



The Future of Web Apps 2006

San Francisco

Conference Notes

By

Juice

Wednesday, September 13	3
Dick Hardt - The Emerging Age of Who	3
Kevin Rose - The Digg Story	3
Tom Coates - Social Change on the Web.....	4
Tantek Celik - Best Practice With Microformats.....	5
Steve Olechowski - Ten Things You Didn't Know About RSS	6
Carl Sjogreen - How We Built Google Calendar.....	6
Mike Davidson - User-driven Content - Is it Working?.....	7
Thursday, September 14.....	8
Michael Arrington - What's Next For Web Applications.....	8
Ted Rheingold - Dogster	9
Cal Henderson - Taking Flickr to Gamma	9
Evan Williams - Funding and Selling a Startup.....	10
Ryan Carson - 14 Things I Wish I Had Known.....	11
Jeff Veen - Designing Better Web App Interfaces	13
Matt Mullenweg - The Story Behind WordPress	14
Copyright.....	16

Wednesday, September 13

Dick Hardt - The Emerging Age of Who

I was ten minutes late to the first session of the CarsonWorkshops' The Future of Web Apps conference because I took the wrong bus. By the time I got to my seat, the auditorium was full with techies illuminated by the glow of laptops and Dick Hardt was on stage talking about Identity 2.0. Dick Hardt is the founder and CEO of Sxip Identity. Dick stated that a good indicator of future behavior is past behavior and our future behavior can be gathered from the fragments of our digital identity that is held on disparate systems.

Not to long ago, Yahoo! bought Flickr and Delicious. In the case of Flickr, users are asked to login with either a Flickr or Yahoo! account. In the case of Delicious, Yahoo! chose not to confuse the matter and left the login system alone. Dick reminded the audience that in the case of Ebay, who owns Paypal and Skype, chose to keep the distinct user systems in place for each site.

One piece of advice suggested by Dick Hardt, and repeated by other speakers, was to integrate your login system. If you have more than one website, try to use the same system so that you can provide seamless access to your new service to existing customers. Typical users have way to many logins to remember as it is. I have accounts in Gmail, Flickr, Delicious, Digg, Yahoo!, Ebay, Paypal, Skype, and a more. As Dick made me understand, it would be nice if my Slashdot karma could translate well to my Ebay seller rating.

Kevin Rose - The Digg Story

Kevin Rose is the Founder and Chief Architect of Digg and according to some fuzzy math from BusinessWeek he is worth an estimated \$60 million. Kevin, like Bill Gates, is a college drop out. As a Digg user and DiggNation fan I feel that Kevin's talk was a rehash of what he always talks about. I feel that I could have learned more if he would spill the beans and inform the crowd on his relationship with Lala, of TikiBar fame. But alas, instead of speaking about Lala, Kevin went into his usual discourse of founding Digg from one idea to nine million page views.

“I want that every feature we roll out to be a tool for self expression.”

As stated earlier, Kevin's stake in Digg is estimated at some \$60 million dollars. So what was his initial investment? Kevin started Digg with a mere \$2000, a web developer working for \$10/hour, and \$99/month hosting plan. All the

components that make up Digg are standard LAMP software. Kevin recommends that developers launch on the cheap and use off the shelf platforms and components.

I thought Kevin's philosophy on design was interesting. According to Kevin there are three schools of design. One is to have the developer do it, which

shows that you aren't even trying and earns you geek creed. The second school of design is to pay 100–500 dollars but people will notice that you paid as much. And finally, the third school of design is to pay thousands of dollars to a creative agency for a design with a lot of high defying fade affects.

Regarding features implemented on Digg, Kevin stated, "I want that every feature we roll out to be a tool for self expression." His design philosophy is simple and rewarding and can be seen in the one click digg, one click burry, and now the one click 'my number one.' I think he should patent the one click digg before Amazon does.

It seems like the fabled Friendster has become the anecdote of choice for those web 2.0 social web entrepreneurs that have become successful. Regarding Friendster, Kevin said that it was "great for people coming together and date but they didn't have a task to perform." I could think of a task to perform for people hooking up at Friendster. Continuing with the lessons learned from Friendster, Kevin said that scaling is tough but if you don't scale "you can friendsterfy and lose users to another service." Kevin mentioned that a great scaling resource is Inside LiveJournal's Backend. As a final note, Kevin suggested that web 2.0 outfits should optimize their queries, hire experienced DBA, and use caching where needed

Tom Coates - Social Change on the Web

Tom Coates is a Social Software Technologist at Yahoo! Tom talked about building social software that is greater than the sum of their parts in the CarsonWorkshops The Future of Web apps Conference. According to Tom,

social software allows us to do "more together than we could do apart." This is an interesting point, because anyone can slap a beta image on some HTML, add tags and Ajax effects, and call it a social web 2.0 thingamajig. But in the world wide web, if you build it they won't necessarily come. So how do you build a web application that is greater than the sum of its parts? Well, in a nutshell Tom recommends the following: Users need to receive a value from their contribution, the contribution provides a value to peers, and the aggregation of the contributions provides value to the organization. In other words, the web application needs to provide a sense of individual, social, and organizational value.

Tom had an interesting slide where he quotes someone whose name now escapes me. The slide read something like, "Two reasons for social software (and everything else): get laid and please Jesus!" Similarly I have read research conducted on Stargate fan sites that claim that the pyramids of Egypt had little to do with the after life and a lot to do with picking up the ladies. In addition to the obvious reasons of getting laid, Tom presented a list of common motives for community involvement. According to Tom (via Peter Kollack and The Economies of Online Cooperation), the motives behind

"Two reasons for social software (and everything else): get laid and please Jesus."

community involvement include: anticipate reciprocity, reputation, sense of efficacy, and identification with a group.

Tom also had an interesting list of the motives behind Open Source developers. The motives in order of importance include: learning to code, gaining reputation, scratching an itch, contributing to the commons, and sticking it to Microsoft.

Tom's final remarks dealt with general advice on how to open up a web application to provide social value. Tom's advice includes, exposing every axis of data possible, helping user to annotate, rate, comment, and share their contributions, allowing users to associate, connect, and form relationships, and finally to provide users with place to represent and express themselves.

In the end of his talk, Tom said, people want to feel like they have an effect on the world.

Tantek Celik - Best Practice With Microformats

Tantek as the Chief Technologist at Technorati has helped shape and define new web standards. In this the fourth session of The Future of Web Apps conference Tantek described the benefits of using standardized Microformats. Microformats are small enhancements to standard HTML tags that help describe and classify the information that is being presented. According to Tantek, a microformat is a small bit of XHTML, a fast and simple way to provide an API. To illustrate this let me describe the Microformat suggested by Google to circumvent comment spammers. Google suggested that links in a blog comments apply a rel attribute with the value of nofollow. Here is an example:

```
<A href="some url" rel="nofollow">some link</A>
```

Another example of the use of the rel attribute is how a designer can indicate that the link indicates the site's license.

```
<A href="some url" rel="license">some link</A>
```

Tantek also describe how easy it is to transform HTML to a vCard. I have worked on a XML to vCard component for work and I can attest the difficulty in understanding RFC 2426, the vCard Spec. The fastest path to supporting .ics in your application is not by reading the RFC 2426 but by using hCard. If I would have known then what I know now I would have bought Google stock and use hCard, a HTML to vCard converter. hCard is a microformat that can define a contact using nothing more than standard HTML div and span tags with some additional attributes that describe the data. There are several other microformats such as hCalendar, hReview, VoteLinks, XFN, and rel-tag that will be of interest to most developers.

Regarding API design, Tantek recommended to build something simple, which publishers can easily implement, and therefore can spread in the wild.

Steve Olechowski - Ten Things You Didn't Know About RSS

Steve Olechowski is the cofounder and COO of FeedBurner. During his CarsonWorkshops' The Future of Web Apps session he gave some insight into RSS feeds. I didn't record all the ten things about RSS that I didn't know, because what you don't know won't hurt you and because they seemed unimportant for someone not working at FeedBurner.

One thing that I have been playing with is how much text should I display in my RSS feeds. I used to display the whole enchilada but then limited to the summary for each post. According to Steve, the more text you display on a RSS feed the more traffic will click through to your site. Steve admits this seems counterintuitive. In addition to more traffic, the more text you display on your RSS feed the more people will subscribe to your feed, this is intuitive.

Another thing I didn't know but that Steve shed light on is that podcasts are more evenly distributed across topics, sectors, and categories than text blog feeds. I also didn't know that 7% of RSS click throughs are made by bots. The last fact from Steve's talk that I took notice of was that RSS feeds are that they are being read on mobile phones.

Carl Sjogreen - How We Built Google Calendar

Carl is the Product Manager of Google Calendar. During his presentation at The Future of Web Apps, Carl shed some light into the development process in place at Google on a typical project and his experience building a great web 2.0 application. Carl said that Google Calendar started as a 'classic' Google product team, that is 1 product manager and 3 engineers. The original idea of a Google calendar system was conceived from customer feedback and internal need. In designing Google Calendar, Carl said, "being smart isn't always best." Carl explained how they over-engineered some usability features in the early Google Calendar version that they had to dumb down or simplify for the general public.

Carl also provided six key insights for your next product or company that I will now copy here verbatim.

The first insight is that easy is the most important feature. Think Google Calendar QuickAdd, Google Maps single text field. Google Maps has a single field as opposed to a whole address, city, state, zip code form.

The second key is to know your real competition. You always have a competitor no matter how innovative you are. The real competition often is the pen and paper. Do well what paper can't, collaborate beyond the same room.

The third point presented by Carl is that visual design matters.

The fourth intuition, which seems counterintuitive at first, is to build products for people who don't want to use them. Carl reminded the audience "not everyone who can benefit from your service actually wants to use it." Get your product in front of the applications people use every day "and then make it

painless for people to start using your product without fully switching into a new way of doing things.”

The fifth insight provided by Carl was to time your launch properly. Carl made it clear that, “you better not launch until you are really ready. ... Launch early and often, but not too early”

Carl’s last key insight is ‘driving usage.’ By driving usage he meant to provide enough touch points into your web application via an open API. Tantek also spoke about providing a simple to use API. By driving usage he meant to let your application to spread to as many site as possible without having user come to your site, think social beyond tags.

Mike Davidson - User-driven Content - Is it Working?

Mike is the founder and CEO of Newsvine. Mike’s presentation at CarsonWorkshops’ The Future of Web Apps conference dealt with user generated content. User generated content is not a new concept; it is just a new buzzword. Amazon is driven by user generated content, user can review reviewers. Amazon, Ebay, Epionions have been using user genrated content for years, since the web 1.0 era. The current examples of user genereated web 2.0 sites are MySpace and Digg. During this conference I head MySpace be used as an example every hour, I head people say things like “I want to be the MySpace of gardening.” I want to be the MySpace of fund of hedge fund software.

Mike said, “Delicious is a great way to extend your memory.” I use it as a reading list, I bookmark what I haven’t read but want to. YouTube is another often mentioned web 2.0 user generated content web application. Mike also mentioned “the interesting thing about Second Life, is that fake products are being sold for real money in a fake environment. That is a market I am interested in being a part of.”

Scaling was the unwritten theme of the conference. Mike advised that social services need to scale. He also recommended to not develop a social network for the sake of a building a social network. People want to don’t want to join a social network, they want to join something meaningful that also happens to be a social network. Flickr is not just a social site; it is primarily a great way for user to organize their photos. Some advice that I need to follow is to always leave comments turn on.

Mike mentioned the Dunbar’s number, which states that the ideal size of a group is roughly 150 people. According to Wikipedia, Dunbar’s number is the “limit to the number of individuals with whom any one person can maintain stable relationships.” It is at around this number that groups function best, where the users can get to know each other and hence care for each other a some level.

Thursday, September 14

Michael Arrington - What's Next For Web Applications

At the Carson Workshops' Future of Web Applications conference in San Francisco, Michael Arrington of TechCrunch presented on What's Next For Web Applications: creating tomorrow's Flickr. According to Michael, predicting the future of web application is difficult. He stated, "Honestly, the people that are most likely to succeed are probably not in this room, and if they were they wouldn't listen to me." After he said those words I didn't feel like I wanted to be there and listen to him. After making that bold statement he continued on to give some advice on choosing the right platform for you next web application. Michael mention that a top contender for the next killer web application is Adobe's Apollo. Regarding Apollo Michael stated, "A whole new class of companies can be built around that platform."

"Raising too much money and spending it is a bad thing."

Michael Arrington is not really a technologist; he is more like a web two point duh commentator. And his post game analysis of the big web 2.0 winners and losers are as followings. The big winners (because they were acquired)

are writely, skype, newroo, flickr, userplan, myspqce, blogger, bloglines, and grouper. The most likely to win (or become acquired) are zoho, netvibes, digg, facebook, popsugar, stumbleupon, and plentyoffish. The web 2.0 companies to watch out for are jobster, riya, zillow, flock, sharpcast, roketboom, wordpress, second life, and odesk. And according to Arrington, the god awful "what were they thinking" is made up of squidoo, inform, gather, pubsub, browzar, and jigsaw.

But Arrington went beyond than just naming names; he listed the share attributes of the winners and losers. The companies listed as winners share passion for their work, leaders as founders, great dynamics, never raised too much money, did something extraordinary. In regards to the extraordinary, Arrington recommended companies read The Purple Cow by Seth Godin. But remember that once all the cows are purple, people won't be interested any longer. In regard to fund raising, Arrington said that, "raising too much money and spending it is a bad thing." Another Arrington tip for success is to "hire slow, fire fast!"

The shared attributes amongst the losers are poor founder/team choices, lifestyle/ego entrepreneurs, raised too much money, over business plan, and forgetting to scale (ala Friendster).

At the end of his talk Arrington again discussed technology choices. On the server end he recommended PHP, just because it is the de facto language for web programming. Arrington also suggested Ruby on Rails as an upstart. On the client side development he recommends .NET/ActiveX, Ajax, Flash, XUL/XAML, and Apollo.

Arrington also talked about market saturation. Arrington recommended that new entrepreneurs stay away from the following over saturated markets: social networking, social bookmarks, video, photos, portal/homepages, and feed readers. As Arrington said, “you have to be pretty special to be successful in these market.” But on the flip side, he recommends developers get into platforms (widgets, systems, backend), desktop apps (via Apollo), office efficiency, cloud storage, identify, developer tools, market destruction, and ENTERPRISE (push web2.0 apps/wikis/social apps/ideas/concepts to the enterprise).

In the end Arrington said, “The best entrepreneurs avoid this type of advice. Invent a new market.” If you have an idea, just do it!

Ted Rheingold - Dogster

Ted Rheingold started Dogster as a joke and now he is laughing all the way to the bank. At the Future of Web Applications conference in San Francisco Rheingold talked about The State, Future and Business of Passion-Centric Communities. According to Rheingold, a passion-centric community is dedicated to a single particular interest with a single mind, single focus. Passion-centric sites are not something new; sites of this nature have existed via newsgroups, forums going back to the web 1.0. And now passion-centric sites are ajaxified and web two point out. For example Dogster is the MySpace, social network, for dogs, where dogs have journals and their top 8 doggie friends. And just like MySpace, Dogster has in my opinion a very web 1.0 look to it but with some web 2.0 features such as tags and blogs.

Rheingold made a few design suggestions for passion-centric sites like Dogster. He recommended plenty of opt-in/opt-out settings, use of color other than white, obvious sections and labels, plenty of pictures. He admitted that Dogster may look like a design mess, but he stated that people just jump into the content that they are interested in. Rheingold suggested you let your users feel like the site is theirs, because in the end it is their data, their groups, and their social contacts that drive your potential site. Rheingold said that “commenting is community,” sites like MySpace are like “digital doritos.” Rheingold reminded the audience, “You are part of a community.”

Cal Henderson - Taking Flickr to Gamma

Cal Henderson is the chief software architect of Flickr and he spoke at the Future of Web Applications conference about Taking Flickr to Gamma. Flickr started as a tool for the development of a massively multiplayer online game by Ludicorp. As it turned out the game was shelved and photo-sharing tool became Flickr. Ludicorp as a game company was a complete failure, but as a photo-sharing platform it was a genius. From Cal’s experience with both failure and success he stated that they started with the things that they already knew and the things that they needed to know. In the case of Ludicorp, the company that developed Flickr, Cal stated that the overlap between what they knew and what they needed to know was just a small bit of HTML.

Cal also advises upstart startups to plan for maintenance. Cal said that no matter how large a site and how many resources available, something will go down. If at all possible provide advance notice of scheduled disturbance. Another suggestion provided by Cal was to try to upgrade, disable, fix a component at a time instead of the whole site. And throughout the process have a clear escalation path. What happens when the system breaks?

Another great piece of advice that Cal gave the audience is to create dashboards to visualize a lot of the data you maintain for your users. According to Cal, if you can't use the visualizations, at least they are kewl. Speaking of kewl, he also said that APIs are kewl but be careful because a bad API can lead to bad programming which can lead to unintentional abuse. With a bad API you might have developers hitting you site several times a second. "What? I hit your site 20 times per second? I thought it was meant to hit it once an hour... Ahh, one decimal place." But with all of this said, APIs force a clean interface and allow for easy regression testing.

Cal also stressed the importance of having clean URLs. Clean URLs are understandable by humans that they can see where they are and can modify the location bar in the browser to where they want to be! Before you go cleaning up your URLs here is the rule of thumb for URLs: Never Break Them. Cal also recommended something that goes against the practice of many Ruby on Rails applications; he suggested that you do not expose the auto-incremented database ids in your URL.

Cal echoed something that Michael Arrington of TechCrunch said earlier in the conference. Cal said that hiring people is tough since good engineers already have jobs. Because his talk was running long, the last five minutes of his talk was just a photostream of advice: Documentation save my life. Release early, release often. Under construction sign has been replaced with a perpetual beta. In the Internet nothing is ever finished, unless it is finished. Make small increments of visible progress. Have developers own the process not the feature; they will feel like they own the application not just a component. No developer is an island. Allow for a one-touch deploy, with rollbacks (ala Capistrano). Automate everything. Be pragmatic and make the system work, if it works then good, if it is also maintainable, even better. And lastly he said, "Ideologically beauty is not a priority."

Here are some visualization tools recommended by Cal: Cacti, Ganglia, and VisualComplexity.

Evan Williams - Funding and Selling a Startup

Evan Williams is the entrepreneur behind Blogger and Odeo. Evan sold Blogger to The Borg of Web 2.0 startups, Google. "We are the Googleplex. You will be assimilated. Resistance is futile." Evan was at the Future of Web Apps conference in San Francisco to talk about his experience with web entrepreneurship. Evan's talk was titled Selling and Funding: Pros and Cons of Bringing in a Third Party.

During his talk, Evan noted several rules for a web startup. These rules included: be user-centric, be self-centered, be greedy, be tiny, and be balanced. Having given out a few rules for a startup, Evan spent most of his time talking about his five best Odeo screw-ups.

Evan's first mistake when building Odeo was to try to build too much. He talked that mentioned that they had written large number of verbose specs and in the end they were not the first in the market.

Evan's second screw up was building a service for people not like themselves. In an essence, Evan stated that they did not 'eat their own dog food.' In general, it is a good idea to use the freaking service you are building!

Evan quoted Markus Frind of Plenty of Fish who said, "The enemy was thinking."

The third major screw up that Evan learned from his second entrepreneur endeavor was not adjusting fast enough. All these web 2.0 applications are all trying to keep up with the Joneses. Someone integrate with Google Maps, your web application needs to integrate with Google Maps. They start tagging your service start tagging. Evan reminded the audience to try to be a purple cow, not a mee too application.

Another screw up mentioned by Evan of Odeo was raising too much money, too early. Raising too much money almost seems counterintuitive. Raising too much money was not heard in the web 1.0 era when champagne bottle was served for breakfast and companies had a \$10 million burn rate a month. Evan said to "think of money as fuel." If you have the fuel before the engine, you start thinking about the fuel. He said, "What do you do with soo much fuel, you drop some on the floor and light it on fire." Kevin Rose and Michael Arrington echoed Evan's thought on raising cash, raising money is about timing.

The last screw up mentioned by Evan of his experience from building Odeo was not listening to his gut, his techie intuition, in hiring, raising moola, and building the 'right' product. Evan quoted Markus Frind of Plenty of Fish who said, "The enemy was thinking." Markus was talking about coder's block, similar to writer's block. Evan said that the best response is to not talk, just type. Don't Talk. Just Code It!

Ryan Carson - 14 Things I Wish I Had Known

Ryan Carson is the Future of Web Applications conference organizer and he spoke about what he wish he had known before starting his online ventures, Hey Amigo, Drop Send, and Think Vitamin. Ryan started his session by saying, "We built three web applications, the first one we don't talk about." Here are the 14 things Ryan wished he had known before he started his applications. Hopefully they will help you, if you didn't already now them. Before I begin I should not that his advice sounds more appropriate from small teams, not multinational conglomerates.

1. Ryan recommends working with people in the same time zone. Ryan said that if you don't work with someone in your time zone you will spend time on the phone when you should be sleeping. Ryan lives in the UK so this might be good advice for him. For people that live in the continental United States or Canada working with people anywhere from Eastern to Pacific Time zone should be fine.
2. Use one user database. Ryan mentions this because his outfit developed several online services, each with its own user database.
3. The third piece of advice sounds like the second, Ryan recommends you use one e-commerce system. In general, when using third party software or services find the right partner and stick with them. Using, learning, and integrating multiple e-commerce systems is not the right use of your time.
4. Ryan disagreed with Kevin Rose on having developers also hack together the UI. Ryan's background is as a web designer so obviously he would recommend hiring a pro front end XHTML/CSS developer.
5. As a web application developer you obsess with features and functionality. Ryan thinks that you should obsess about your website's copy. Since web applications don't come in a nice shrink wrap or with anything physical that can give users a sense of satisfaction, your content, design, and text should give them that warm, fresh, and trust worthy feeling. People skim at 60 mph, design your site for that and catch your users attention.
6. Work with top-notch hardware partners. When working with partners have a list of support resources before you need it. Echoing Cal Henderson, Ryan suggests planning for maintenance. When it comes to hardware don't be special, work with off the shelf components for which there is a lot of development and support resources.
7. Not really a technology related advice but Ryan suggested that it is always a good idea to not cut corners. As we all know, trying to save ten or fifteen minutes with a hack can eventually cost days of man-hours.
8. Again echoing Cal Henderson of Flickr, Ryan suggests that once you go live with a web application you measure performance, activity, and usage. Ryan recommends you measure what feature users are using, which features they are not. If you don't know why users aren't active, you can't fix it.
9. According to Ryan, when building a web application, you are not done when you launch. If you are contracting out the work to offsite/offshore developers be clear to clarify with them what happens are the launch. Make sure you understand how much a new feature or update is going to cost you once you launch.
10. Even before the web application is code complete you need to work out the details such as FAQ and help sections, spell check, testing the e-commerce system, etc.
11. The eleventh point made by Ryan was a few quick tips. Ryan recommends that you make easily available logos, screenshots, and contact details for the

press. Use a monthly CSV file for invoices. Add an About Us page so that user can get a hold of you (they want to know that real people are involved, put phone number, contact, photo, build trust). Make contact easy.

12. Add a ton of stuff to your FAQ/Support.

13. Be nice to nasty customers. Ryan said that you could disarm and convert angry user to paying customer by just “I’m sorry…”

14. For the last piece of advice he wished he had known before building a web application, Ryan quoted other speakers. Ryan said that the first, and maybe even the second, version of an application are always throwaways. Marketing and promotion can be a full time job; if you build it you need to promote it before they will come. User’s can be trusted; you don’t have to validate an email to have users start using your system.

Jeff Veen - Designing Better Web App Interfaces

Jeff Veen of Google is the Project Lead of MeasureMap and he spoke at the Future of Web Applications regarding designing a better web user interface. Jeff start off his talk by saying that buzzwords are often used as a proxy for intelligent thought. He also reminded the attendees that booms and bust are cyclical. What I like most of Jeff’s talk was when he said that web applications need to go beyond visual design and move into interaction design.

Another key theme from Jeff’s conversation was about building trust with your users. Jeff recommends that the UI be developed in such a way that it builds a sense of trust with users. Another way to build trust with users is to allow the threat your users as peers and share the users own data in graphs and by providing power tools to passionate users. You should think of new ways to expose data to your users. A social network is built on social trust, don’t be antisocial with your users!

Another key observation made by Jeff was the all the speakers were Anglo-Saxon males! This is an important observation because the trend online is that English is no longer the language of the internet.

Jeff also talked about the four design principles of design, which include discoverability, recoverability, context, and feedback. Discoverability as a design principle means that applications should suggest features but not distract users from their goals. It is always a good way to provide users a way to recover their actions; actions should come without a cost. A web application should display context at all times, such as time, place, user name, position, rank, and any other pertinent information. And lastly, by feedback Jeff is referring to how the system responds to users’ actions. Ajax is a great way to provide feedback to users. Jeff quoted Bruce Sterling in saying that Ajax is like “roller skates for the web.”

Matt Mullenweg - The Story Behind WordPress

The last speaker of the Future of Web Applications conference was Matt Mullenweg of WordPress fame. Matt's talk was a nice way to end two days of great presentations. According to Matt, in the future of web applications everybody will have their 15 pixels of fame. Matt also said that, "The luckiest thing I did for WordPress is allow for plugins." In Matt's view some open source software projects try to resolve conflicts with more options. In his view, a better way to solve conflicts is by enabling plugins. In addition to conflict resolution, he believes that plugins limits forks. A word of advice regarding plugins is to never break the API they are built upon. Breaking APIs is worse than never having any at all!

"The luckiest thing I did for WordPress is allow for plugins."

"If one thing that I know about the future of web apps is, you will be spammed." Matt believes that spammers are like cockroaches that will be here long after us. Social software are things that gets spam. To combat spammers, the

scourge of the internet, Matt has developed Akismet. Akismet is a learning web service spam filter for blogs. From his experience with WordPress.com and Akismet Matt, like other conference speakers, believe that you need to plan for success. According to Matt, "WordPress.com has been lessons in scaling. ... All hosting sucks." Since hosting sucks you can buy the cheapest hardware available, because you can easily miss utilize expensive hardware with a bad hosting service. The best you can do is hope that your web hosting doesn't suck all the time.

According to Matt, in Computer Science there are only 3 numbers, 0, 1, and n+1. For you to scale well you need to solve for n+1. Another lesson shared to the audience was that painkillers make mover money than vitamins. People appreciate a service more that solves a pain point, like blog spam. Be a pain killer, not a vitamin!

According to Matt, four points are key for the future of web applications...

- global – 118n, m17n, l10n, multiple language support.
- personal – care about what the individual is interested in.
- useful – useful to the user.
- humble – frame the service in a sense of 'me', from the user's point of view.

Just to wrap up the my coverage of the Future of Web Applications conference I am going to jot down some quotes from Matt...

- If you can't be the most passionate user of your product, it is going to be hard.
- Overnight success, contrary to popular belief take about 2 to 3 years.
- Don't feel like the market is saturated, or that everything is done.

- Create value for an x number of user. the x number doesn't have to be that large anymore.
- Build for yourself first.
- Working for someone else is dead.
- This does not mean working alone.

Copyright

The Future of Web Apps San Francisco 2006 Conference Notes is Copyright (c) 2006, Juix.com. It is licensed under the terms of the Creative Commons Attribution-ShareAlike 2.5 agreement:
<http://creativecommons.org/licenses/by-sa/2.5/>