get work done. Jen recalls:

We sat in these red velvet chairs in the bar of the Bohemian Hotel in Orlando, and when it was time to eat we would go into the dining room and we would eat. And we'd come back and we'd work, and we were on our laptops and actually going through Trac as well. And doing bug scrubs, but then we would stop and we would have just conversations and we'd go outside maybe or we'd go out to lunch. And so we kind of mixed it up, and it was just so helpful. Both in terms of just getting to know each other better, and the actual work.

Meetups continue to be part of core development. Members meet at WordCamps or at dedicated meetups to do code sprints and to discuss the general project and development direction. They've involved small focused core team meetups, to large meetups involving the whole community. They provide an opportunity for people to discuss issues without the barrier of a screen, and also to take time away from work, to socialize, hang out, and generally get to know one another better. When community members meet, they generate new ideas and thrash out old ideas in detail. They aren't, however, without side-effects on the wider community. Meeting up in person, by its nature, excludes everyone who isn't present at the event. This means that some are excluded -- perhaps they can't travel to the event for budgetary or health reasons. It's important, therefore, in a free software project, to balance offline meetups with online activity which allows everyone to have a say.

The canonical plugins project was discussed at the first core meetup. WordPress' plugin repository was growing, and many plugins did the same thing. Large, complex plugins can transform WordPress, though some plugins had poor code quality, some were out of date. Sometimes a developer drops a plugin, leaving users without support or updates. This can be a big problem if a user relies heavily on the feature. The core development team felt that some plugins warranted a similar process to the core

development process, where a group of coders lead a plugin's development, deciding what goes in and what doesn't, in a similar way to how WordPress, BuddyPress, and bbPress are developed. Rather than having ten SEO plugins, or ten podcasting plugins, for example, there would be one canonical plugin, sanctioned by core with its own official development team.

WordPress' core development team supported the proposal, and [Jen wrote about it](#); the strong relationship between WordPress core and canonical plugins would ensure plugin code was secure, that they exemplified the highest coding standards, and that they would be tested against new versions of WordPress to ensure compatibility.

Much of the early discussion was about what to call this cluster of plugins. The community voted and decided on "Core plugins." A team that included Westi, Aaron Campbell ([aaroncampbell](#)), Austin Matzko ([filosofo](#)), Stephen Rider ([Strider72](#)), and Pete Mall ([PeteMall](#)) started work.

[Health Check](#) which checks a website for common configuration errors and known issues, [PodPress](#), a popular podcasting plugin which had been abandoned, and a proposed plugin to shift the post by email functionality from core into a plugin, were the first canonical plugins.

Plugin developers weren't so enthusiastic. They liked having ownership over their own plugins. The WordPress project was accused of trying to stifle the growing plugin market. Much of the discussion went on among developers, and how core plugins could influence them. Justin Tadlock [said](#):

There's been some great discussion here, but it seems a little one-sided. Most of the people leaving comments are developers. What would make the conversation much better would be to hear from more end users.

I will refrain from sharing my opinion until I can gauge what a larger portion of the user base is feeling because, quite frankly, that's the portion of the community that influences my opinion the most.

User feedback on the thread was positive: any help sifting through the mass of WordPress plugins is a great thing for users. Why should users have to sift through fifty contact form plugins when there could be one officially sanctioned plugin which would serve 80% of users? This could only have a positive influence on users. And the project did not intend to kill off the plugin market. Mark Jaquith [tried to calm developer fears](#): "Core plugins will be safe and stable," he wrote, "but limited in scope and probably a little bit boring and not completely full-featured."

The core plugins project, however, was beset by other problems. While there was potential, tools to work effectively across a broad group of plugins didn't exist. "We didn't have the toolset at the time to do it properly," says [Aaron Campbell](#), "without stepping outside of the WordPress ecosystem quite a bit and relying on something like a GitHub or something like that, which we prefer not to rely on for things that are really kind of the core of WordPress."

To properly manage core plugins, the WordPress project's infrastructure would have had to change. When the plugin repository was built it was designed with one author per plugin; developer collaboration tools didn't exist. While the plugin repository has become more flexible (it's now possible, for example, to have multiple plugin authors), traditional developer collaboration tools are still lacking. A system to pass contributions in, discuss patches, modify patches, and merge them are useful for multiple developers to work on a single plugin. To develop WordPress, tools have been built around the Subversion repository to make that possible, and the modified Trac instance encourages collaboration. But each core plugin would need a replica of the infrastructure that WordPress uses itself, and it became obvious that the plan was untenable.

To add to these problems, many of the developers were burned out from WordPress 2.9 and were taking a break with the promise of a major release cycle in WordPress 3.0. They didn't have the energy or time to commit to building core plugins as well as developing the core product itself. Despite the coverage the project got across the community, and despite the initial burst of energy, the canonical plugins project eventually fizzled out.

34. Update Notifications Pt 2

Community blogs became important gathering points for those who wanted to communicate outside the core project's official channels. One of these was WP Candy, a blog originally started by Michael Cromarty and later taken over by Ryan Imel. Another was WP Tavern, run by Jeff Chandler. As places that the community could discuss, share ideas, and vent -- outside the central project channels -- both WP Candy and WP Tavern hosted many major community debates.

WP Tavern also had a forum which was a focal point for debates. One of the biggest of these was around data collection.

When WordPress 2.3 shipped with update notifications, the initial debate quietened down. Lynne Pope ([Elpie](#)) re-opened the discussion in 2009 on the WP Tavern forums. She [also posted to wp-hackers](#). Lynne pointed out that Matt had said that data collection would be reviewed in WordPress 2.5. She discussed URL collection and asked whether WordPress had a need to collect them, suggesting that an anonymous identifier could replace the blog URL. In the discussion thread on WP Tavern, Lynne went further into her concerns:

wordpress.org is not a legal entity so there is nobody to sue if data is misused. You can't sue a community. There is no disclosure about what data is collected or how it will be used. People are just supposed to trust that volunteers working on an open source project can be relied upon to keep personal data private?

She [referenced WordPress.org being cracked in March 2007](#); if someone were to crack WordPress.org again, they would have access to all the data.

Since much of the data is freely available on the internet, many were unconcerned about the collection, but Lynne pointed out that what isn't readily available is the totality of that information, i.e., WordPress version, PHP version, locale setting, plugin information, and the website's URL as a package. In 2009, people were growing more concerned about internet privacy. People signed up for social media services that collect huge amounts of data which can be used to target advertising. A 2009 paper reported that personally identifiable information could be [leaked from social networks](#) and third parties could link that data with actions on other websites. At the same time, internet users became more vocal about privacy concerns and more likely to speak out about them. In February 2009, for example, when Facebook changed its Terms of Service to say that user data would be retained by Facebook even if the user quit the service, the company faced an outcry and was [forced to backtrack](#).

In the midst of all of these concerns about privacy, WordPress was a bastion of the independent web. If a person has privacy concerns, they can avoid using social media; though they can create a website using something like WordPress, on their own server -- with complete control over their own data. For some people in the community, data collection tarnishes this independence. There is potential for abuse, and even if there is trust in the people who have access to the data now, there's nothing to guarantee that someone else, with different motives, won't have access to that data in the future.

Mark Jaquith, who had originally opposed the data collection, [responded to Lynne on wp-hackers](#):

The more I thought about it, the more my knee-jerk objections faded away. Your server is doing an HTTP request, so the server knows your server's IP address. You can figure out what blog domains are hosted on that IP with a search on Bing or several other search engines. So if WordPress.org really wanted to know your URL, it could find it. Again, that's just based on the IP address, which you HAVE to send for HTTP to work. If your URL is discoverable, and your IP address has to be sent, withholding the URL doesn't actually get you more privacy, ultimately.

Mark's original fears eased as he became more involved with the core software development and development on WordPress.org. "I'd like to be able to say that there was some like big philosophical argument that won me over," [he recalls now](#), "but mostly I became closer and more involved with the core team and got more access to the backend to see that okay, nothing nefarious is going on here."

While wp-hackers saw some of this discussion, it mostly took place on the WP Tavern blog and forums. The discussion thread generated 291 responses -- the most popular post in the forum's lifespan. It was also [heavily discussed on WeblogToolsCollection](#) under a post from Jeff Chandler titled "Is WordPress spyware?"

The issue was [reopened on WordPress trac](#), and [was proposed for the development chat agenda](#). Some community members

felt that their privacy concerns were valid and weren't being taken seriously. In the thread, Matt posts that WordPress.org only stores the latest update sent but no historical data. Historical data is only held in aggregate so that statistics can be provided for plugin and theme developers.

Mark joined in the discussion on WP Tavern to share some of the reasons he changed his mind:

- an IP address, which must be sent by the server, is not significantly more anonymous than a URL.
- URLs allow WordPress to verify the identity of a blog. When URLs are hashed it's no longer possibly to verify the blog identity. Without proper verification, systems that involve plugin rankings based on usage or popularity are open to manipulation and abuse.
- the privacy policy was updated to cover api.wordpress.org.

The core developers stuck with their *decisions, not options* philosophy; no option was added to turn off update notifications. By the time it was raised again, in 2009, the project could apply another one of its philosophies to making the decision -- the 80% principle: if 80% of users find something useful then it belongs in core, if not then it belongs in a plugin. A number of plugins were created to [disable update notifications](#); while they did get some use, it was only a fraction of the number of WordPress users. The benefits to WordPress users of having a clear sign that they should update their website or their plugin was more important than adding a preference to satisfy a small number of people within the WordPress community.

WordPress continues to collect data about sites. The data is used in a number of ways. It affects decision making. The project can make informed decisions about which technologies to support. Using the data, it was possible to tell that, in 2010, around 11% of WordPress users were using a PHP version below 5.2, and that fewer than 6% of WordPress users were using MySQL 4.0. Using that information, the development team could be confident about [dropping support for PHP 4.0 and MySQL 4.0](#). Browser usage data helped in the decision to deprecate support for Internet Explorer 6. The data helps when there is a security issue with a plugin. The project can detect how many sites have a plugin active and get a handle on the severity of the issue. In the case of a security issue in a popular plugin, web hosts are informed so that sites with insecure versions of a plugin can be blocked at the host level. Update notifications were also an important stage in the road towards automatic updates for minor releases, which were introduced in WordPress 3.7.

35. The WordPress Foundation

Throughout this time, Automatic held WordPress' trademarks. Trademarks are an important asset -- to companies, to free software projects -- to anyone who has a product to protect. The trademarks represent a project's reputation -- it's a symbol of official endorsement. Free software licenses protect the terms under which a program can be distributed, but they don't grant trademark rights. In a [2009 article on trademarks in open source projects](#), [Tiki Dare and Harvey Anderson](#) report that of the 65 licenses endorsed by the [Open Source Initiative \(OSI\)](#), 19 don't mention trademarks, 19 prohibit trademarks in publicity, advertising, and endorsements, and a further 26 explicitly exclude trademark rights.

Trademark law protects a piece of code's marks and branding -- not the code itself. A software project's code exists in its community's commons, but the trademarks do not. The growing issue of free software trademarks was conceded in 2007 with [GPLv3](#). One clause states that the license may be supplemented with terms "declining to grant rights under trademark law for use of some trade names, trademarks, or service marks." This reflects that free software communities accept that a trademark is not necessarily linked with the software.

Users and consumers associate a trademark with the original project; the trademark denotes trust, quality, and dependability. It also helps verify identify: if you're dealing with a company bearing the Widget Company logo, then you expect to be dealing with Widget Company. This matters in the free software community as much as in the corporate world. A free software project's trademark carries certain assumptions: that the project maintainers and developers are either involved with or officially endorse

that product or service, and that it meets quality standards. “Being the source of code arguably matters more than source code in an open-source business,” write Dare and Anderson. “The code is easily replicated, as it is open, but the trust associated with source (or origin) is not replicable. Trademarks are all about source.”

Automattic registered the WordPress trademarks in 2006. But some contributors felt that they had as much right to the trademarks as Automattic -- those who had helped build the software or started their own local communities, before Automattic registered the trademarks. Some community members felt that the community owned the codebase and should own the trademarks, not the corporate entity. What Automattic did have, and the community at large didn’t, was the structure and money to protect the trademarks from abuse and dilution. This often came at the cost of good relations with the community.

Automattic had always intended to place the trademarks with the WordPress Foundation, which is a separate entity from the company. Automattic acted as a short-term trademark guardian, protecting the trademarks until another body could take over. This was made clear to the company’s investors at the outset. Phil Black, one of Automattic’s first investors, recalls knowing that the trademarks would not remain with the company. “It wasn’t like Matt was proposing something to us where we felt like a significant asset was being lost,” says Phil now, “a significant asset was being transferred to the right third party as we thought that it should have been.”

The Foundation counterbalances Automattic, providing checks and balances should Automattic ever be acquired. “Let’s say Evil Co. ran Automattic,” says Matt, “and Evil Co. only cares about making money. The balance between WordPress.org and the WordPress Foundation and WordPress.com is such that I think even Evil Co. would do the right thing with regard to the community and the code and everything.”

The Foundation took longer than expected to set up; various factors needed to be in place before it launched. For one thing, before the trademarks could be transferred, they needed to be properly secured and protected. Toni Schneider managed this during his early days at Automattic. It also took a long time to register the non-profit with the IRS. Because non-profits are tax exempt, it can be difficult to set one up. Matt wanted the Foundation to hold the trademarks, but simply holding trademarks is not considered a legitimate non-profit activity. Applications sent to the IRS were denied for several years. In the end, the WordPress Foundation received 501(c)(3) status as an organization charged with educating people about WordPress and related free software projects.

The [WordPress Foundation’s website](#) states its mission:

The point of the foundation is to ensure free access, in perpetuity, to the software projects we support. People and businesses may come and go, so it is important to ensure that the source code for these projects will survive beyond the current contributor base, that we may create a stable platform for web publishing for generations to come. As part of this mission, the Foundation will be responsible for protecting the WordPress, WordCamp, and related trademarks. A 501(c)3 non-profit organization, the WordPress Foundation will also pursue a charter to educate the public about WordPress and related open source software.

The WordPress Foundation was launched in January 2010. Automattic transferred the trademarks in September. As part of the transfer, Automattic was granted use of WordPress for WordPress.com, but not for any future domains. Matt was granted a license for WordPress.org and WordPress.net. As well as transferring the trademarks for WordPress itself, the company also transferred the WordCamp name. As with WordPress itself, this protects WordCamps as non-profit, educational events in perpetuity.

The [community was pleased](#) with decoupling WordPress the project from Automattic the company. It gave people more confidence that Automattic was not out to dominate the WordPress commercial ecosystem. Despite some initial confusion about how people were allowed to use the WordPress trademark, eventually it settled down.

In the same year, both Matt and Automattic explored different ways to support the WordPress project. Matt set up another company, called Audrey Capital. Audrey is the home for Matt’s investments and it allows him to hire developers for tasks that he doesn’t have time for; this separation offers some balance between people who contribute to WordPress and who aren’t at Automattic.

“The idea was to kind of have five people at Automattic working on WordPress core, five people at Audrey working on

WordPress core, and then I'll try to get the different web hosts to each hire a person or two each, and so there'll be between the three fifteen-ish people, full time on WordPress.org," says Matt now.

Developer Samuel Wood ([Otto](#)), was Audrey Capital's first employee. Matt and Otto share a love for barbecue. Otto was looking for someone to sponsor a BBQ team at the international barbecue festival in Memphis. Knowing Matt likes barbecue, Otto asked Matt to sponsor the team. Matt said yes, and came to Memphis for the festival. The following year, Matt sponsored the group again. This time, after the festival, Matt asked Otto to come work for him at Audrey. Andrew Nacin (who would go on to become a lead developer of WordPress) joined Otto, and since then, Audrey has hired three more people to work on the WordPress project.

At the same time, changes at Automattic would have an ongoing influence on the project. By August 2010, the company had more than sixty employees. The sheer number of people meant that the completely flat company structure was becoming unmanageable. The company moved to a team structure in which teams of people worked on different projects, themes, for example, VaultPress, or support. One of the new teams inside Automattic was the Dot Org Team. This team of people are dedicated to working on the free software project. Ongoing members of the team were Ryan Boren and Andrew Ozz, WordPress lead developers, Jen Mylo, formerly WordPress' UX lead responsible for the Crazyhorse redesign, and Andrea Middleton, who manages WordCamps. Teams are fairly fluid inside Automattic, and people come in to the Dot Org Team to work on the WordPress project, often bringing with them the knowledge and skills that they've developed elsewhere in the company. Employees can also do a "Dot Org rotation," which means that they work on the WordPress project for a release cycle and then move to a different team.

The Dot Org team has helped mitigate the effects of hiring from the community. Seven out of the first 10 Automattic employees came from the WordPress community, and over the years the company has hired a number of contributors. While this has been good for individuals and for Automattic, it hasn't always had a positive effect. When WordPress.com was almost the sole focus, contributors worked on the core project, which benefitted both the project and Automattic. Some early contributors who became Automattic employees found themselves spending less and less time on the project. Mark Riley, for example, who was employed to do support, found that he had no time to help out in the WordPress.org support forums. For other people, this has happened slowly, over time, as Automattic expanded into other products, going in one direction, while the core project evolved into a different place, one with governance and structures, wildly different from the days of throwing up patches, heated discussions on wp-hackers, and waiting for Matt or Ryan to commit code.

The Dot Org team has helped to form a bridge between the company and the community, ensuring that there are people within Automattic whose attention is 100% on growing the WordPress product and project. In 2014, the number of people working on WordPress from Automattic expanded even further. The Dot Org team split into two: a developer team that works on the core software and on WordPress.org (team Apollo), and a community team that is focused on events and the community (team Tardis).

36. WordPress 3.0

2010 was the year of WordPress 3.0, which brought software, development process, and project structure iterations. A much wider commit policy was put into place, and all these changes shaped how the project works today. WordPress' philosophy -- *deadlines are not arbitrary* -- was also seriously challenged.

New committers were added during the 3.0 release cycle. In January 2010, Dion Hulse, ([dd32](#)) received commit access. In the [post announcing Dion's commit access](#), Matt outlines a new goal for the project:

One of the goals for the team in 2010 is to greatly expand the number of people with direct commit access, so the emphasis is more on review and collaboration. Right now commit access is tied up with being a "lead developer," of which

we've always found a small group of 3-5 works best, but now we want commit to be more a recognition of trust, quality, and most importantly activity, and something that can dynamically flow in and out as their level of commitment (har har) changes and decoupled from the "lead dev" role.

This was a big change from when every patch had to go through either Matt or Ryan. Also significant was decoupling commit access from the lead developer role. There was a need to extend trust to contributors without necessarily giving them leadership roles within the project. While there were no formal rules, several contributors received commit access: Ron Rennick ([wpmuguru](#)), focused on multisite, Dion focused on HTTP, Daryl Koopersmith ([koop](#)), worked on front-end development, while Andrew Nacin ([nacin](#)) acted as a codebase generalist.

As well as a symbol of trust, this extended commit access meant that tickets and bottlenecks could be cleared quickly. When lead developers got distracted by life and work, Ryan Boren carried the load. Adding new committers removed the bottleneck, ensuring that tickets got reviewed and committed.

April 13th 2010, was WordPress 3.0's release date. But as with prior releases -- despite having a deadline -- development focused on scope and headline features, rather than the date. Three main features were defined in the [scope chat](#): merging WordPress MU with WordPress, menus, and a new default theme to replace Kubrick.

Merging WordPress MU with WordPress came for several reasons: WordPress MU was difficult to maintain, the code was 95% the same as the main WordPress codebase, and Donncha had to manually merge new features after a release. Also, WordPress MU hadn't gained attention from plugin developers -- it felt separate to the main WordPress project. To give users a clean upgrade path, WordPress MU merged with WordPress. An alternative was to add a multisite feature to core, though this would have left WPMU users languishing on an outdated piece of software that might end up being forked.

Ron Rennick assisted with the merge. He had been using WordPress MU for years for the homeschooling blogging network that he ran with his wife, Andrea. He had written a script which ran on a WordPress install to turn it into a WordPress MU install. He was able to reverse engineer his script to bring MU functionality into WordPress, ensuring that there was a utility to convert a single WordPress install to a Multisite install.

Ron ran a diff [fn^1] against the WordPress MU code, looked for differences in the codebase, and [merged them into WordPress core](#). Ryan Boren and Andrew Nacin cleaned up the code. Ron also features absent from WordPress MU in plugins, like [domain mapping](#) which was originally developed by Donncha.

The original plan was to allow users to upgrade to WordPress Multisite by just clicking a button. "The reason we decided not to do that," [recalls Ron](#), "is that a lot of the shared hosts would not have been happy if their users could just take any WordPress install, click a button -- without actually knowing anything about what was going to happen -- and convert it over to Multisite. The decision was made to actually make it a physical process that they had to go through." To change a WordPress installation into Multisite, WordPress users have to edit wp-config. They need basic technical knowledge.

Terminology was one of the biggest headaches surrounding the merge. In WordPress 3.0, the project decided to move away from the word "blog," and instead refer to a WordPress installation as a "site." More and more people were using WordPress as a CMS, so the "site" label felt more appropriate. However, in WordPress MU, the parent -- example.org -- is a site, while the subdomain blog.example.org is a blog. With WordPress MU and WordPress merging, which one was the site? An individual WordPress install, or a WordPress install that hosted many blogs? WordPress MU became WordPress Multisite, but that wasn't the end of it.

To complicate matters further, WordPress MU had a function `get_site_option`, which gets options for the entire network, and `get_blog_option` which gets options for individual sites. The functions, therefore, don't relate entirely to the user interface. That was just one of the functions that caused problems, [as Andrew Nacin noted](#).

Custom post types and custom taxonomies were two big changes in WordPress 3.0. The default content types in WordPress are posts, pages, attachments, revisions, and nav menus (from WordPress 3.0), and the default taxonomies are categories and tags. In WordPress 3.0, custom post types and taxonomies were given a user interface. So instead of being restricted to just posts and pages, developers could create themes and plugins that had completely new types of content for users: testimonials for a business site, reviews for a review site, or books for a book database website, as examples. This opened up totally new

avenues for theme and plugin developers, and those building custom sites for clients.

Menus was the big user-facing feature for WordPress 3.0. At the time it wasn't easy for people to add navigation menus to their websites. Menu plugins were popular, so much so that it was obvious that the feature met the 80/20 rule. (Would 80% of WordPress users take advantage of the feature? If yes, put it in core, if not it stays in a plugin.) The [menus ticket was opened at the start of January 2010](#). The first approach was to make the menus interface act much like the widgets interface. By mid-February, however, little progress had been made. Competing proposals existed on how menus should work. Ryan and Jen contacted WooThemes to discuss bringing their [custom woo navigation](#) into WordPress core. This was just days before the planned feature freeze on February 15th.

WooThemes' custom navigation plugin made it easier for users to add menus to their websites. Developer Jeffrey Pearce ([Jeffikus](#)), was introduced to the core team and assigned to work with Ptah Dunbar ([ptahdunbar](#)). The plugin needed to be modified for core, and core needed to be updated to prepare for the feature. At the time, core updated jQuery to match the version WooThemes' plugin used. The original WooThemes plugin created new database tables for the menus. As a general rule, WordPress avoids adding tables to the MySQL database. Instead, core developers used custom post types, which let them use existing core functionality.

However, Jeffrey struggled with a factor that every distributed free software project has to work with: time zones. Jeffrey is based in South Africa. core development work happens mostly on US time zones. The development chats were at 20:30 UTC, which was 22:30 in South Africa. Additionally, Ptah and Jeffrey kept missing each other on Skype.

An environment in which everyone had a voice and an opinion isn't something that WooThemes was used to within their development process. Adii Pienaar, co-founder of WooThemes, described the process as excruciating. One of the main points of contention was that WooThemes had originally put the menus in the center of the screen in the user interface, with the boxes to add menu items on the right hand side. WooThemes had put time into designing it that way but the [menu interface was flipped around](#) to match WordPress' left to right interface convention.

The menu integration was one of the first times that the project had worked directly with a commercial business (other than Automattic). While the process was not always completely smooth, both sides benefitted from collaborating. WordPress got its menu system. While not exactly the same as the menu system that WooThemes created, it accelerated development. WooThemes got the satisfaction of seeing its code used by every WordPress user. [Not everyone felt that WooThemes received adequate credit](#), though Jeff didn't share this viewpoint. "Just to have our name on the contributors' wall -- that to me was good enough," [he says](#). "It's nice just to be able to say, I built a part of WordPress. No one can ever take that away from me. That was recognition enough for me."

WordPress 3.0 ended up being a huge release, and while discussion around menus went back and forth, shipping was delayed again and again. By April, the core team was still sending around wireframes -- [discussing whether menus should be pulled from 3.0](#). Matt reiterated one of WordPress' key philosophies:

Deadlines are not arbitrary, they're a promise we make to ourselves and our users that helps us rein in the endless possibilities of things that could be a part of every release.

Feature-led releases meant delays. Menus caused the hold-up with WordPress 3.0, with prevarication over the user interface and implementation. The final release was packed full of features that included the new menus, the WordPress Multisite merge, custom post type and taxonomy UI, custom background and headers, and an admin color scheme refresh. Any of these features could have been pushed to the next release, but there was no willingness to do so.

One of the more controversial changes in WordPress 3.0 was a [new function](#) called `capital_P_dangit`. This function corrects the misspelling of WordPress from Wordpress to WordPress, capitalizing the letter p.

People felt that [WordPress was messing with their content](#) -- that transforming Wordpress into WordPress was the start of a slippery slope. For some, it was overbearing pedantry, for others, censorship. After all, if someone wanted to write Wordpress instead of WordPress, why shouldn't they? WordPress has no business changing what people wanted to write. Some saw it as incommensurate with the core freedoms of the project: [openness, freedom, and community](#).

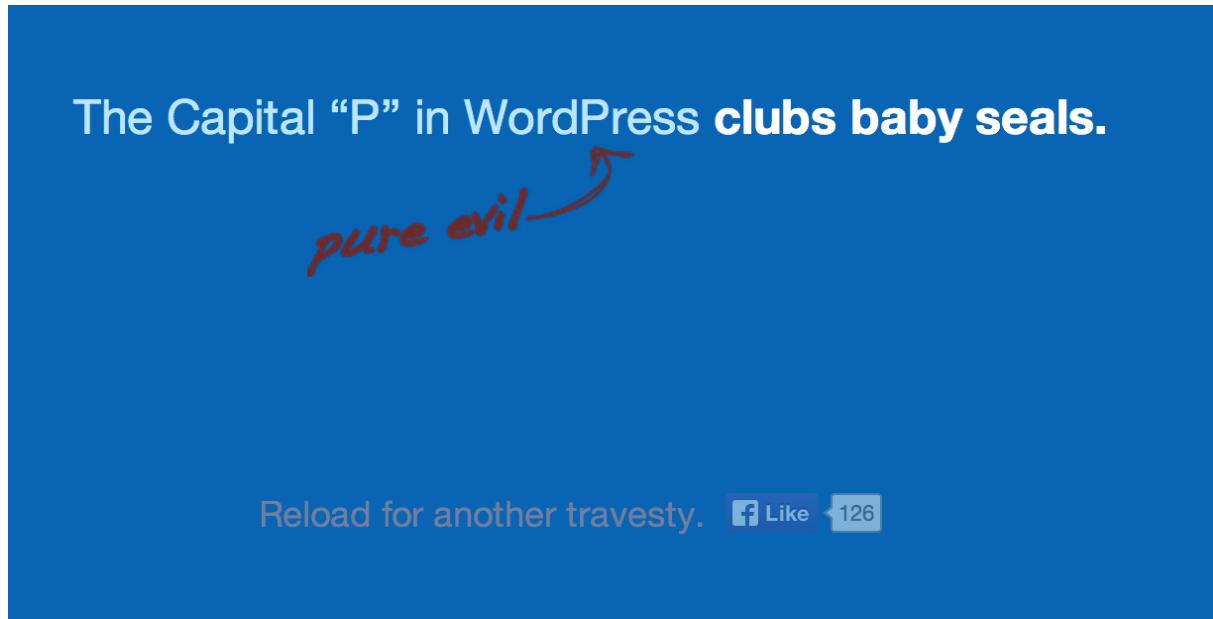
A bigger problem was that in some instances the filter was breaking URLs. This was reported before WordPress 3.0's release, but because the filter had already worked well on WordPress.com, it wasn't fixed. For example, a user reported on Trac that his image, named "WordpressLogo_blue-m.png" was broken because it had been renamed to "WordPressLogo_blue-m.png." Upon upgrading to WordPress 3.0, other users, those with folders named Wordpress, experienced the same problem. As well as folders, URLs with the spelling Wordpress were broken. Hosts saw an uptick in support requests. "When 3.0 arrived," says Mike Schroder (dh-shredder) of Dreamhost, "we had a deluge of support with broken URLs due to `capital_P_dangit()` applying to all content. This was a particular problem because it was popular among customers to use `/wordpress` as a subdirectory for their install. We helped customers with temporary workarounds in the meantime, but were very happy to see the issue fixed in 3.0.1." Mark Jaquith added a fix, but many contributors believed WordPress should never have broken users' websites in the first place.

The `capital_P_dangit` function isn't the only function in WordPress to filter content. Other filters include emoticons, autop, shortcodes, texturize, special characters, curly quotes, tag balancing, filtered HTML / kses, and comment link nofollows. From the core developers' perspective, changing Wordpress to WordPress didn't actually change the meaning of the sentence, except in edge cases such as the sentence "Wordpress is the incorrect capitalization of WordPress."

Core developers became frustrated by the hyperbole around the filter's inclusion and time spent arguing about it. In a comment on Justin Tadlock's blog, Mark Jaquith said:

Calling corrections censorship is absurd. It is no less absurd when the capitalization of a single letter is called censorship. There is actual censorship going on all around the world at this very moment. I'm damn proud of the fact that WordPress is being used to publish content that makes governments around the world afraid of the citizens who publish it. I'm incredulous that people are making a fuss about a single character (which is only one of dozens of automatic corrections that WordPress makes). It's free software that is easily extended (or crippled) by plugins. If the thought of going the rest of your life without misspelling WordPress it too much to bear, you have an easy out. Take it, take a deep breath, and try to pick your battles."

A website created by Mark reflects the position of many of the core developers on the `capital_P_dangit` discussion:



While this had all of the hallmarks of a bikeshed, there were some procedural issues that community members felt ought to be taken seriously. Whether the WordPress software capitalized the P in WordPress or not, the method with which the function was added to core broke accepted procedure: no ticket was opened, no patch uploaded to Trac. The code was simply committed. For some this set up an "us and them" mentality whereby some core developers could commit code as they saw fit, whereas everyone else in the community was subject to a different process.

Despite these development snafus, with WordPress 3.0, the platform matured, making "WordPress as a CMS" a reality. It also

introduced a new default theme for WordPress, ushering in a new approach with new annual default themes. Gone was Kubrick, with its bold, blue header. The new theme, Twenty Ten, showcased WordPress' brand new menus feature.

WordPress 3.0 ushered in changes to the project and the development process. It opened up WordPress to a new generation of people who became increasingly active. Over the coming releases, some of those committers would take on leadership both in terms of development and in the wider community.

[^Fn-1] - a diff is a comparison tool that compares the difference between two files.

37. The WordCamp Guidelines

By the time WordPress 3.0 came out in 2010, the WordPress community had grown as much offline as it had online. 107 WordCamps had taken place across the world, in countries as diverse as Thailand, Germany, the Philippines, Canada, Israel, and Chile. These events create spaces in which WordPress users, developers, and designers get together to listen to presentations and talk about WordPress. They also create opportunities for people to meet project leaders, and for those who have only ever met each other online to meet in person, socialize, and get to know each other better. WordCamps attract new contributors, and developers meet with users to learn about problems they have with the software.

WordCamps have always been informal events, and through the early growth of WordCamps, organization was just as informal as the events themselves. In the beginning, interested community members just decided to organize a WordCamp. They contacted Automattic for stickers, buttons, and other swag. A blog [for WordCamp organizers](#) was set up in 2009, so that organizers could communicate with one another.

In May 2010, Jen took over as [central WordCamp liaison](#), initiating changes to WordCamp organization. Jen said that "WordCamps are meant to promote the philosophies behind WordPress itself." Without any real structure and oversight, things happened contrary to WordPress's core philosophies. For example, WordCamps accepted sponsorship from people and companies in violation of WordPress's license, the GPL. While non-GPL compliant developers and companies are welcome to attend WordCamps, they're not able to organize, sponsor, speak, or volunteer. This is because WordCamps are official platforms of the WordPress project, and the project doesn't want to endorse or publicize products contrary to its own ethos.

There were issues with the way WordCamps were being run in the absence of central oversight. Organizers were left to get things done. Many WordCamps ran without budgets, despite being events that had a lot of money running through them. Some organizers had taken money for themselves; one WordCamp was set up, sponsorship money received, the WordCamp never happened, and the sponsorship wasn't returned. Another opened for registration just so that the organizer could compile a mailing list to which they could send marketing emails.

WordCamps needed better oversight, including clear guidelines. From May 2010 onward, that started to happen. In some quarters, [the changes went down badly](#). Others [were more relaxed about the changes](#) but they, too, asked why people who promoted non-GPL products would be banned from speaking. While many WordPress theme shops were 100% GPL, those that hadn't followed suit were indignant that they would have comply with the GPL just to speak at a WordCamp.

There were frustrations about the way that the guidelines emerged. They were published without consultation with WordCamp organizers and appeared to be an edict from above. Brad Williams ([williamsba1](#)) recalls now: "I think it probably would have been more beneficial across the board for some more open conversations between the Foundation and the organizers to make sure, one, that these guidelines make sense and that we're all on the same page and if there was any concerns get those out in the open." As with other past decisions, the guidelines weren't part of a conversation between the Foundation and the WordCamp community. When they appeared, they seemed unilateral and the rationale was poorly communicated.

Jen published [the first version of the WordCamp guidelines](#). They were that WordCamps should:

- be about WordPress.
- be open to all, easy to access, shared with the community.
- locally organized and focused.
- open to lots of volunteers.
- stand alone events.
- promote WordPress' philosophy.
- not be about making money.

Some guidelines were in response to problems within the community. It is important, for example, that WordCamps be about the software. At least 80% of the content should be about WordPress. Presentations about social media, SEO, and broader technology issues should not be the event's main focus. Others were more about ensuring that events about WordPress follow the same organization and ethos as the rest of the project. Like the project, WordCamps ought to be volunteer-driven, from the organizers, to the speakers, to the volunteers who help out during the day. The WordPress project was built by volunteers and it is the same with WordPress-events. WordCamps should also be accessible to anyone who chooses to attend. This means keeping the ticket price intentionally low.

The [guidelines have been refined over the years](#), with community feedback. While not everyone remains happy about them, they have not prevented WordCamps from flourishing across the world.

38. Dealing with a Growing Project

By the time WordPress 3.0 launched, more people were interested in the WordPress community and ecosystem than ever before. Always diverse, the community was becoming even more so. In the early days the project attracted developers, bloggers, and people interested in helping others via support or writing documentation. But several factors meant that people of different backgrounds were getting more involved.

WordPress 2.7 demonstrated that making software is not just about writing code; design, user experience, UI expertise, and testing are all very much part of the process. To create user-friendly, useful software requires a diverse set of eyes. And the project needs to attract other contributors in addition to developers who enjoy writing code and releasing it for free. This means looking for ways to engage new contributors by making clear the different pathways to contribution.

Jen Mylo had been trying to explain the pathways that someone could take to get involved. While development, documentation, and support are obvious ways to participate, in 2009, Jen wrote about the different ways that people can participate in the project. One was [graphic design](#). After a 2008 icon contest was successful, the project tried to find more ways for designers to contribute. A new Trac component for graphic design tasks was initiated, and designers were invited to get involved in WordPress' evolving visual design. People were also encouraged to get involved through [usability testing](#). WordPress 2.7's success was due to the user testing that was part of the development cycle. The development team was keen to replicate this process in future releases. The final way Jen outlined included sharing [ideas, feedback, and opinions](#).

As well as her work on the development blog, Jen and other project leaders went to WordCamps to speak about getting involved with the community. Community development was becoming as important as software development.

Project infrastructure changes also affected the project's growth, particularly around the third-party developer community. Both themes and plugins had become easier for users to access. Plugins usually received improvements more quickly: the plugin directory launched in March 2007, WordPress 2.3 later that year introduced the plugin update notification system, and from WordPress 2.7 onward, users could install plugins straight from their admin screens. Theme improvements were similar, just a little later: the theme directory was launched in July 2008, and the theme directory arrived on admin screens with WordPress 2.8 in 2009. Giving users more access to plugins and themes meant giving third-party developers more access to users than ever.

Both the theme and plugin directory were growing exponentially.

The theme directory was becoming more difficult to manage with the exponential theme growth. Joseph Scott ([josephscott](#)) developed the first version of the theme directory, and spent much of his time reviewing themes and providing feedback. A lot of the issues were related to security, but he also advised on best practices. There came a point, however, when the work was too much for one person.

Joseph [wrote](#):

The theme directory has been chugging along for more than a year now. During that time we've tinkered with the review process and some of the management tools, but haven't really opened it up as much as we'd like. It's time to rip off the band-aid and take some action; to that end, we're looking for community members to help with the process of reviewing themes for the directory.

No one knew how the experiment would turn out. A [rush of people signed up for the new mailing list; guidelines were drawn up](#). Though the overall response was positive, some felt that the guidelines were too restrictive, that some of the theme requirements should instead be recommendations. On the WP Tavern forums, Justin Tadlock outlined some concerns, including the fact that the guidelines didn't allow for themes with custom template hierarchies or custom image systems. Some people felt that the theme review team should restrict itself to looking for spam and other objectionable practices. Other developers simply decided to [remove their themes from the directory](#).

For those reviewing themes there were problems, particularly with manpower. The theme review queue was backed up with 100 themes by July 2010, with new themes being added every day. Only three to four people were actively reviewing themes. Over time, automated processes and new tools have improved the theme review process. For example the [theme check plugin](#) tests the theme against all of the latest theme review standards.

But despite the teething problems, the theme reviewers have managed to create a system that benefits both users and developers: users get safe themes approved by WordPress.org and theme developers get feedback and help on their themes. There have been other, long-term, benefits. [Joseph says now](#):

looking back on it over the long term it's been nice to see some folks who started way back then and have gone on to be very successful as far as developing their own themes, commercial themes, while still supporting free and open source at the same time. I think over the long term it's been rewarding to see that jumping off point for people to be able to continue to produce more themes, and very well regarded and high quality themes.

As the WordPress project diversified into concrete groups, communication needed to improve. core product development was discussed on [wpdevel.wordpress.com](#), and the other teams were scattered over mailing lists and [wordpress.com](#) blogs. Toward the end of 2010, a new blog network was set up on WordPress.org -- [make.WordPress.org](#) became the new home for WordPress contributor teams, with blogs for core, UI, theme review, and accessibility. Over time new blogs such as support, documentation, plugins, and community, have been added.

Each [make.WordPress.org](#) blog runs the P2 theme. Created by Automattic for internal communication, P2 is a microblogging tool that allows users to post on the front end, similar to Twitter, without the 140-character limit. Threaded comments on the post allow for discussion. Unless made completely open, only people who are editors of a blog can write a post while anyone is able to comment.

The "make" blogs are an important centralized space on WordPress.org for contributors to gather. If a contributor is interested in core development they can follow the core blog, forum moderators can follow the support blog, people with a UI focus can follow the UI blog. Contributors can subscribe to the blogs and follow along with what's going on from their email inboxes. While mailing lists are publicly archived, they often give the feeling of privacy because the contributor emails from their inbox and the mailing list archives are often difficult to find. By moving everything onto P2-themed blogs on [make.WordPress.org](#), the conversation's tone has changed -- everything takes place in public. This encourages a more respectful attitude among community members. The focus is on getting work done rather than devolving into arguments.

These sorts of focused communication improvements have helped the project to build capacity and grow. What often happens

is that a need appears, a call is put out, and if enough people answer the call, the project moves forward. Sometimes those projects, like the drive for core plugins, aren't successful, at other times they create a new area of contributor activity that people who weren't so involved with the project before can really participate in. This sort of community and capacity building is an important part of running a successful free software project. For a free software project that has a large user base to succeed, there needs to be enough volunteers to protect the long-term sustainability of the project. As the community grows, needs change and new contribution areas open up. A project that grows so far beyond its hacker roots that it encompass a diverse set of contributors is a healthy one.

39. Thesis

In mid-2010, one theme was still a GPL holdout: Thesis -- from DIYThemes. Created by theme designer Chris Pearson and blogger Brian Clark, Thesis was one of the most popular themes around. It's feature-heavy, giving users many more options than most WordPress themes. Users can customize every element of their website via the user interface. On the original about page [Chris states Thesis' aim](#): "I wanted a framework that had it all -- killer typography, a dynamically resizable layout, intelligent code, airtight optimization, and tons of flexibility." Tech blogs featured Thesis and high profile bloggers, including Matt Cutts, [Chris Brogan](#), and [Darren Rowse](#) adopted it.

Chris Pearson was well-respected in the community, having already developed Cutline and PressRow before moving into the premium theme market with Thesis. Thesis itself was influential to theme development. A [mid-2009 ThemeShaper post talks about "The Pearson Principle"](#): "Bloggers want powerfully simple design on an equally robust framework." With themes such as Revolution and Premium News, theme developers were already moving into creating themes packed with features, but Thesis cemented that move. It became the era of the theme framework [^fn-1], when a theme framework had as many features as WordPress itself, with heavy user customization available via the options panel. [Chris called this "ubiquitous design"](#) -- a theme was no longer just a website skin but a tool to build your website. It took another four years before theme developers stopped packing themes with options and started moving features into plugins.

While theme vendors adopted the GPL, Thesis held out. Discussions [between DIYThemes and Automattic](#) went nowhere and relationships were fractious. In June 2009, with discussions ongoing between Brian and Toni, a blogger's comment thread was [hijacked](#). A long debate about Thesis and the GPL ensued. Matt urges people to move away from Thesis, saying "if you care about the philosophical underpinnings of WordPress please consider putting your support behind something that isn't hostile to WordPress' core freedoms and GPL license."

Later, during three weeks in July 2010, the debate between WordPress and Thesis reignited after Chris Pearson's interview [on Mixergy](#). All of the major tech news sites took notice. In it, Chris shares Thesis' revenue figures, putting a conservative estimate at 1.2 million dollars within 16 to 18 months.

Just over a week later, [Matt and Chris took to Twitter](#). Matt was unhappy about Chris flaunting revenue while violating WordPress' license. Cutting remarks went back and forth until Andrew Warner from Mixergy decided to set up an [impromptu, live debate to discuss the issues](#). The hour-long discussion airs both sides of the argument. Matt argues that Thesis is built on GPL software -- WordPress -- and therefore should honor the license. By not doing so, he suggests that Chris is disrespectful of all WordPress authors and that he's breaking the law. For Chris, adopting the GPL means giving up his rights and losing his protection from piracy. He argues that "what I've done stands alone outside of WordPress completely," and that Thesis "does not inherit anything from WordPress." The argument descends into a rambling discussion of economics, and the conversation ends with Matt saying that he's willing to sue Chris if he's not willing to comply with the GPL.

Matt, Automattic, and WordPress took public action against Thesis following the interview. Matt offered to buy Thesis users an [alternative premium theme](#), consultants using Thesis were [removed from the Code Poet directory](#) of WordPress consultants, and Chris Pearson's other themes, Cutline and PressRow, were [removed from WordPress.com](#).

Matt wasn't the only person involved with WordPress to come out against Thesis. Other lead and core developers wrote about their GPL / Thesis stance. [Ryan Boren wrote](#), "where do I stand as one of the primary copyright holders of WordPress? I'd like to see the PHP parts of themes retain the GPL. Aside from preserving the spirit of WordPress, respecting the open source ecosystem in which it thrives, and avoiding questionable legal ground, retaining the GPL is practical." Mark Jaquith [wrote about how WordPress themes don't sit on top of WordPress](#) but are interdependent on it. A theme is combined WordPress:

in multiple different places, with multiple interdependencies. This forms a web of shared data structures and code all contained within a shared memory space. If you followed the code execution for Thesis as it jumped between WordPress core code and Thesis-specific code, you'd get a headache, because you'd be jumping back and forth literally hundreds of times.

Even developers who don't agree that a theme is necessarily derivative of WordPress thought that Thesis itself was derivative. Developer Drew Blas [wrote a script that took every line of WordPress and compared it with Thesis](#). His script pulled up a number of instances of Thesis code taken from WordPress. core developer Andrew Nacin [pointed out](#) that Thesis' own inline documentation read "This function is mostly copy pasta from WP (wp-includes/media.php), but with minor alteration to play more nicely with our styling."

A former employee of DIYThemes [left a comment on Matt's blog](#):

check out Thesis' handling of comments (thesis/lib/classes/comments.php). Large chunks of it are ripped right from WordPress. I know they are... because I'm the one who did the ripping. Whether I informed Chris of that or not doesn't matter because I no longer have any of our old chat logs to prove one way or another, but suffice it to say the latest public release of Thesis (and numerous versions before hand) contain obviously GPL code. Whether those portions get rewritten in the impending 3.0 release, I don't know... but for Chris to claim that he was responsible for and devised all of Thesis at 13:33 or so in the debate... Well, he was lying to you, either intentionally or not.

On July 22nd -- not even a week after the initial Mixergy interview -- [Chris Pearson announced](#) that Thesis would be released with a split license. The public furore compounded by pressure from inside DIYThemes forced Chris to capitulate. Brian Clark drafted [the license](#), shortly before leaving DIYThemes, citing "completely different opinions about the direction of the development of Thesis, the running of the company, and our relationship with the WordPress community." When Thesis 2 launched in 2012 it had a new, proprietary, license.

The debate around Thesis and the GPL had far-reaching implications on everyone involved. [Prominent blogs moved away from Thesis](#). Brian Gardner's Genesis theme became a popular choice. Thesis and Chris Pearson became less prominent in the community, focusing instead on cultivating and building a large customer base. The debacle also made it clear that WordPress is prepared to go to court when developers flagrantly abuse the license. There was, for a while, a relative calm in the community around the GPL. WordPress.org supported commercial theme sellers whose themes were 100% GPL and tolerated those that packaged their themes with two licenses. It would be another four years before the community found itself in another argument about the GPL, this time between the WordPress project and Envato. And when it did happen, it was focused on the spirit of the GPL, on the four freedoms, as opposed to the legality of the license.

[^fn-1] The term "theme framework" is often used to refer to different things. In some instances, a theme framework is a base, or "starter" theme that a developer can build from. In other cases it's a drop-in code library that facilitates development. But it's also used in marketing to users, when a theme framework is a feature-heavy theme with multiple options.

Endmatter

March 2008 - June 2010

Releases: WordPress 2.6, 2.7, 2.8, 2.9, 3.0

Jazzers

McCoy Tyner John Coltrane Chet Baker Carmen McRae Thelonious Monk

Features

WordPress 2.6 Post revisions Press This! Theme Preview

WordPress 2.7

Crazyhorse redesign Built-in upgrades QuickPress

WordPress 2.8

One click theme install

WordPress 2.9

Trash/undo Image editor Video embeds Post thumbnails

WordPress 3.0

WordPress Multisite Twenty Ten default theme Custom menus Contextual help Improvements to Custom Post Types & Taxonomies

Community

- Crazyhorse redesign
- Software Freedom Law Center decision on WordPress themes
- 1st core meetup
- WordPress Foundation launched
- WordCamp guidelines introduced

Companies launched

- Pagely
- 9seeds
- Odd Alice
- Pressable (formerly ZippyKid)
- WooThemes
- Inpsyde
- Human Made
- WP101
- WebDevStudios
- Event Espresso
- Array Themes (formerly Okay Themes)

Contributors

PART 6

40. The Transition to Release Leads

From late 2011 onward, the development process saw experimentation and refinement. The development process wasn't optimum: it lacked accountability, there were bottlenecks and missed deadlines. Each release from WordPress 3.4 saw a major change to the development process, and it wasn't until WordPress 3.8 that these experimentations in process settled down.

This experimentation period started at the core team meetup in Tybee in December 2011. At that meetup, most of the lead developers, committers, and active core developers discussed the project's issues and roadmap. This discussion informed the [scope setting meeting for WordPress 3.4](#) on January 4th 2012.

The scope chat notes cover the [breadth of issues discussed](#). The team acknowledged that there was a problem with process. Jen notes some of the issues: "lack of good time estimation, resource bottlenecks, lack of accountability, unknown/variable time commitments/disappearing devs, overassignment of tasks to some people, reluctance to cut features to meet deadline." The team proposed changes to solve these problems.

They decided to split feature development into teams, where two contributors would lead each team. This was to reduce instances where community members over-extended themselves and the chance that a contributor would abandon a project because of competing life priorities.

Ideally, these feature teams should be led by lead developer or committer paired with a newer contributor to help engage and mentor new contributors. Each team was responsible for posting regular updates and for delivering their feature on time. They created a schedule with overlapping cycles that encompassed development, UX, and bug fixes to reduce bottlenecks.

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8
Team A	UX	Dev	Test	Bug	UX Dev	Test Bug	UX Dev	Test Bug
Team B	Bug	UX	Dev	Test	Bug UX	Dev Test	Bug UX	Dev Test
Team C	Bug	UX	Dev	Test	Bug UX	Dev Test	Bug UX	Dev Test
Team D	Bug	UX	Dev	Test	Bug UX	Dev Test	Bug UX	Dev Test
Team E	Bug	UX	Dev	Test	Bug UX	Dev Test	Bug UX	Dev Test

The proposal for the 3.4 release process.

They also decided that releases would have a specific focus. WordPress 3.3 had been a cluster of disparate features, some of which had been pulled because they weren't ready. WordPress 3.4 was the first cycle to have an overarching focus -- appearance/switching themes. The development team worked on improving both front-end and admin features that adjust a site's appearance.

Between WordPress 3.4 and 3.5 the approach to project leadership evolved. These changes started back in WordPress 3.0 when Dion Hulse became a committer. Though Dion received commit access, he was not made a lead developer. This was the first step toward decoupling the lead developer position from commit access. The team felt that commit access, while being a symbol of trust and responsibility, should not necessarily lead to a person becoming a lead developer. To reduce bottlenecks, the project needed committers but not necessarily more lead developers. Separating the roles offered opportunities for strong coders and patch triagers. Additionally, some contributors may not be suited for or even want a more general leadership role.

And yet, some were confused about what the lead developer title actually entailed; individual core team members saw the role differently and no one agreed on what the lead developer title meant. For Mark Jaquith, the lead developer role has changed organically over the years. To begin with, the role related only to coding. [Over time it transitioned, particularly around the growth of WordCamps](#): "we started going around and talking to users about some of the philosophy behind WordPress and the direction we wanted to take. It became more of a general leadership role in practice." The role also evolved as lead developers worked with the community: often, lead developers responded to comment threads to address issues, or speak up on behalf of the project.

However, the role had never been formally clarified. While the lead developers handled many coding decisions, their additional responsibilities and authority were unclear. Did they have authority across the project? Should they make decisions in areas such as WordCamps, documentation, or theme review?

The lead developer role "didn't really have a set of responsibilities assigned with it or expectations," [says Matt](#). According to him, the lead developer leads development, not other areas of the project; he believes that coupling authority and responsibility with a development role makes it difficult for non-coders to achieve project authority and responsibility.

[Matt had articulated as much, as early as 2005](#), when he stated that commit access did not equate to community status. It may never have been Matt's intention to automatically confer authority on people with commit access, though extending commit demonstrated that committers had earned trust, and, as such, people naturally looked to committers as community and code leaders. Since roles and responsibilities were undefined, people simply perceived commit access as a general leadership role. The only non-coder who had an official leadership role in the project, other than the lead developers, was Jen, but she was an integral part of the core development team and there was no clear path for anyone to follow in her footsteps.

In June 2012, these differences and the confusion around the role brought conflict. Since WordPress 3.0 onward, a new generation of developers had been driving development. Contributor changes were marked between the 2010 core meetup in Orlando and the 2011 meetup in Tybee. In 2010 just the small team of lead developers (Mark Jaquith, Andrew Ozz, Peter Westwood, Matt, and Ryan Boren along with Jen) met up. In 2011 the meetup had new faces including Dion Hulse, [Jon Cave](#), Daryl Koopersmith, and Andrew Nacin.

From time to time, someone arrives and has a large influence on the project. In the early days, Ryan Boren drove WordPress' development, but post-WordPress 3.0, it was Andrew Nacin. "Nacin like Ryan is one of those guys that just has an ability to get in a flow, and just really crank and get focused intensely, and get through a ton of work," [says Matt](#).

Nacin's project influence grew and between WordPress 3.4 and 3.5. Ryan Boren proposed that since Nacin was driving releases just as strongly, he deserved recognition and should be made a lead developer.

Rather than appointing Nacin as a lead developer immediately, Matt proposed an organizational shift in the project. Matt argued that the lead developer title was historical and non-meritocratic, that leadership roles should be held by those driving the project today. Matt wanted opportunities for a new generation of developers to assume project leadership roles. He proposed that, instead of having a small group of lead developers, the project move to release leads, nominating Andrew Nacin and Daryl Koopersmith to assume the role in the next release. Matt's proposal offered clear authority to individuals for a given release; release leads would have final say, both over the lead developers, and over Matt. Some in the "historical and un-meritocratic" roles perceived this move as an attempt to remove the old guard to make way for the new. While a number of the lead developers aren't active in core on a daily basis, they are in regular contact with those who work on core, providing advice on architecture and development.

On reflection, Matt says that there was a misunderstanding. He said that he didn't mean to infer that the people holding lead developer roles were worthless or irrelevant, but that the roles did not reflect the reality of the project today. "A source of some of the conflict," [says Matt](#), "is this idea that the lead developers sort of had the same role as me where they had sort of purview over everything across all parts of WordPress."

The lead developer role discussion raised dissatisfaction around project decision making. Many of those who ran day-to-day development felt that unilateral decisions were made without consultation with the rest of the team. Matt had taken a less active role in recent years, as Jen and Ryan, supported by the other lead developers, drove the project, yet decisions had been made and announced without consultation. This was a culmination of other issues within the core development team and in the wider project: checking in code without consultation, providing feedback only toward the end of a release, and decisions around WordPress.org -- the site Matt owns, but that influences the entire community.

In response, some core team members -- including lead developers and other team members -- sent a group email to Matt to express their discontent with how the project had been run. They felt that Matt's perspective on the project's organization, authority, and responsibility, didn't reflect the project's realities. While Matt believed that the lead developer role was a code leadership position only, leadership extended beyond coding in practice. The group proposed an official project leadership team, and to retain the lead developer title as a meritocratic title for those core developers who had demonstrated leadership, code expertise, mentored new developers, and who had been strongly aligned with WordPress' philosophies. While the group happily supported the release leads proposal, they felt that the decisions for architecture, code, and implementation should rest with the lead developers, and that decisions affecting the project should lie with a leadership team.

In response, those involved scheduled a Google Hangout to discuss the issues, air grievances, and find a way forward. As a result of the hangout, some things changed, and the [first 3.5 release cycle post](#) reflects some of those changes. Andrew Nacin was promoted to the lead developer role, and core development adopted release leads with responsibility for an individual release cycle. The role of lead developer remains within the project.

They chose media as the scope for the 3.5 release; it was an area that the development team had wanted to tackle for some time but with such a large scope, it hadn't been attempted. Daryl Koopersmith created a plugin, called [Mosaic](#), which was the prototype for the new media management. Much of it was written in Javascript, where Daryl's expertise lay. But as a PHP free software project, there were few who could help him. As a result, Andrew and Daryl spent between 80 - 100 hours a week working on the release.

While the release itself was well received, and media received the overhaul it needed, the actual development cycle wasn't such a success. It was a huge amount of work, involving lots of feedback, codebase iterations, and coding a whole new feature from scratch. There were four major iterations to the user interface, including one 72 hours before the release. This meant that the new "release leads approach" got off to a faltering start. The intent was to have release leads guide and lead a release, not necessarily spend hours carrying out heroic coding feats. Once again, the release cycle was driven by focusing on a single feature and the only reason it was complete was because two coders broke their backs getting it done. But it was the next release cycle, 3.6, that made it clear that it wasn't the structure of the development team that was broken, but the actual model of developing features for a specific release.

41. The Community Summit

2012 saw the first en-masse invitation-only WordPress community get-together: The Community Summit. Up until then, contributors got together at WordCamps, and the core team had smaller, focused meetups, but there had never been a large community event focused on issues that face WordPress software development and the wider WordPress community. Community members could nominate themselves and others to receive an invite; a team of 18 people reviewed and voted on who would be invited. The attendees -- active contributors, bloggers, plugin and theme developers, and business owners from across the WordPress community -- came to Tybee Island, Georgia, to talk about WordPress.

The main event, held at Tybee wedding chapel, was a one-day unconference. A few informal days for working on projects were scheduled afterward. In the morning, attendees pitched suggestions for discussion, discussion groups formed around tables, and twice during the day individual groups shared their proposals for taking action.

The subjects discussed were broad, covering the spectrum of development and community issues. Development-specific topics discussed included mobile apps, improving deployments, using WordPress as a framework, multisite, JavaScript, the theme customizer, and automatic updates. There were also discussions about how to increase developer experience including better information for developers on UI practices. Broader community discussions were focused on the role of the WordPress Foundation, open sourcing WordCamp.org, the GPL, and women in the WordPress community. There were discussions about different WordPress.org teams such as UI, accessibility, theme review, and discussions about improving documentation. The summit attendees included people from all over the world and attendees talked about internationalization and global communities. Business owners raised issues such as managed WordPress hosts and quality control in commercial plugins. And, finally, there were discussions about making it easier for both individuals and businesses to contribute to the project.

With so much discussion, and many different ideas raised, some proposed ideas moved forward, while others were dropped without sufficient contributor support. Concrete changes that came about from the discussions at the summit include:

- Better theme review process documentation to increase consistency and transparency.
- A documentation and Codex roadmap. (developer.wordpress.org eventually launched.)
- Language packs included in core in WordPress 4.0.
- Headers added to the P2 themes to tell contributors what to do to get involved in that area.
- Published a make/events sub-team list.
- Automatic updates for core.
- Individual plugin reviews on WordPress.org.
- Open sourced the WordCamp.org base theme.

As well as creating a space for contributors to discuss issues, many contributors met for the first time at the summit, and the in-person meeting helped to invigorate the community.

It was around this time that a new team formed: a review team for the plugin repository. Up until that time, much of the work on

the plugin repository was done by Mark Riley, who was responsible for reviewing plugins. The community felt that plugins should be subject to the same rigor as themes. The question of plugin review was posed on both [community blogs](#) and on [wp-hackers](#). The number of plugins had grown rapidly and some people had concerns about code quality.

Otto started to review plugins too, and later Mika Epstein ([ipstenu](#)) and Pippin Williamson ([morduak](#)) joined in 2012, when the plugin review team started to become a proper team in its own right. Later Boone Gorges ([boonebgorges](#)) and Scott Riley ([coffee2code](#)) joined the team.

The plugin review team faces its own challenges -- different from those the theme review team faces. A theme is a specific group of template files with a defined structure. It calls functions, it requires a header, footer, and a sidebar. But a plugin can be anything at all, so there's no way to automate the review process. It's a lot of work. Yet the benefit of introducing this review process is that it has cleared out malicious plugins, spam plugins, and plugins with security holes. A set of [guidelines has evolved](#) to protect WordPress users.

Again, a small group of contributors created a team that addressed a specific need within the project. This has continued ever since the summit, including teams that formed to develop training programs for people who want to teach WordPress, moderators for WordPress.tv, and contributors who help to support meetups. The summit provided an opportunity for people to get together, to talk about their own interests, meet like-minded contributors and move projects forward. It was also a time for the community to be together as a community, to get to know one another socially, instead of just through the text medium that dominates communication online.

42. The Spirit of the GPL

By early 2013, much of the GPL discussion had settled down. While not everyone liked the fact that the WordPress project would only support 100% GPL products, most people had accepted it. Many, however, were taken by surprise, by a sudden flare-up around not just the legalities of the GPL, but the "spirit" of the license. In a 2008 interview, Jeff Chandler asks Matt about the spirit of the GPL. Matt says that the spirit of the GPL is about user empowerment, about the four freedoms: to use, distribute, modify, and distribute modifications of the software. Software distributed with these four freedoms is in the spirit of the GPL. WordPress was created and distributed in this spirit, giving users full freedom with regard to the software.

The Software Freedom Law Center's opinion -- with regards to WordPress themes, however -- gives developers a loophole, one that helps them achieve GPL compliance, but denies the same freedoms as WordPress. PHP in themes must be GPL, but the CSS, images, and JavaScript do not have to be GPL. This is how Thesis released with a split license -- the PHP was GPL, while the rest of the code and files were proprietary. This split license ensures that the theme is GPL compliant yet it isn't released under the same spirit as the GPL's driving user-freedom ethos.

The loophole may have kept theme sellers in line with the GPL, but WordPress.org didn't support that approach. [In a 2010 interview](#), Matt says "in the philosophy there are no loopholes: you're either following the principles of it or you're not, regardless of what the specific license of the language is." Theme sellers that sell their themes with a 100% GPL license are supported by WordPress. Those that aren't don't get any support or promotion on WordPress.org or on official resources. This is also one of the WordCamp guidelines, introduced in 2010; that WordCamps should promote WordPress' philosophies. If a speaker, volunteer, or organizer is distributing a WordPress product it needs to be 100% GPL, i.e., the CSS, JavaScript, and other assets need to be GPL, just like the PHP.

In early 2013, ThemeForest, the theme marketplace run by Envato, came under scrutiny. Envato is the company behind a cluster of blogs and marketplaces that sell everything from WordPress themes and plugins, themes for other CMS, to photographs, videos, and illustrations. WordPress is just one aspect of their business, though a significant one, and ever-growing. Envato was one of the theme companies in 2009 who brought themselves into compliance with the GPL, by releasing their themes with two

licenses: GPL for the PHP, and a proprietary license for the rest of the files and code.

ThemeForest has long been a popular choice for individual theme sellers. It offers exposure and access to a huge community of users. With the theme shop marketplace becoming increasingly saturated it became more and more difficult for new theme sellers to break through. Theme shops like StudioPress, WooThemes, and Elegant Themes dominate the market. ThemeForest offers everything a theme seller needs: hosting, sales tools, ecommerce, and a shop front. It provides a way for people to sell themes without having to do all of the associated set up work that can take up so much time. Besides, theme sellers make good money out of selling on ThemeForest. As early as December 2011, Envato [announced its first theme seller to make a million dollars in theme sales](#).

The matter came to a head in January 2013 when ThemeForest author Jake Caputo received an email from Andrea Middleton ([andreamiddleton](#)) at WordCamp Central. He was told that, as an author on ThemeForest, he was not allowed to participate at official WordPress events. Jake had already spoken at two WordCamps, had plans to speak at a third, and was helping to organize WordCamp Chicago.

The issue was over theme licensing and the WordCamp guidelines. WordCamps are official WordPress events that come with WordPress Foundation's seal of approval. [One of the stipulations](#) of being an organizer, volunteer, or speaker, is that you fully embrace the GPL. This goes beyond GPL compliance and means that you pass on all of WordPress' freedoms to your users. In short, that all of your WordPress-related products are 100% GPL. The guidelines state that you must:

Embrace the WordPress license. If distributing WordPress-derivative works (themes, plugins, WP distros), any person or business should give their users the same freedoms that WordPress itself provides. Note: this is one step above simple compliance, which requires PHP code to be GPL/compatible but allows proprietary licenses for JavaScript, CSS, and images. 100% GPL or compatible is required for promotion at WordCamps when WordPress-derivative works are involved, the same guidelines we follow on WordPress.org.

For vendors on ThemeForest, the only option available to authors was to use two licenses, in which the CSS, JavaScript, and images were distributed with a proprietary license. For Jake to become 100% GPL, he would have to stop selling on ThemeForest and find a new outlet for his themes. This meant losing access to the more than two million members of ThemeForest and a significant portion of his income.

WordCamp Central's actions angered some community members; some thought it was unfair to ask theme sellers to give up their livelihood simply to speak at a WordCamp. Besides, couldn't theme author sellers simply not mention Envato or ThemeForest in their presentations?

Others supported WordPress.org, seeing the move as consistent with their GPL stance. If a developer wanted to sell on ThemeForest it was their choice, just as it was the WordPress project's choice to support only 100% GPL products.

On both sides, people were frustrated for ThemeForest's authors. While the issue had little influence on the powers-that-be at WordPress or Envato, theme authors stuck in the middle were the ones to suffer. They had no choice about their license and moving off ThemeForest could jeopardize their livelihood in the short term.

The argument raged in the comments thread of Jake's blog, spiralling to other [WordPress community blogs](#), and to the [ThemeForest forums](#). Matt joined in the discussion on Jake's blog, [saying that](#) if ThemeForest authors had a choice about licensing and could release their theme under the GPL, then "Envato would still be breaking the guideline, but Jake wouldn't, so it'd be fine for Jake to be involved with WordCamps."

Collis Ta'eed, CEO of Envato, [responded on WP Daily](#), [^fn-1] outlining the reasons for Envato's licensing model. As a designer himself, Collis' main concern is protecting his designer's rights while ensuring that customers are able to use the resources they purchase.

As with so many disagreements in the WordPress community, it came down to a difference in emphasis. While the WordPress project emphasizes user freedoms, Envato emphasizes creators' rights. Both felt strongly that they had the moral imperative, and for either to back down would be to violate principles that underpinned their organization. The WordPress project places user freedoms over and above every thing else. If this meant excluding theme authors who sold on ThemeForest then so be it.

Collis, on the other hand, wanted to make sure that his author's rights were protected. His theme authors should feel confident that their themes were safe from piracy. He was also worried about having a GPL option for authors. He wrote "I worry that external pressures will force an increasing number of our authors to change their license choice, some happily, some not." Having just one (split) license meant that authors wouldn't be forced into that situation. Collis' professional background influenced his decision about licensing. He was a designer, and he felt that he had to protect other designers. The license at ThemeForest was one that made sense to him as a designer.

From the project's perspective, theme authors had a choice whether to sell their themes on ThemeForest, or sell their themes commensurate with the WordPress community's ethos (and thus speak at WordCamps). But in a podcast on [WP Candy](#), Jake talked about how he felt that he didn't have a choice about where to sell his themes. ThemeForest had become such an important part of his income that he would have to forfeit that income if he moved elsewhere. After the podcast, [Collis wrote a second post on WP Daily](#), in which he said:

I think I've been wrong in my stance. I would like to change that stance, and feel that ThemeForest should offer an option for authors, if they choose, to sell their themes with a GPL license covering the entirety of the theme.

He announced in the post that he would open a survey for ThemeForest authors to see if they wanted a GPL opt-in option. "I felt pretty guilty that our authors were paying some sort of price for selling with us, that felt pretty wrong," says Collis now. The results demonstrated that an equal amount of verified authors would license their themes under the GPL as those who wouldn't, though 35% of sellers said that they didn't know what they would do. It also showed, however, that theme authors who wished to distribute their themes under the GPL were mostly unconcerned about piracy, while the majority of those who wished to stick to a split license or who didn't know were more concerned about piracy and the resale and redistribution of their products. On March 26th, Collis announced that theme authors [could now sell their themes on ThemeForest with a 100% GPL license](#). Jake was [once again allowed to speak at WordCamps](#).

[^Fn-1] WP Daily has since been acquired and its content moved to Torque magazine.

43. The Problem with Post Formats

The 3.6 release cycle was a challenging one, precipitating a new approach to the development process. Two core WordPress philosophies, *design for the majority* and, again, *deadlines are not arbitrary*, were tested. The 3.6 release cycle process followed the contours of a cycle started in WordPress 3.4. There was a unified theme for the release -- this time [content editing](#). Small teams worked on key features. This approach didn't account for the fact that some features need more research and development than can be achieved within a single development cycle. And it was in the WordPress 3.6 cycle that the flaw in this approach became obvious.

Post formats was 3.6's headline feature. Introduced in WordPress 3.1., [post formats](#) is a theme feature that allows theme developers to give specific types of post a unique visual treatment. Much in the same way that Tumblr has different types of posts, like quotes or images, theme developers can style types of posts in different ways.

Post formats lacked a standard user interface. In WordPress 3.6, release leads Mark Jaquith and Aaron Campbell tackled the problem. The release cycle was made up of a number of different stages: in the scoping chat, the release lead decided on the release's key features, created teams, and assigned feature leads. Feature leads ran their teams. The release leads coordinated with the teams and made the final decision on what went into the release.

Carrying out major user interface changes in just a few months is challenging at best. WordPrss 3.5 demonstrated that to meet the deadline, the release leads needed to put in heroic coding efforts.

The 3.6 release encountered problems; [features were dropped](#) as contributors discovered they'd overcommitted themselves and

dropped out. The biggest issue, however, was around the post formats user interface, inspired by Alex King's [Post Format's Admin UI](#). Much thought and study went into the post formats UI. Which UI would offer users a logical, intuitive workflow, without making the UI or the experience needlessly complex?

The problem was that despite time spent on wireframes and development, the team ended up unimpressed. They had come up with the specifications, built to those specifications, and were unhappy with the final result. "It's like ordering something from the restaurant that sounds great," says [Aaron Campbell](#), "but as soon as it sits in front of you and you smell it, it's like, 'Ahh, definitely not what I was in the mood for.'" Even during WordPress 3.6's beta period, community members [experimented with better approaches to the problem](#).

The screenshot shows the WordPress dashboard with the 'Posts' menu item selected. On the right, the 'Add New Post' screen is displayed. At the top, there's a toolbar with icons for Standard, Image, Gallery, Link, Video, Audio, Chat, Status, Quote, and Aside. Below the toolbar is a title input field labeled 'Enter title here'. Underneath the title field is a media library button 'Add Media' and a switch between 'Visual' and 'Text' editor modes. A rich text editor toolbar is visible below the text area. To the right, a 'Publish' sidebar contains buttons for 'Save Draft', 'Preview', 'Status: Draft Edit', 'Visibility: Public Edit', 'Publish immediately Edit', 'Move to Trash', and a prominent blue 'Publish' button.

The post formats user interface

April 29th was WordPress 3.6 target release date. On April 23rd, Jen, who had by that point stepped back from her involvement in development, [said that post formats weren't ready](#). She said that the user interface was confusing, underscoring WordPress' *deadlines are not arbitrary* philosophy. The thread, in addition to highlighting the post formats UI flaws, showed that not everyone on the development team was behind the *deadlines are not arbitrary* clause. [Ryan Boren wrote](#):

The four month deadline is so fanciful as to be arbitrary. It always has been. Historically, we just can't do a major release with a marquee UI feature in that time, certainly not without heroic efforts of the 3.5 sort. So we end up facing decisions like this. Every single release we wonder if we have to punt the marquee feature. Punting often requires major surgery that can be as much work as finishing the feature. Punting is demoralizing. Four month releases are empty promises that bring us here.

In the end, [Mark and Aaron pulled the feature](#). A lot of work had to go in to [removing it from core](#); the release was heavily delayed, appearing on August 1st, 2013 -- three months after the intended release date. The team promised to turn the post formats feature into a plugin, but the plugin never materialized.

Again, a user-facing feature held up a WordPress release. Because features were tied to the development cycle, it meant that the release cycle's duration restricted and compromised UI features and/or major architectural changes. Just like tagging before it, post formats had proved too complex a problem to solve in a few short months. It isn't always easy to make interface decisions. It's harder to iterate on design than on code.

When the release deadline approaches and a feature isn't ready, the development team rushes to try to get it finished. The release is delayed by a week, and then by another week, and in some extreme cases, as was the case with WordPress 3.6, the release is delayed by months. By that point, it becomes impossible to give a firm date for when a release will happen. And the process becomes more complicated as the release lead oscillates between trying to improve a feature and deciding to pull it. Up until 3.6, there was no contingency plan built into the development process that allowed for these challenges in designing a user-facing UI.

The solution to this problem was available, and always had been available, within WordPress' infrastructure. Twice before, core user features had been pulled from plugins into core -- widgets and menus. Widgets had been built for the WordPress.com audience, turned into a plugin, and brought in to core. Menus had stumped the core development team and they solved that problem with a plugin. In both these cases, design and testing for the feature had taken place long before it got anywhere near core. And as the WordPress 3.6 cycle was dragging on, a small group of designers was working on a new WordPress feature in a plugin: it was a project called MP6. The project would be the flagship for a new approach to development that had a lasting impact on how features are developed in WordPress.

44. MP6

By 2013, the WordPress admin had seen little change since the Crazyhorse redesign in 2008. WordPress 3.6 looked the same as WordPress 2.7, apart from some minor color changes. 2013 was the year in which change happened, though this change didn't only result in a new look for WordPress. The changes influenced the way in which WordPress features were developed, introducing approach of using a plugin for feature design, feedback, iteration, and eventual merge with core.

In January 2013, [Ben Dunkle proposed new, flat icons](#). The WordPress admin looked outdated, particularly on retina displays where the icons were pixelated. Flat icons would scale properly and also enable designers to color the icons using CSS. [Andrew Ozz checked in the changes](#).

The changes kicked off huge discussions about icons, [a new Trac ticket](#), and a [long discussion on the UI P2](#). People were divided on the icons. Some liked them but felt that they didn't fit WordPress' admin design; modern icons only emphasized how dated the rest of the admin had become. Consensus didn't materialize. Mark, who was release lead at the time, decided not to put the changes in WordPress 3.6. Instead designers interested in redesigning the admin could play with the design in a plugin called [MP6](#).

Matt Miklic (MT), the designer who had put the original coat of paint on Crazyhorse, helmed MP6. Via Skype, Matt asked MT and Ben Dunkle to think about how a redesign might work, set parameters, and asked them to reimagine WordPress' admin. MT believed that Crazyhorse's UI research ought to be respected. Instead of iterating the layout and functionality, they would focus on an aesthetic refresh.

Both Matt and MT were keenly aware of the issues and challenges that had arisen in each major WordPress admin redesign. They wanted MP6 to be different.

Shuttle, for example, had been cut off from the rest of the community, designed by a cloistered group of designers trading comps while slowly losing touch with "the client." Shuttle's issues were compounded by the fact that no one person was responsible for overall vision; there was no accountability.

By contrast, the Happy Cog team had looked at WordPress with a fresh set of eyes. Their distance allowed them to treat WordPress as a piece of software, not as something that they were devoted to. They stayed in touch with their client -- Matt -- but were removed from the community's thoughts and opinions. MP6 solicited feedback from all of the people with a stake in the project. That brought its own challenges -- whose feedback was the most legitimate? What should be integrated? When was someone just complaining for the sake of it?

Crazyhorse emphasized the importance of in-depth user testing. The interface has to be both aesthetically pleasing and something users can interact with. With all of the testing that had gone into Crazyhorse, MT knew that he didn't want to carry out a structural overhaul and the extensive testing and data gathering needed to prove improvement.

The MP6 project took a different approach. Like Shuttle, a group of designers worked alongside the free software project. Instead of a mailing list, they had a Skype channel so that they could talk in private, but anyone was allowed to join in. "Even

though the group worked in private," [says MT](#), "the door into the group we were working on was open, so if anyone said they were interested they could just walk in." This provided an opportunity for people to participate who were perhaps less comfortable with WordPress' chaotic bazaar. Designers traded ideas and feedback -- without the danger of someone coming out of nowhere and tearing an idea down before it was even fully formed.

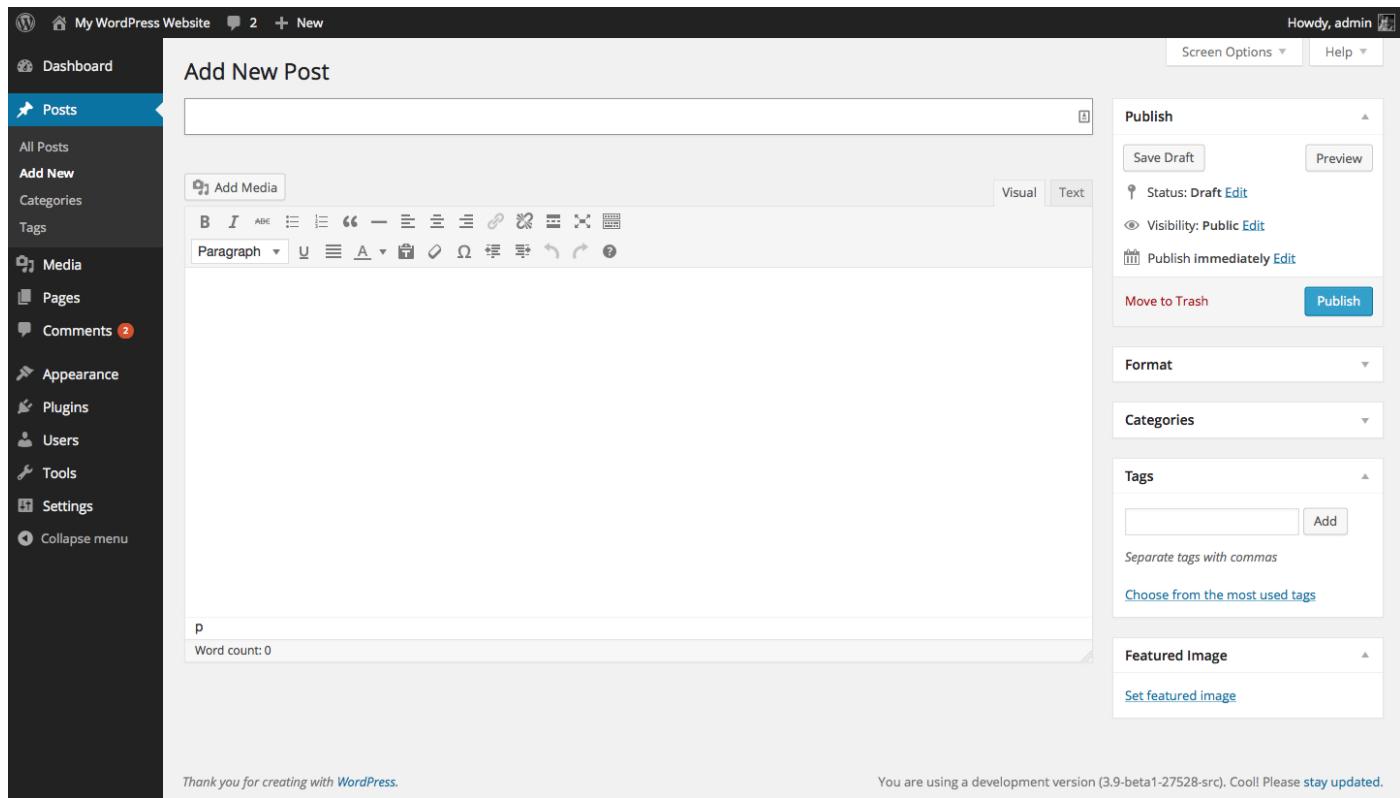
The MP6 project took advantage of WordPress' plugin architecture. The work was carried out in a plugin hosted on the WordPress plugin directory. Anyone could install the plugin and see the changes in their admin. Every week, the group shared a release and a report open to public feedback. This open process meant that community members could be involved on different levels. It also meant that the group could never steer too far off course. The core team were always aware of what was going on with MP6 and could provide feedback. More designers were involved than with Shuttle: the group grew to fifteen members. With MT as lead, they avoided the "too many cooks" problem. The designers and the community accepted that MT had the final say on the design.

The MP6 project was [announced in March 2013](#). The design process began with MT playing around with the CSS. He started out with a unified black L-shaped bar around the top and the side of the admin screen: "the idea," [he says now](#), "was that the black would feel like a background and the white would feel like the sheet of paper lying on top of it, so it would unify these disparate things." Once MT assembled the basic visual, the contributors refined the design. These changes happened iteratively. The community saw a report and a new plugin release each week, on which they could offer feedback.

Challenges arose. [Google web fonts](#) caused heated discussion. Web fonts are designed specifically for the web. They're often hosted in a font repository. A designer uses a CSS declaration to connect to the repository and the font is delivered to a website or app. MP6 uses the Open Sans font, which means that the font in WordPress' admin screens is hosted on Google's servers. Whenever a user logs into their admin the fonts are served from Google. Some don't want to connect to Google from their website; this also causes problems for people in whose countries Google is blocked. Bundling the fonts with WordPress, however, requires a huge amount of specialized work to ensure that they work across platforms, browsers, and in different languages. In the end, they decided to use Google web fonts. A plugin was created to allow users to shut them off.

Despite minor hitches, the MP6 project went smoothly. Joen Asmussen, who'd been a part of the Shuttle project eight years earlier [says](#) "I would say that MP6 did everything right that Shuttle did wrong." Over the eight years since the first attempt to redesign WordPress' admin, WordPress had matured. There was more sensitivity to what it takes to run projects within a free software environment. When things are done behind closed doors people feel disenfranchised, and yet the bazaar style model doesn't suit every single aspect of software development. It's within this tension that a balance needs to be struck, with space for ideas to flourish.

The MP6 plugin was merged with WordPress 3.8, released in December 2013, demonstrating that, while it may take a while to get there, harmonious design in a free software project is possible.



The Write screen in the WordPress 3.8 admin

All of this happened as 3.6 rumbled on. Development continued on the core product, development of the MP6 feature happened separately. It wasn't constrained by WordPress' release timeline. MT and the designers iterated quickly; users installed the plugin to test it and offer feedback. This was a new process that hadn't been possible before. In order to test out new features in the past, a person would have to be running trunk. By developing the feature as a plugin, a community member could just focus on helping with the sole plugin that they were interested in.

MP6 was proving to be a success, and in the summer of 2013, it was decided, for the first time, to develop two versions of WordPress simultaneously -- 3.7 and 3.8. WordPress 3.7 was a two-month, platform-focused, stability and security release lead by Andrew Nacin and John Cave. For 3.8 potential new features would be developed as a plugin.

Nacin wrote:

This “features as plugins” method* will allow teams to conceptualize, design, and fully develop features before landing them in core. This removes a lot of the risk of a particular feature introducing uncertainty into a release (see also 3.6, 3.5, 3.4 ...) and provides ample room for experimentation, testing, and failure. As we've seen with MP6, the autonomy given to a feature team can also allow for more rapid development. And in a way, 3.7 provides a bit of a buffer while we get this new process off the ground.

While the project prepared itself to merge the first feature-plugin with MP6 in WordPress 3.8, there was an opportunity to do something talked about for years in WordPress -- automatic updates. This was the culmination of years of work. Automatic updates had long been a goal, previously unachievable. Automatic updates needed the proper code structure to be in place on WordPress.org, as well as community trust. Community members needed to be okay with WordPress changing their site automatically.

The data collected by WordPress ever since WordPress 2.3 allows WordPress.org to create personalized automatic updates. WordPress uses the data to make sure that a site only receives an update that is compatible with its PHP version, for example, and to make sure that each WordPress site receives an update that works with their settings. In the small number of cases in which there is a failure, the user gets an error message with an email address that they can use to email the core developers who will fix their website. As of late 2014, automatic updates are for point releases only. So while major releases of WordPress are not automatic (3.7, 3.8, etc) point releases are (3.7.1, 3.8.1, for example). This means that security updates and bug fixes can

easily be pushed out to all WordPress users.

Within the space of just two short releases -- 3.7 and 3.8 -- big changes transformed the software and the development process. Automatic updates means that WordPress users are safer than ever. WordPress 3.8 saw the first release in which a feature developed as a plugin was successfully merged with core. This finally decoupled core development from feature development. So many past delays and setbacks were because a specific feature had held up a release. Now, a feature would only be slated for a release if the bulk of the work was already done. It also gave developers more scope for experimentation, and created safe spaces for contributors who were nervous of getting involved with core development. A feature-plugin group is a good introduction to the project. It allows contributors to work together in a small, controlled environment. While the MP6 admin redesign was [the first plugin integrated under this model](#), since then, feature-plugins have brought in a new widget customizer, a new theme experience, and widget functionality changes. Experiments are ongoing in image editing, front-end editing, user session management, and menus.

Endmatter

June 2010 - December 2013

WordPress 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8

Jazzers Django Reinhardt George Gershwin Sonny Stitt Grant Green Elvin Jones Count Basie Charlie Parker

WordPress 3.1 Internal linking Admin bar Post Formats Network Admin

WordPress 3.2 Refreshed dashboard Twenty Eleven Theme Distraction Free writing Retire support for PHP4, MySQL4, IE6
Faster updates

WordPress 3.3 Drag and drop uploader Pointer tips

WordPress 3.4 Theme customiser Custom Headers and backgrounds New XML-RPC API

WordPress 3.5 New media experience Twenty Twelve theme Underscore.js Backbone.js

WordPress 3.6 Twenty Thirteen theme New revisions interface Post locking Audio/Video API

WordPress 3.7 Automatic updates Better language support

WordPress 3.8 Admin redesign

Community

- community summit

Companies set up

- ManageWP
- 10up
- MailPoet
- WizzPress
- Make Do
- Devrix
- Required GmbH
- Range

- CyberChimps
- BruteProtect
- WP Valet
- Themes Kingdom
- Artesans
- WP Sessions
- Bob WP
- Code For The People
- Reaktiv Studios