# The developer evangelism handbook

This handbook will get you on the way to be a great developer evangelist for any product or company. Of course your approach needs tweaking for different markets and audiences - and in accordance with your own personality - but the main principles are the same for everybody and anywhere in the world.

Developer evangelism is a totally new field of work and the first hurdle you will encounter is people asking what a developer evangelist is and why any company would need a role like that.

# Defining developer evangelism

A developer evangelist is a spokesperson, mediator and translator between a company and its technical staff. Every day millions of dollars are wasted in companies because non-tech people and tech people either don't communicate at all or completely miss each other's points.

Developers are what makes IT work. Yes, ideas are what start a great product and IA/Design (or call it UX if you want) makes it work for users but to make it work technically you need developers. Sadly enough developers don't get much credit for their work and are generally considered "deliverers" rather than "thinkers", which is simply not true. What this also means is that telling something to developers as a company or getting them excited is quite a task.

**Tip:** If you are a clever company you also open your products to third party developers and release interfaces to the world. This could be as simple as an RSS feed, APIs, SDKs or being a man and releasing the whole thing as open source. The benefits are that millions of developers out in the world can find issues with or uses of your products that you never thought of. Innovation can happen anywhere - not only in a meeting room inside your company.

Both the world and your company is full of dedicated, highly skilled technical people that are ready to solve anything technical that needs solving. You can get them as excited as a 10 year old on a sugar rush who gets a puppy to play with - for us geeks the puppy is technology and the sugar caffeine.

You can make geeks find solutions for almost anything - if you speak their language. If you don't then they will most likely appear as weird, non- communicative and in general not as excited about working for the company as - for example - the marketing department is.

The trick is to understand that to be a developer - especially a web developer - you need to have a

certain way of seeing the world. And this way of seeing the world makes you suspect things to fall in any which way. If your message means less work for the developers out there it is a great start to come from. If it means extra work on top of what is already on their plate (and developers always get maxed out) then you'll be out of luck.

## Start with the right mindset

The main thing never to forget as a developer evangelist is the technical part. It is very easy to get into the habit of just writing one presentation after another and re-use materials but this way you will not have much impact.

If something new comes out of your company that should get out to developers take it and access it like an outside developer would. Develop something with it, then document what you have developed and then start writing how you build the thing - and voila - you've got half an article or presentation already finished.

As a developer evangelist it is very important that you used to be a developer. The more projects the better. For example working in an agency is different to working for local government or a large multi-national company. Your job will be to make your company's technical offerings attractive and interesting to a large variety of developers, and you really only can do that when you know their pain.

Of course it is important to be a technical expert, but there are so many more little annoying parts to delivery inside a company that you should be aware of. If you don't have the experience in having to deal with them (and the frustrations they bring) you will have a much harder time giving developers the ammunition they need to sell your services to their boss.

Developer evangelist is a role that is a change for developers, not for people coming from HR, PR or marketing. Your main job is still to code - but this time examples, training materials and explanatory demos rather than live products.

## Find your role and play to your strengths

Not everybody can and should be an all-around evangelist. It is enough if you find your place in the whole spectrum of evangelism. Think about what you love to do the most and then start creating something. The most common niches of the whole job to go into are:

- Writing code tutorials
- Blogging

- Public speaking
- Training
- Social web coverage

Check the rest of the handbook and see what resonates best with you. Then start evangelising. You have nothing to lose and will most probably be very surprised how easy things become if you concentrate on one job at a time.

# Remove the brand

One very crucial part to becoming a successful developer evangelist is to remove the brand from your thinking.

Yes, you work for a certain company that builds a lot of products, some of them cool - some of them terrible. The point of developer evangelism is not to get people excited about the brand or the company behind it though.

Instead it is about the products the company releases and even more specifically about getting developers excited about playing with them.

This only works when you are excited about the product. If the department that builds the product fails to get you interested in the product, don't talk about it - instead, work with the department to find the thing that makes it worth while for you. Anything you evangelise should answer one simple question: **What is in it for me?**

The products you evangelise need to get you excited and you need to know where they are going and who to ask for detailed information about them. If the product has no internal stakeholder or team that is a danger sign.

People working on products are actually not the right people to advocate them - they are far too close to the subject matter to find obvious flaws in the documentation or are fine with overly complex ways of invoking a certain functionality as they are used to them.

Your job is to offer them a way to translate this into easier understandable documentation and examples and challenge their current state of affairs. You do however need them to do a good job so be careful not to burn any bridges by being too critical. People spent a lot of time building what you want to tell the world about and are protective about it.

The real power of removing your brand goggles is that you will be able to work with rather than against the competition.

# Work with the competition

As a developer evangelist you have to keep your independence. Yes, you are a specialist in the

technologies of your company, but if you are oblivious to the world outside the company and preach a mono-culture you will not get far.

**Fact:** Your independence and your integrity is your main weapon. If you lost it you are not effective any longer. People should get excited about what you do because they trust your judgment - not because you work for a certain company.

Developers are terribly loyal to solutions and technologies once they are happy with using them. It is very rare to find a developer who is OK with jumping from PHP to Java to Ruby to C# to Python and in between Windows, Unix and Mac.

On the contrary - we do spend a large part of our online time bickering at each other that our favourite language is so much more powerful than anything else. Quite pointless, really, but it shows that developers have passion and passion is a good thing.

The same happens with companies. You get Microsoft fanboys, Google enthusiasts, Yahoo fans, Apple disciples, Adobe followers and they all hardly ever mix without arguing with each other about why their favourite company is the best.

All this means that as soon as you start talking about your brand exclusively the cards are stacked against you - either you'll preach to a choir or get shot down in flames.

You work around that in a few ways:

- **Remain an independent voice** - talk about everything that gets you excited regardless of where it came from.
- **Become a specialist in a certain underlying technology** or methodologies that all these companies and languages rely on.
- **Keep your finger on the pulse** - have a good collection of RSS feeds to go through daily and tell the world about things you found and tried out.

The really fun thing is that every company out there wants to do what you want to achieve - make developers happy using their products. Therefore it is very important that you keep an eye on and in contact with the competition as much as it is important to be aware of what your own company is up to.

**Example:** A really interesting moment happened last year at the Ajax Experience in Boston. I was part of a panel of JavaScript library developers each representing a certain library. The audience was full of fans of all the different libraries and eager to see a big fight on stage. The first thing we did though was tell the audience that there is no point in comparing and bickering as all libraries want to do the same thing - make browsers behave and JavaScript development more predictable. We even pointed out what we appreciate the most about the other libraries. In the end the audience went home with a much more detailed picture about what each library does better than the other - not from the people building it but from their competition. Everybody won.

## Show respect to the competition

## Show respect to the competition

You can't be a professional evangelist and bad-mouth the competition at the same time. We all are professionals and work on projects to make the web a better place. Different companies have different approaches and different internal red tape to battle. Pointing out weaknesses of the competition is a cheap shot.

You should also not forget that we are seemingly working in an industry that moves and shakes the world but a lot of this is inflation. The amount of people on the speaking, training and advocating market is very small indeed and whoever you cross will come back to you very soon indeed. Better to work together than to annoy each other.

Showing respect and interest also means that - once you realise that you can trust another - you start sharing ideas, resources, conference opportunities and even give each other sneak previews of things to come. And that gives everybody an advantage and makes our jobs easier.

## Acknowledge when the competition is better

This is a hard pill to swallow for a lot of people - especially for marketing departments - but bear with me. **If your competition has a better product than yours and people ask you which one is better do admit that this is the case**. Whilst this sounds like admitting defeat in reality it shows a few things you can use to your advantage:

- You come across as someone who appreciates good technology.
- You come across as someone who does not fear competition but welcomes it.
- Your competition feels that they've done an amazing job and will look closer at your products in return.
- You will learn what people love about the competitor's product and can feed that back to your own product team.

Another thing to remember is that the product might be better but for a different audience. What gets developers excited does not necessarily mean mom and dad users can deal with it.

**Example:** Many people get confused when they see that as somebody who works for Yahoo I have a gmail account as my main email. The reasons are that I had it two years before I started at Yahoo and for me and the amount and kind of emails I get daily it is the right interface. Yahoo Mail, on the other hand is dead easy and feels very natural for people who use Outlook and get more office style emails.

## Know about the competition

This is a classic marketing, advertising or even development step: before you build or promote something look around and do a competitive analysis.

In the case of developer evangelism you need to be up to speed with what your peers are up to as you will constantly get questions about it. “How does this compare to X, the new product by Y?” is a very common first question.

If you can answer that, your tech integrity gets quite a boost and - let’s face it - it is fun to poke at the things our peers produce.

## Build mashups using competitive products

Using the web services of your competitors is a great way to check several things:

- What do they do well, that could be done better in your company’s services?
- Where do you get stuck? This is something to avoid in your own services.
- How does it compare to your own services? What are the differences? Remember, these are the questions that people will ask you, and you learn best by doing.
- How can these services be mixed with your own and how do they compliment each other?

Building something with a brand new API also helps your technological integrity and you can feed back problems you found to your peers in the other company. I’ve done that with several Google and Microsoft products and actually managed to get to know the right people when I have questions that way.

If you find that a new API by another company compliments one of yours nicely, build a mashup and show that to the world - this will be beneficial for both APIs and people see how easy it is to mix services from various companies.

# Remove the brand

One very crucial part to becoming a successful developer evangelist is to remove the brand from your thinking.

Yes, you work for a certain company that builds a lot of products, some of them cool - some of them terrible. The point of developer evangelism is not to get people excited about the brand or the company behind it though.

Instead it is about the products the company releases and even more specifically about getting developers excited about playing with them.

This only works when you are excited about the product. If the department that builds the product fails to get you interested in the product, don't talk about it - instead, work with the department to find the thing that makes it worth while for you. Anything you evangelise should answer one simple question: **What is in it for me?**

The products you evangelise need to get you excited and you need to know where they are going and who to ask for detailed information about them. If the product has no internal stakeholder or team that is a danger sign.

People working on products are actually not the right people to advocate them - they are far too close to the subject matter to find obvious flaws in the documentation or are fine with overly complex ways of invoking a certain functionality as they are used to them.

Your job is to offer them a way to translate this into easier understandable documentation and examples and challenge their current state of affairs. You do however need them to do a good job so be careful not to burn any bridges by being too critical. People spent a lot of time building what you want to tell the world about and are protective about it.

The real power of removing your brand goggles is that you will be able to work with rather than against the competition.

# Work with the competition

As a developer evangelist you have to keep your independence. Yes, you are a specialist in the

technologies of your company, but if you are oblivious to the world outside the company and preach a mono-culture you will not get far.

**Fact:** Your independence and your integrity is your main weapon. If you lost it you are not effective any longer. People should get excited about what you do because they trust your judgment - not because you work for a certain company.

Developers are terribly loyal to solutions and technologies once they are happy with using them. It is very rare to find a developer who is OK with jumping from PHP to Java to Ruby to C# to Python and in between Windows, Unix and Mac.

On the contrary - we do spend a large part of our online time bickering at each other that our favourite language is so much more powerful than anything else. Quite pointless, really, but it shows that developers have passion and passion is a good thing.

The same happens with companies. You get Microsoft fanboys, Google enthusiasts, Yahoo fans, Apple disciples, Adobe followers and they all hardly ever mix without arguing with each other about why their favourite company is the best.

All this means that as soon as you start talking about your brand exclusively the cards are stacked against you - either you'll preach to a choir or get shot down in flames.

You work around that in a few ways:

- **Remain an independent voice** - talk about everything that gets you excited regardless of where it came from.
- **Become a specialist in a certain underlying technology** or methodologies that all these companies and languages rely on.
- **Keep your finger on the pulse** - have a good collection of RSS feeds to go through daily and tell the world about things you found and tried out.

The really fun thing is that every company out there wants to do what you want to achieve - make developers happy using their products. Therefore it is very important that you keep an eye on and in contact with the competition as much as it is important to be aware of what your own company is up to.

**Example:** A really interesting moment happened last year at the Ajax Experience in Boston. I was part of a panel of JavaScript library developers each representing a certain library. The audience was full of fans of all the different libraries and eager to see a big fight on stage. The first thing we did though was tell the audience that there is no point in comparing and bickering as all libraries want to do the same thing - make browsers behave and JavaScript development more predictable. We even pointed out what we appreciate the most about the other libraries. In the end the audience went home with a much more detailed picture about what each library does better than the other - not from the people building it but from their competition. Everybody won.

## Show respect to the competition

## Show respect to the competition

You can't be a professional evangelist and bad-mouth the competition at the same time. We all are professionals and work on projects to make the web a better place. Different companies have different approaches and different internal red tape to battle. Pointing out weaknesses of the competition is a cheap shot.

You should also not forget that we are seemingly working in an industry that moves and shakes the world but a lot of this is inflation. The amount of people on the speaking, training and advocating market is very small indeed and whoever you cross will come back to you very soon indeed. Better to work together than to annoy each other.

Showing respect and interest also means that - once you realise that you can trust another - you start sharing ideas, resources, conference opportunities and even give each other sneak previews of things to come. And that gives everybody an advantage and makes our jobs easier.

## Acknowledge when the competition is better

This is a hard pill to swallow for a lot of people - especially for marketing departments - but bear with me. **If your competition has a better product than yours and people ask you which one is better do admit that this is the case**. Whilst this sounds like admitting defeat in reality it shows a few things you can use to your advantage:

- You come across as someone who appreciates good technology.
- You come across as someone who does not fear competition but welcomes it.
- Your competition feels that they've done an amazing job and will look closer at your products in return.
- You will learn what people love about the competitor's product and can feed that back to your own product team.

Another thing to remember is that the product might be better but for a different audience. What gets developers excited does not necessarily mean mom and dad users can deal with it.

**Example:** Many people get confused when they see that as somebody who works for Yahoo I have a gmail account as my main email. The reasons are that I had it two years before I started at Yahoo and for me and the amount and kind of emails I get daily it is the right interface. Yahoo Mail, on the other hand is dead easy and feels very natural for people who use Outlook and get more office style emails.

## Know about the competition

This is a classic marketing, advertising or even development step: before you build or promote something look around and do a competitive analysis.

In the case of developer evangelism you need to be up to speed with what your peers are up to as you will constantly get questions about it. “How does this compare to X, the new product by Y?” is a very common first question.

If you can answer that, your tech integrity gets quite a boost and - let’s face it - it is fun to poke at the things our peers produce.

## Build mashups using competitive products

Using the web services of your competitors is a great way to check several things:

- What do they do well, that could be done better in your company’s services?
- Where do you get stuck? This is something to avoid in your own services.
- How does it compare to your own services? What are the differences? Remember, these are the questions that people will ask you, and you learn best by doing.
- How can these services be mixed with your own and how do they compliment each other?

Building something with a brand new API also helps your technological integrity and you can feed back problems you found to your peers in the other company. I’ve done that with several Google and Microsoft products and actually managed to get to know the right people when I have questions that way.

If you find that a new API by another company compliments one of yours nicely, build a mashup and show that to the world - this will be beneficial for both APIs and people see how easy it is to mix services from various companies.

# Work with your own company

As a developer evangelist you will find that a large part of your work is dealing with internal developers and unpredictable changes to your company.

This can actually be a much harder job than dealing with outside developers. The reason is that people on the inside do not only see the outcome of the products and disagree with that but are also part of the journey towards these - and that journey can be painful, frustrating and sometimes just plain confusing.

Your job as a developer evangelist is to listen to developers, understand their problems and communicate with management to try to sort the issues out. You also need to ensure that you remind people of facts instead of chiming in with bad rumours - what the outside world says about the company and products might not necessarily be the real picture.

**Example:** It is pretty scary just how big a business company rumours is. Sites like Valleywag, Alley insider and lately sadly enough more and more TechCrunch are always on the lookout for a juicy story that rivals the celebrity rag mags. Proper investigation and separation of rumours and facts is not high on the list - what stirs emotions is what a successful news post is about.

There are a lot of things you have to be aware of when you are a developer evangelist. One of them is the enemy within the company.

## Prepare for prejudice

You have a very unique and new role which spans various traditional roles. Especially for developers moving away from the day to day grind can easily be seen as betraying the cause. Prejudiced developers are amazingly proud of being the delivering part of the company and consider anyone who does not code superfluous. I am sure you met these people and heard statements like “I don’t know why we need designers, I know how to use Photoshop, too!”. It is somewhat ironic that the people whose standing in the company you try to improve are the ones that are likely to be opposed to your role.

In the end, this is something you will have to live with. You will burn a few bridges and you will have to suffer a lot of internal and external abuse and nagging but it is important to keep your eyes on the goal. If working for 12 years as a web developer has taught me one thing then it is that delivering awesome work pleases first and foremost yourself. In order to have an impact in the company and to get things changed you have to take your hands off the keyboard and start talking and persuading.

**Don’t get discouraged by people seemingly stabbing you in the back** - in reality this is exactly the miscommunication and lack of people skills that you try to help out with. See it as a challenge and

not as a show-stopper. Once you can show successes of improving the standing of developers in the company you can go back and call a comparison match. Most of the time this is not necessary any longer though as the terribly grumpy people either left the company or others start telling them to stop.

## Deal with company changes

As the IT market is constantly changing so do the companies in it. This can be a great thing but in most of the cases it will mean that you have to deal with happenings that are beyond of your control. Mergers, acquisitions, new products, products being discontinued, rounds of layoffs, redundancies - all of these things happen, so be prepared.

**Fact:** The sad fact is that none of them ever have anything to do with the quality of your work. The concept of shareholders buying into a company because they believe in it has been dead for years - it is all about trading stocks and moving them to make money in the process. This means that any publicly listed company has 3 months time to make any new product a massive success as this is the time in between shareholder meetings. If there is no immediate success then the product gets canned and the people on it with it. This is a terrible state of affairs and it leads to mediocre and short-win driven products instead of great, lasting experiences. And the latter is what good developers are all about.

The thing about you being a developer evangelist is that you are in the spotlight of the world and the company so you'll be the first to be asked about an opinion when there is a big change in your company. This can have disastrous effects - either you violate a company policy and tread very hard indeed on the toes of your legal, HR, marketing and PR team or you sound like a company drone to the geeks in the company. Here are a few things to be aware of:

- **Every change in the company has a legal process** - while you are not an official press channel to the outside world your statement can be misquoted (oh and it will) and get you into legal trouble with your own company - liaise with PR and legal as soon as possible when there is a change. Then tell the developers in your company the legal implications.
- **There is no "off the record"** - neither internally and especially not when you are talking to the press/bloggers/outbound channels.
- **Switch to listening mode** - during the first few hours after a dramatic change listen to everybody and keep your eyes and ears open to what people say. This helps you to stop people from completely destroying their professional standing in the company and the market and to learn about what is really happening. Don't be part of the noise that drowns the facts.
- **Don't act emotional and make assumptions** - almost any change in the company will annoy people. In the heat of the moment this can lead to very dangerous comments, assumptions and truisms which will come back to bite people in the butt a week later. Be aware of this and don't fall into the same trap. Instead be ready to bail these guys out later on. Get the facts and before you say anything have a backup that can verify what you are saying.
- **Make your knowledge level clear** - if people ask you what is going on don't say "no comment" as that implies you know something but are not allowed to say it. Simply state that you are not in a position to know yet but that you are investigating.

**Tip:** This last point is very important. As your job is to communicate with everybody in the company people already assume that you have a much more in-depth knowledge than they have. As developers get very paranoid about changes in the company this can mean that people think you know all about what annoys them but hold back as you are siding with “the company” or “the management”. Don’t lose your contact with and the trust of the developers in the company over something you have no control over.

**Example:** One very interesting example of a change was when in one round of redundancies a broadly liked and respected very outgoing and extremely talented developer got made redundant. The developer was also part of a team that built one of the most important products at the time. The mailing lists where flaming, people were twittering about the unfairness of it all and in general everything pointed towards the company having completely lost its marbles. Speaking to HR I got to learn that the reason he was picked was that **in a previous round of redundancies he had filed for voluntary redundancy** and naturally these were the employees who were picked first. In addition to that talking to the developer himself I found out that he was totally OK with the decision and looked forward to having more time to look after his pet freetime projects and turning them into businesses. All the emails and comments going out failed to either address the developer himself or HR for a statement and this resulted in an avalanche of misinformation and bad blood.

## Be there for internal developers

You traveling the world and seemingly spending your work time aimlessly surfing the web and twittering can give a wrong impression of you losing interest in being a developer. Work around that by not losing the connection with the people in your company that build things and listen to what they do.

**Tip:** Don’t get bogged down in detailed problems of other developers though. You should be a voice of reason, understanding and pragmatic solutions and you can only be that if you see the work being done from a vantage point.

More importantly listen very hard when developers feel hindered in delivering their job. Then talk to their management about these problems. Keep these talks anonymous and show the impact these problems have on delivery, employee retention and quality of the products your company builds.

Any change for the better you can cause and any improvement in the areas people are concerned about that can be attributed back to you allow you to not be part of the delivery without losing geek points. You achieve this by talking in the right language at the right time to the right people. Developers are always asked to deliver yesterday what is defined in a week’s time and never feel the chance to voice their concerns. Make them aware that you can be their spokesperson and that you are there to demand time for discussing problems with their managers.

**Example:** I was very touched when one of my former colleagues asked me to have a few late night phone calls (over Skype) to talk about his future in the company and an offer he had from another company. I was also very annoyed about how frustrated he was about not feeling any joy or

company. I was also very annoyed about how frustrated he was about not feeling any joy of empowerment in his job. The reason was that he didn't dare talking to his manager. My main question in these calls was how he feels about going to the office every day and he said that he dreads it. The decision was clear: if you do not like going to work and you don't feel you have a chance to cause any change - leave. He is now in a new role in a different company, feels very excited about the challenges there and left the company not in anger but with a feeling of accomplishment and knowledge that there are caring people in his old company and not only the ones that made him leave.

## Work with PR and marketing

As stated in detail beforehand your role as a developer evangelist means that you are in between classic outreach departments like PR and marketing and developers. The danger there is that these departments could see you as competition.

Therefore it is immensely important that you keep on good terms and in constant communication with these departments. The reasons should be obvious:

- **You don't want to give mixed messages** - different views, yes, but you should both at least promote the same products to different audiences.
- **PR and marketing know legal implications you don't** - so make sure you chat with them before prematurely releasing information or promoting products that are about to change drastically.
- **These departments have already established communication channels** which can give you a good way in to speak at conferences and work with the press.
- **You can learn from their experience** - most probably these departments will have people that are on the job longer than you have been and can predict patterns.
- **You can feed back state-of-the-art developer information** - validating the impression PR has of a new product with realities makes sure that over-ambitious advertising doesn't promise developers functionality that you'd have to explain is not available yet.
- **Sharing contacts can open doors** - you might have a way into companies and publication channels that PR always wanted to have but couldn't find.

Only by liaising with the other outreach channels of the company you can make sure that you give the same message. You don't want to be seen as a threat.

## Be known as an outward channel

As a lot of people talk about the company to the world on different levels it is a good plan to tell the company about your outward communication channels. This ensures that you are not approaching the same people in parallel and possibly give mixed messages effectively undermining your and your company's credibility. It also shows the company that you are a person that reaches where they can't. List all the places where you publish:

- Blogs

- Blogs
- Magazines (print and online)
- Mailing lists
- Forums
- Conferences
- Professional groups and bodies
- Social networks
- Social apps

Also make sure to tell people that you have invite codes and accounts for products (if you have them of course). This works a treat and stops people from having to sign up themselves if they just want to have a look.

## Train other evangelists and developers

Just like clever developers share as much of their knowledge with other developers you should plan for training people in the company to do what you do. The reason is the same in both cases: you can share the workload and you can be sick or take a holiday. Another thing you can do is target new ideas and spend time on other plans what to do with your (professional) life.

**Training evangelists is a tricky thing** - by definition evangelists should be found and empowered and cannot really be "made".

Just like you use your powers of communication and persuasion to bring products to developers you can use them to bring potential evangelists out of the woodwork:

- **Make the company aware of the communication channels** to the outside world. Say that the blog you have is successful and that you are very happy to publish in-depth blog posts about current work and best practices used in the company. Then offer help with writing those.
- **Tell the company about events** - both the ones you organise and ones that happen in the area. This is especially necessary when you can't attend them. Offer to support a developer who wants to go with free goodies to give out (stickers, shirts...) - if they go and bring back photos and information how it went to write a blog post together.
- **Offer specialised internal training and talks** - as the prospect of evangelism might still be alien and even scary to people cut the training offer down to things that can be applied in any professional role: writing for the web, public speaking tips, finding great web content (well, basically the different chapters here).
- **Share great responses from the outside world** - send out for example a newsletter of "happy geek quotes" with tweets and blog posts about how a certain event or product launch was received in the developer world.
- **Ask people to challenge your products** - run some internal competitions to change or collect ideas about how people in the company would like your products to change. In my company we have hack days for that and more and more companies start doing the same.

**Tip:** A lot of this will tie in nicely with communication channels that PR and marketing already use. Ask them for help instead of doing your own thing and creating confusion and trespass on their territory.

## Share useful technology

As a developer evangelist, you will have the hand on the pulse of technology. Not everybody has the time to keep up the same way - actually hardly anybody has. That's why a very interesting part of your job is to communicate great technology finds with your company.

So if you find great tools that make everybody's life easier, share them with the company. This can include things like screenshot tools to stop people from sending you Word documents with embedded resized bitmaps, translation tools, communication tools and basically anything that you use to save time every day.

**Example:** One thing you will find is that if you are an early adopter of technology the main market will get to know some of them a few months later. Twitter and Oprah using it is one example. Another is blogging. In my current job I had great success by using both and then helping marketing to set up blogs and HR to use Twitter. This creates an amazing amount of goodwill with these departments and it is an easy step for you as you are already excited and knowledgeable about these technologies.

# Prepare for outreach

As a developer evangelist a lot of your job is going out there and tell the world about the things your company does (or technologies, techniques and methodologies it uses).

Your success in this is to a very large part relying on how the world sees you - are you a tech guru or somebody who just tries to sell their company or some product?

**Your integrity is your main weapon** - you have to make very sure that it stays intact. This means first and foremost that you need to prepare properly before going out there.

## Get your facts right

You will be asked to talk about a certain new product. Make sure that you are up-to-date on the matter before you go and speak about it. Do not promise things that are not under your control. Talk to the product team and ask them **in meticulous detail** what the product is about, what works, what doesn't and so on. Be as skeptical as possible as this is what the people you are about to talk to will be.

## Know the audience and their needs

Your communication should be targeted to the audience. People came to listen to you or read your article with a personal agenda - if you fulfill that agenda you win. Know what people expect and need and you can deliver. Otherwise you need to hope for the best which is never a good plan.

Going to conferences costs money. Going to free events where you speak costs time. Make it worth while for the people who do either and try to get something into your presentation that they can go back to their company with to wow their bosses. That way they will be able to go to more conferences and your other events.

**Example:** I once had to give a talk about Yahoo BOSS to a search engine optimisation crowd. They loved that Yahoo's search index is open for remixing but also were **very** aware that in the country I gave the talk Yahoo only had 5% of the search market. I worked around this by building Keywordfinder, thus giving them a cool tool to get keywords related to certain topics and showing them how versatile BOSS is.

## Have expert backup

You cannot be the expert in everything. In the best case, when giving a presentation try to have an expert at hand to answer tricky questions for you. If there is no expert available at the time note down the question and follow it up after consultation. Do not promise to come back to someone and then forget to do so - that'll make you look like you needed a fast way out! There are far too many speakers out there who play the "I'll get back to you" game.

Under no circumstances try to wing it and say things you are not sure the product team will be able to deliver. You are here to promote what can be used, not put pressure on your colleagues by promising the world the moon on a stick.

## Choose the right medium

Your communication should be in the right format for the intended audience. This can range from slides, over videos and audio to live coding exercises or online step-by-step examples.

**Tip:** My rule of thumb is - the more technical the audience, the less you should use powerpoint or keynote. Show how you can code with the product, not how shiny it is or what its workflow is.

## Plan for failure

Things will go wrong and you need to be prepared. In the case of a presentation do this:

- **Have your slides online somewhere** - in case your local copy dies.
- **Have a memory stick with your data on it**, in case you need to use a computer that is hard-wired into the AV system.
- **Prepare to not have your slides available** and still be able to do a Q&A; session.
- **Don't expect any technology to be available** - bring your own connectors, power cables, network cables...
- **Don't expect to be able to go online** - or bring a 3G stick as backup if you really need to be.
- **Aim for a 800 by 600 pixels resolution** and expect the worst possible colour setting and very low contrast.

**Example:** I once was asked to give a presentation in a pub for about 60 people and when we arrived there was no projector or TV or anything bigger than my laptop. The workaround was to ask a colleague to bring a printout of the presentation and show it Bob Dylan - Subterranean Homesick Blues style.

# Get speaking opportunities

Like any other skill, becoming a great public speaker means first and foremost doing it a lot. Sure, you can read all about the art of public speaking but the main trick is to get in there and overcome your fears and uncertainties.

The crux is that it is not easy becoming a speaker at conferences. This is partly based on the decision of most conference organisers to play it safe and only use well-known speakers. This again is based on the fact that audiences in surveys keep requesting well-known speakers and this is - sadly enough - how we keep the speaker circuit from changing.

There is however a way out, which is to the most part based on getting known and using the great and newer concept of unconferences.

## Go to Barcamps

Barcamps are un-conferences, meaning they are gatherings of enthusiasts of networking and presenting. Unlike conferences they are free and one of the really interesting rules of Barcamp is that everybody who attends also has to give a presentation.

These presentations are normally 15 minute slots and can be about anything that gets you excited at the moment. I've seen the full range of technical talks at barcamps but also knitting tips and a very thorough explanation why money is a big scam - which turned out to be very true a few months later.

Barcamps are a great opportunity to get your first experience in public speaking and allow interested people and conference organisers to spot you as an upcoming talent.

## Go to Meetups

Meetups are informal gatherings of people in the business to chat about hot topics and generally meet and get to know other local peers. Lately a lot of them have started to have a quick presentation to start the event and this is your chance to get a foot in the door and have an expert audience (or actually an audience of experts that are not the normal conference crowd) to speak to.

## Write articles

A lot of conference organisers also run online magazines. Carsonified for example, who run the "Future of..." conference series also run Thinkvitamin as their magazine.

Offer to write articles for magazines and keep a look out for article writing competitions and you'll find yourself invited to speak faster than you think.

**Tip:** Whilst in the last years magazines were flush with great content, personal blogging has made it harder for publishers to get good content every few weeks, which means that there is an opportunity for you right now.

## Offer Brownbags

Brownbag presentations mean that you offer to come to a company during their lunch break and give a presentation. In case you are wondering - that's where the name comes from - as Americans bring their lunch in brown paperbags.

Brownbag presentations are a great opportunity right now:

- **They are not that common yet** - so offering them to a company makes the person agreeing to it an innovator.
- **They mean that you reach people** that normally don't go to conferences but have quite a big word-of-mouth power.
- **Your presentation is not disruptive to the daily deliveries** of the company but adds extra value to a normal break.
- **They don't mean much traveling** as you can do them at local companies (granted, living in London makes it easy).
- **You get on the company's blog and extranet** and via that known to their clients and partners.

## Ask questions at conferences

The final idea to plant in your head is to **never be shy to ask questions at any meeting or conference**. I am a hundred percent sure that my success in my job had its main breakthrough after I asked detailed technical questions to the speakers at the @media2005 conference.

I had spent four months arguing with my boss to get tickets to the conference and that it is important for my team to go there and wanted to get my money's worth - and everybody going to conferences should do the same.

If you don't ask you don't get an answer - it is as simple as that. If your question is technically valuable and interesting you can be quite sure that the rest of the audience had the same problem, but were too afraid to ask.

# Deliver a talk or workshop

Once you got invited to speak, got your facts right and prepared for all kind of technical failure you can think about delivering your talk or workshop. This is where you need to make sure you do things really right as data is one thing but data being delivered in an engaging manner is so much more powerful.

**Fact:** There are no bad students or a bad audience - just bad workshops and talks. Your mood, dedication and enthusiasm do become those of the audience - if you are not happy, they won't be happy.

Public speaking is an art and there are a lot of tricks you can learn from acting or other forms of performing, but in the end all it boils down to is being prepared and happy to do what needs to be done. Someone has to tell all these people about the cool things you want to talk about - and if you don't do it some sales guy will. There are a few things to think about when delivering your materials:

## Be yourself

You will find dozens of books and videos on how to be a great presenter - however **nothing makes you a better presenter than being who you are**.

You should not have to play a role or dress up. If you believe in what you do, you will be great. Your best asset is confidence.

Confidence does not come naturally but it will get easier the more you present. Prepare your materials and expect everything to go wrong and there won't be any bad surprises.

If things go wrong - and they will - take them in stride. Say flat out when you made a mistake and get on with it. One of the main things to do is to give the audience the impression that whilst you are an expert you still are a human being, prone to error just like everybody else in the audience is.

## Invite communication

All of what you do as an evangelist is about communication.

You are a sender that brings a message to the audience but you are also a receiver that brings the issues, concerns and ideas from the outside world back to the company.

If you give a talk tell people that it is OK to ask questions. Make space in your presentations for that. Stop after a complex part of the talk and ask the audience if all of that was understandable or if you should repeat some detail. Ask them questions and have small presents for people who answer

should repeat some detail. Ask them questions and have small presents for people who answer.

**Tip:** It is generally a good idea to ask the audience questions from time to time. Ask them what they do (show of hands) and if they had experience with the product already and so on. This will give you an idea how to pace the rest of the talk but also makes people feel that they participate and keeps them from nodding off.

## Prepare takeaways

People should have the chance to concentrate on what you are saying and shouldn't feel that they have to jot things down to keep up.

- Have a URL where they can download your information afterwards and show this as one of the first slides.
- Have all the links in a presentation as a tag on delicious (or any other social bookmarking site).
- Say upfront what you will cover and what they will get out of it .

Making people guess makes them feel uneasy and that is not what people should feel like when they listen to you

## Plan time for and own the questions and answers

Plan for time for a "Questions and Answers" session after your talk (most conference organisers will do that anyways - but be generous). These sessions allow people to ask exactly what they need answered and go back and have a go at solving their problems immediately.

One thing that is very important is that **you need to be in control of the Q&A;**. A lot of times you will have people who don't ask questions but profile themselves instead. Deal with that accordingly - and swiftly. People will have real questions that need answering.

**Tip:** If you find a person in the audience that talks for a minute about their skills before asking a question cut in. Ask the person for their name and single them out as an expert. **"Great, Steve here knows a lot about issue XYZ, so during the break you can ask him about issues dealing with this. Steve, let's collect some of the questions around this and work through them later together."** That way you gave Steve a chance to deliver his knowledge, made him feel a million dollars and another person has a chance to ask a real, burning question.

## Be honest and real

If you don't know an answer - **do not speculate**.

Instead a great message to give is to ask the audience if someone knows. Normally there are other speakers or "silent experts" in the audience that can help you out. That way you show that you are open

to learn, too - and the stigma of the “arrogant speaker” is broken.

If there is no answer offer to investigate further and swap contact details with the person who asked the question. There is no harm in not knowing something. There is harm in lying though.

## Follow up communication

**Whatever you do - it is important to cuddle afterwards.**

In the case of a presentation this means that you should make sure you email everyone who gave you a business card (which can become time consuming but is something you can do on a train).

**Tip:** Make sure to blog, upload recordings, photos and slides as soon as possible. This shows respect to those who came to see you talk, and invites those who missed it.

Have contact options available after your talk (normally on the last slide) - email, twitter name and so on. The best plan is to have communication channels for that that are not your company mail or IM name you use at work. For starters this allows you to be more selective in answering but more importantly giving out company communication channels to anyone can be a security risk.

# Write great posts and articles

Writing for the web is a specialist skill and far too often you'll find people applying rules of other media when it comes to writing online articles or blog posts.

The main thing to remember is that text on the web is not published in one place but spreads throughout the web as bookmarks, links and references. Therefore it is very important to chunk texts up into easily digestible and repeatable chunks and make your headlines and document titles work without the context of the full text.

## Simple is not stupid

Writing very simple texts is hard work. Writing stupid texts is less hard. Consider the following traits of simple texts:

- Writing in simple terms takes a lot of work and thorough understanding of the subject matter. You need to be very familiar with the topic to be able to explain it in very easy terms.
- If you explain things in as easy as possible terms you make sure that you reach the largest amount of readers.
- Simple wording allows non-native speakers to get a chance to understand what the whole thing is about and maybe spend some time with a translation tool to make it work for them.

There is however a fine line between explaining complex things in simple terms and sounding condescending. Have someone else look over your texts to avoid this issue.

Read and re-read what you've written (take breaks in between) and make it easier in every iteration. Comparisons with real life objects work very well to simplify complex matters.

**Example:** At the last Open Hack day of Yahoo I talked to several press people and they were thoroughly confused about the concept of a hack day and open APIs. I explained it this way: "If Yahoo were a car company our newest models would be here totally dismantled and we'd allow people to play with the parts. Maybe they'll find a cool new way to assemble them that is the breakthrough we wanted in making cars greener and more efficient."

## Say what it is - don't sugar-coat it

Your heading and introductory text are the most important things of a blog post. Both determine how easy it will be to find the post in the future.

Newspapers have conditioned us to write clever, witty and interesting headlines with pop references and alliterations. That is fun, but you already have bought the paper once you read them. Web headlines become links and bookmarks and need to make sense without the rest of the text. For a technical blog post state what the post is about - do not try to be too clever. Pop references are even worse as these don't translate well to other cultures.

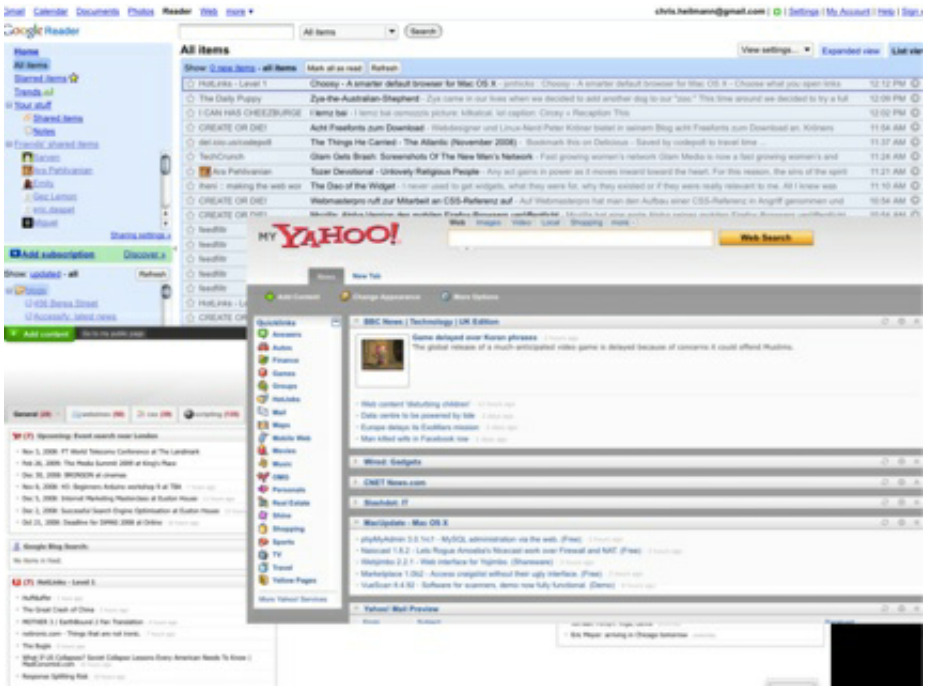
So do you want to be creative and witty for a minute or do you want to provide valid information for several months?

Blog posts should work like news items in radio:

- At the start of any post state what happened, where and how.
- Continue to explain what is coming in the post.
- Then go into details

This will prevent any confusion and put interested people on the way to find out more.

If you wonder where your blog headline needs to make an impact just think of RSS readers and personal homepages as shown in this screen shot:




Each of these grey lines are a headline of an article or blog post and this is where it counts to be informative. Does your headline both tell the reader what is going on and entices them to click it?

## Size matters

Technical writing for online use is about keeping things short and to the point. People are busy and want the facts.

So in order to write great posts, write them, read them, delete what is not needed, read again, delete more and so on. If you cannot take anything away any longer, you've reached the point of publication.

If you have a lot to cover, why not split it up into several posts? This will allow you to tease at the end of the first post that the next will be published in a few days and give you repeat visitors and readers on your blog.

## Add media

If you can, add relevant media to the post. An introductory photo invites the eye and lures the brain into reading what happened. We're lucky that these days embedding video, audio and slides is as easy as copy+paste.

Embedding ties our information together in a nice, easy to digest bundle. It also allows visitors to skim over the post the first time and come back to take in the rest (watch the video, download the podcast) later.

This also helps people who have a hard time reading but are very much capable of listening or seeing.

When embedding media make sure to also write explanatory text - images need good alternative text and videos at least a description of what they show.

**Tip:** Slides get much better by providing a text alternative or your notes. SlideShare automatically creates text versions of your slides for you. Personally I start with my notes and then create my slide decks from them - that way I do always have an HTML version to link to from the deck.

## Structure your content

Structuring your content is very important. Giving readers landmarks to take in your information one chunk at a time allows them to skim the post before deciding to read it and as people are busy this goes down really well.

You achieve this by using a hierarchical heading structure. This will also help people with assistive technology like screen readers to jump from section to section quickly .If you add IDs to the headings people can bookmark parts of the document and send links to friends that will bring them directly to where they want to be.

Structuring a text also means using short sentences. It means paragraphs dealing with one thing at a time. It means using lists to explain step-by-step processes or give an overview of what is available. For large documents it also means providing a table of contents which allows people to jump directly to where they need to be.

## Time-stamp your content

## Time-stamp your content

If you eat food past the “best before” date you get sick. If you don’t time- stamp your publications they will be considered great forever - even if they are harmful by the technical standards of the future. They’ll be quoted - sometimes badly - and re-iterated over and over again and taken as gospel.

**Example:** When I started as a web developer tables for layout were the only way to build a site. And you had to know a lot of tricks to make those work for Netscape 3 and IE4 and other fun browsers. I published a few articles on how to do that. Nowadays table layouts are counterproductive to web development and actually render much slower on modern browsers. If my articles didn’t have a date on them maybe people would still consider them relevant. All the “CSS layout is too hard” posts out there surely hint at that.

Our technical environment moves at breakneck speed. What was “best practice” half a year ago might very well be “considered harmful” now. So let’s make sure that readers know when a certain document was written before following its advice even now.

## Cite to prove

The last, very important point in this section is to cite other sources and link to content you have built upon. By citing other sources (and reading them of course) you validate your thoughts and facts. Readers don’t have to trust you blindly – they can make up their mind by comparison.

# Write excellent code examples

Code examples are what makes your posts and articles very relevant to developers. They say more than dozens of pages about a certain product and - even more importantly - they invite people to play with your products.

## Solve a problem with your example

One of my biggest bug-bears are “hello world” examples that don’t do anything useful. Instead I aim to give out code that solves a real problem. I will come back to the “hello world” dilemma in the presentation tips chapter but for now, let’s just say that writing **“hello world” code teaches people to write code, but not how to solve issues with it**.

Good code examples should answer the “how does this technology help me solve a problem?” and not the “how do I use this?” question. The“how do I use this?” question can be answered by documentation. Code examples should get people excited about using the product and entice them to dive into the documentation to find out about the details.

So instead of starting by reading the docs yourself, check what the tool can do and look for a problem you always wanted to solve that can be solved with this technology. Then solve it using the product and explain what you’ve done as the code example.

## Show a working example

One of the first things to show in a code example is a working implementation. There is nothing more powerful than a way to see what the thing does by clicking a link or entering some data and sending off a form. You telling the readers that it works is one thing - the readers being able to try it out and seeing it for themselves is much more rewarding.

**Tip:** If the code you explain is part of a larger interface, behind a firewall or needs authentication then you can still show how it works by recording a screencast.

The working examples should of course work but also be pretty and smooth. A lot of examples fail to please the eye or actually violate a lot of usability basics - don’t give a shoddy first impression.

## Explain the necessary environment

One of the things you want to avoid is people getting excited about your code and then not being able to

make it work in their own environment. You get around this issue in several ways:

- **Write defensive code** - check for dependencies before trying to access them.
- **Check online and HTTP status** - If your code needs to pull data from a web service say that the user needs to be online to run it. For example I've had many a complaint that my Ajax examples don't work offline or without running a localhost. Obviously I failed to explain that Ajax is meant to work over HTTP.
- **List what is needed** - say what your specs are, f.e. "needs PHP5 with cURL and imageMagick" or "Firefox 3.5, Opera 9 or Safari 4 needed".
- **Provide a simple test script** that checks for the right setup. (For an example see the test file for GeoMaker).
- **Provide a developer key for the demo** - but make it obvious to tell people that to implement your code they'd need an own key. Provide a link where to apply for one.

You won't be able to predict all things people can do wrong when implementing your code but these are some good ways of preventing frustration. Which brings me to a very important part of code examples: allowing for copy and paste.

## Write working copy and paste code

Copy and paste is probably the most used way of learning code. You can write documentation until your fingers bleed, but the biggest use case is that developers will check a code example, copy and paste it, fiddle around with it until they get stuck and then start reading the docs.

Therefore it is immensely important to **write very good copy and paste examples**. Any bad coding habit you add in your code will be copied and become part of a live implementation.

Copy and paste examples that don't work are not only useless but also disastrous to your reputation and very frustrating for implementers. Therefore make sure that the following points are covered:

- **Link all resources** - point images, CSS and script resources to web locations rather than linking them locally or relative. People will copy the code and paste it in a local HTML file and not download the dependencies.
- **Provide the full script upfront** - people will copy and paste chunks of a script and complain that they don't work without realising that other parts are missing.
- **Validate and secure your code** - copy and paste code needs to be excellent as in many cases it will be what people use. Make sure your code plays well with other code and doesn't cause any warnings or errors.

**Tip:** The best way to keep code examples and code in sync is to generate the code from the source comments. I've written a PHP script that does that for me called Tutorialbuilder which automatically applies some of the tips here.

## Have the example as a download

Providing the demo code for download as a zip file should be one of the first things you do in your example. You can ask readers to download, unpack it and code along with you (this works well with screencasts and video tutorials).

In any case it keeps your article in the mind of the readers as they have something on their hard drives reminding them of it.

Offering a zipped version of your demo code can be annoying as every change means you have to re-pack the demos. Luckily by using a hosted code solution like GitHub this job is done for you automatically.

## Write clean and clever examples

Again, I cannot stress enough that **your code examples should be the cleanest and cleverest code you ever write**.

It is very tempting to show a quick and dirty solution to get people going and earn immediate kudos for creating the shortest code ever but this is what coders do to impress each other and not what evangelists do.

You want to **show how to write excellent solutions** with the product you evangelise and every shortcut you take will be copied and taken as an excuse not to write excellent code in live projects. This is not what we are here for.

On the contrary - demo code can advertise best development practices and show them in context rather than just as an academic exercise. Showing a JavaScript that protects scope by using the Module pattern and uses closures cleverly allows you to cross-link to these ideas and maybe get some developers to take them on as part of their coding habits.

## Build code generators

A very nice touch to add to a solution are code generators that allow implementers to just add some parameters, hit a button and get the code they wanted. This is amazingly powerful.

**Example:** Probably the most impressive example of this is Dav Glass' Grids Builder which made it dead easy for people to use the YUI grids without needing to learn all the class names and IDs. Another success I had lately was GeoMaker showing off the power of Placemaker.

The only danger there is that some people will never bother learning the real implementation tricks, but on the other hand these readers are not likely to do that anyways. In any case you'll get a lot more people look at the product.

# Prepare great slide decks for presentations

**Slides are a tricky thing to get right**. The main problem with them is that as a developer evangelist you have a technical audience (in most cases) and slide decks are anathema to us. The term "death by powerpoint" is much more than a Dilbert cartoon. Sadly enough a lot of our day to day life in offices consists of sitting in a room trying to look alert whilst slowly dying inside as some old-school presenter shows us just how many bullet points you can cram into one slide.

Again, as with the other chapters a lot of what you will read here will vary for your experience and environment but I found the things I am sharing here very helpful in my quest to bring technical goodness to the starving masses of under-appreciated developers. Furthermore, I got a lot of good feedback and high viewing figures on SlideShare for my slides which can be an indicator that I am doing the right thing.

## Know your stuff

The biggest mistake that presenters do is to rely on their slides as their main source of information. If you don't know the subject matter or you are not excited by it or you haven't done much work with it you will give a bad presentation. Nothing makes you a better presenter than confidence in the subject and hands-on knowledge.

You will sooner or later be asked to stick to company approved material or "re-use this great deck XYZ has done". Try to avoid this as much as you can. A presentation is you telling people about what you think is important that they hear about. If you have no clue what the issues with the product are or if you don't really care about it you will get into trouble. Technical audiences are amazingly good in spotting what you don't know and will make that the first question in the Q&A session. It is a geek alpha male thing and you will have to be prepared for it.

There is nothing more painful than a presenter turning to his slides and reading out what is standing there. You don't want to become the people that bored you to death before. Also, as mentioned in the "Deliver a talk or workshop" chapter, audiovisual equipment hates presenters and your slides might not be available for you for one reason or another. If you know the subject matter and you are excited to talk about it you will give a memorable talk regardless.

Furthermore, this is you presenting. If the slides are not your style or your language you will appear stilted and you have to remember what the deck says. Public speaking is about giving information in an entertaining fashion - not acting. You should not have to play the role of "corporate speaker" but instead be you. Only then you will be believable and effective. More on this later.

## Start with the content - not the slides!

The first mistake people make is to see the slides as their presentation. The slide deck is an aid to make your presentation easier to take in and more enjoyable for the audience. For you as a speaker they are the narration thread - a reminder of what you want to cover in your talk. A good speaker can keep a room of people interested without any slides whatsoever. A good slide deck, however, gives people memorable moments and information they might miss if they just listened to you.

## Start with a highly portable format - HTML

When I write a new slide deck I start with a text editor. I write the story of my presentation and I follow the same rules as for writing online articles. That way I make sure of a few things:

- **I know the content and the extent of what I want to cover** - which also allows me to keep to the time limit when presenting.
- **I have the information in a highly portable format for people to read afterwards** - by converting it to HTML later on or blogging these notes.
- **I already know all the links that I want to show and can create easy-to-find versions of them** - for example by bookmarking them in Delicious.
- **I don't get carried away with visuals and effects** - which is a big danger when you play with good presentation software.

**Tip:** Having these notes makes sure that you will have something for people after the presentation to read. You can mention this before your presentation and give them the URL. This relaxes audiences immensly as the first question at every conference I get is whether the slides will be available or not.

## Pick a presentation tool that helps you present

Once you know the content, you can start putting together your slide deck.

Choose whatever presentation tool that makes you happy and allows you to simply put your slides together. Personally I use Keynote, but if forced I can also stick to PowerPoint.

There are a lot of presentation tools out there that work with HTML and in browsers and use web standards. I've written one of these myself in the past as it was the thing to do (Opera even comes with one built into the browser). Over the years I found them all to be sub-optimal. The reasons are the things a presentation tool has to do for you, which are:

- **Display your slides on the screen regardless of resolution** - some projectors support 800x600, others go up to 1280x1024.
- **Use, crop and resize images easily** - you will use a lot of imagery and it is never in the right format.
- **Allow you to position elements freely on the screen** - sometimes you need things next to

- **Allow you to position elements freely on the screen** - sometimes you need things next to another, sometimes you need to overlay a URL over an image.
- **Support remote controls** - as you **should** walk around during your presentation you should be able to use a remote control instead of hitting the space bar.
- **Have a way to transition smoothly from one slide to another** - this is a subconscious thing but it makes your slide presentation so much more enjoyable.
- **Be full-screen** - browser bars or copyright lines and headers are distracting the viewers.
- **Have a way to blend things in one at a time** - this helps your narration and you don't need to repeat content in several slides.

I am sure giving enough time all of these are possible in browser-based presentation systems, too, but why bother wasting time on this when there are perfectly capable systems available?

**Tip: ** whilst Keynote is the bee's knees when it comes to functionality and style, the file format it creates (essentially a packed folder) is a pain to send to other people. Either you have to zip it up or - if you just need to hand out the deck without further editing - export it as a PDF. When you export make sure to uncheck "Print each stage of builds" and "Add borders around slides".

## Illustrate, don't transcribe

Once you wrote your content and picked the right tool it is time to write your slides. As mentioned before, what you should remember here is that your slides are not your presentation but its outline. The slides are there to keep your narration flowing and illustrate to the audience what you are on about at this point in your talk.

Human communication is to a large part body language and you standing there and reading from your slide deck or -even worse- turning away from your audience to see what happens on the big screen is communication suicide.

In addition to that it means that you cannot concentrate on the audience. Checking the audience and their body language is a large part of giving a good presentation. **It is not about you celebrating yourself and giving a show but about you bringing information to the audience in an engaging and interesting way**. You can only do that when you can see the effect you have - not when you have to read what you want to say.

Thus you need to find a single sentence or even a word, a picture, a screenshot, some graph or some illustration that explains and accentuates what you want to talk about in this segment of your presentation. That way you don't overwhelm the audience with things to read and look at but you enable them to concentrate on you and you enable yourself to be free in your delivery and - if needed - alter your delivery style to stop the audience from nodding off or leaving.

Let's have a quick example. The following information is what I had in my notes:

> The way to have fun with the web of data is to distribute ourselves around the web and bring the

data back to our sites.

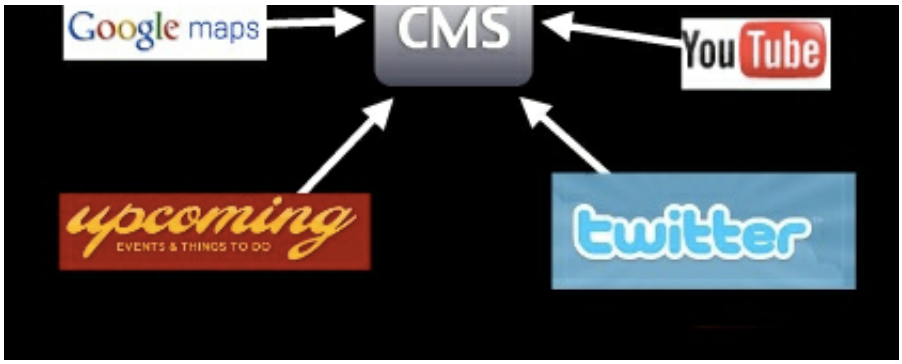
The first step is to spread our content on the web:

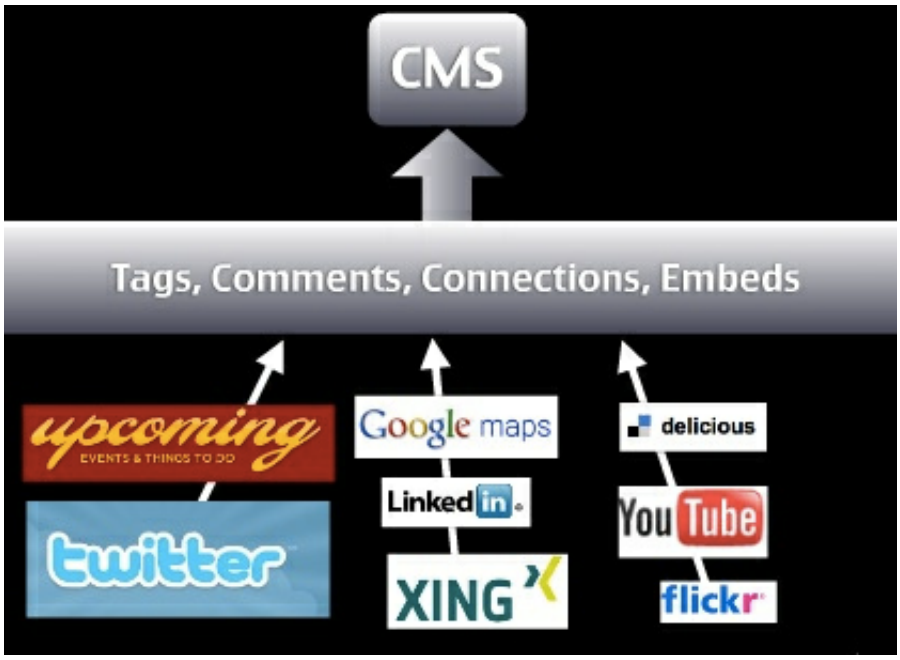
- upload photos to Flickr
- bookmark and tag URLs atDelicious
- write short and succinct news updates at Twitter
- upload videos to YouTube
- link addresses and set up driving instructions with Google Maps
- write CVs and bios at Xing or LinkedIn

The benefits of this approach are the following:

- The data is distributed over multiple servers – even if your own web site is offline (for example for maintenance) the data lives on.
- You reach users and tap into communities that would never have ended up on your web site.
- You get tags and comments about your content from these sites. These can become keywords and guidelines for you to write very relevant copy on your main site in the future. You know what people want to hear about rather than guessing it.
- Comments on these sites also mean you start a channel of communication with users of the web that happens naturally instead of sending them to a complex contact form.
- You don't need to worry about converting image or video materials into web formats – the sites that were built exactly for that purpose automatically do that for you.
- You allow other people to embed your content into their products and can thus piggy-back on their success and integrity.

The two slides that went with the information above where the following:




Instead of telling all of these things, I gave people a visual, using the logos of the companies as something they already know and showing with a few arrows what I want to bring across. I was able to talk through the services one by one and say what people can do with them. The second slide then showed the benefit of piggy-backing on the integrity of these services. Add a practical example of what can be done with this approach and you have yourself a great segment of a talk.

## Use and find images

**Images can be a very good way of getting a message across**. You've probably seen beautiful presentations with inspiring pictures of swooping eagles and calm waterfalls but this is becoming cliche really fast.

I use images for two reasons: to have something unexpected and fun (yes, mostly kittens) in my slides or to connect to a real life scenario.

**Example:** When giving a security talk unstead of showing a web site with a security flaw I show a lock that has been picked, or a badly locking door, or something similar that bridges the gap between a hard to grasp concept and something that is obviously a problem or a bad solution. If I talk about badly implemented accessibility I don't show ugly web sites but a wheelchair ramp with a step in it or a wheelchair accessible toilet behind a door that is too narrow.

**Use imagery to illustrate your point, not to "make it pretty"**. Pretty imagery might be more distracting than helping and for making it pretty you got colours and typography.

**Finding images to use these days is easy**. A free and good resource for images is Flickr. Make sure that you use the advanced search and that you tick the boxes for Creative Commons licensed photos that you are allowed to use commercially, manipulate and build upon. The latter is needed if you want to crop the photos.

**Example:** Say for example you want the photo of a lock, check this search link. Each of these photos you can use in your slides and all you have to do is to thank the original photographer by mentioning them by name. I normally include the URL of the photo on Flickr in the slide, too, so that other people can re-use the photo if they want to.

No need for expensive stock photography of multi-ethnic people in suits high- fiving or shaking hands - tap into and participate in Creative Commons and we all have more interesting slides.

**Screenshots are amazingly powerful**. Instead of just pointing out a resource on the web your audience can check later, make a screenshot of the web site and overlay the URL on the slide. That way people have a visual idea of what the resource looks like and get a much stronger "Oh I remember this" moment when they visit it.

The same applies to interfaces of systems - if you show and explain you reach more people than when you just explain.

**Tip:** For creating screen shots I use Skitch, which is not only terribly easy to use, but also allows me immediately to add some arrows and explanations and upload the final product to Flickr.

## About code examples

Code examples are what a lot of presenters spend far too much time on getting right. You want a good mix of readability and at the same time make it easy for you to change the code. Presentation software by default is not meant for code display. There is no monospace setting, quotes get replaced by "smart" quotes, indentation is all wrong, you have less space than in your code editor and many other problems.

Code examples are however very important as they show people how they can immediately use what you are talking about and you bring the topic you cover into an area where they feel home as they use it daily anyways. Showing a few lines of code and what they do in a browser is much more powerful than ranting about the amazing features of the product you talk about. It comes back to the "what is in it for me?" that you should always try to answer with your evangelism.

Here's what I do: I write the code in my normal editor (TextMate), bump up the font size a bit and then take screenshots. This has a few benefits:

- **I have nice colour coding** which increases readability and helps understanding the code.
- **I have the right font and code layout** and none of the “magic quotes” annoyances.
- **I maintain the code in one spot** and a code change means simply having to create another screen shot.

Provide live demos and downloadable source packages of your code as explained in the “Write excellent code examples” chapter and everybody wins.

## About sound and videos

Sound and video are powerful tools for training and illustrating. For example I found that a two minute screencast of some system makes it much easier for people to find their way into a system than lots of clever copy (which you’d need anyways cause not everybody can see and hear video).

In presentations, however, I find it more distracting and anything else. Yes, it enriches your slides and teleports you into the post 1990 presentation league (and on Macs it even works), but let’s face it - it is also terribly interupting.

As a speaker you normally want to be the person listened to. You also can use your body language to emphasize the message you want to give. From time to time you point out information on your slides but you bring back people to your narration. If you use music or video in a presentation you create a pause as all the senses of the audience are busy following what is going on on the screen. It also creates some time in your presentation where you are part of the audience as you turn and watch the screen (facing the audience while they watch a video is creepy).

Given the disruptive nature of multimedia elements I try to avoid them whenever I can. They look cool but you’ll find soon that they are more hassle than they are worth. For example:

- You expect the AV equipment of the location you give your talk to be able to show video and have audio for you and your computer (good luck with that one).
- You expect the projector to be able to show video. Especially with windows machines and older projectors video is often not visible because of refresh rate issues.
- You make your slide deck impossible to distribute unless you turn it into a video.
- You lose the rhythm of your talk, in essence you create a break that you need to pull the people out of again.

That doesn’t mean you cannot talk about videos and screencasts. I normally upload them to YouTube, take a screenshot of them and show them as a slide together with the YouTube URL (which I can link and also provide in the HTML notes). That way you can talk about the video and don’t lose five minutes of your presentation. You can also explain what happens in the video and how it is relevant to what we are talking about. That way people who can’t see videos still have an idea what is going on here.

**Example:** one exception to my rule is when you want to provoke a very emotional response and make

people understand something beyond their own experience that is very human. One example is that when I talk about the accessible version of YouTube I created. Rather than me explaining what effect these changes to the interface had it is more powerful to show how Lizzie, a user with learning disabilities and Kirin, a blind user can now enjoy online video. Especially Kirin's end sentence "The thing is the power … this has given me the power that it should give me in the beginning." is a wonderful emotional moment to get you back into your talk.

If you really feel that you need to use video or audio in your talk then use it at the beginning or the end of it. That way you either come in as a spectator and become the speaker or get out on the same level as the audience.

In most cases video and audio are extra bells and whistles. And a good talk doesn't need those. This also applies to transitions and effects.

## Don't bling it up

As mentioned before in this book you can put lipstick and a wig on a pig but it still would make a lousy date. A lot of presenters are happy to use every transition and animation the presentation software comes with but that doesn't make it better - especially when they look bad on a slow computer.

Animation rhymes with moderation and this is what you should always keep in mind. You want to make a point with your presentation and not overload the audience with whooshes and blinking shiny things that distract from your content.

Use transitions to make your slides smoother, use animation if you want to reveal something bit by bit and avoid having to jump from slide to slide.

**Example:** I like to use "fade through colour" as the transition effect between my slides which smoothly fades in and out in a second. I only use animation when showing screen shots and showing zoomed smaller parts overlaid or to focus on one part of an interface.

Used correctly, animation can be a very powerful tool to make a step by step process more obvious. If you are not skilled in usability and design though it will most likely appear tacked on and, yes, tacky as animation has been traditionally used to spice up very boring presentations.

The other issue is that animations can actually work against the flow of your presentation. Sometimes you want to speed things up and if you hard-wired long and complex animations you stand there waiting for your slides to catch up with your narration. Sometimes you also have AV equipment that cannot show animations and that makes you wait for something that never happens.

**Example:** when I was in Madrid, my talk was streamed on a video service of the university which meant the streaming server had to VNC into my laptop. As everything is wonky over VPN neither the animations nor the transitions worked.

One thing you can do is use overly outrageous animation in an ironic manner to show how annoying they can be. I've done that before when talking about accessibility or using JavaScript libraries. Only to be used in countries that understand irony though.

## Keep it brief

**Keep your talks brief and if possible cover one topic**. If you need to cover more than one make sure that you have a good narration flow from one to the other to avoid them appearing stitched together.

Presentations should bring home one message and that one well. This could consist of several sections but the overall story should be obvious. Try to give the whole talk one main theme and return to this in each of the sections.

As mentioned earlier, your slides should contain only what is necessary and not more. There is no point in reading from your slides as that would make you a member of the audience and you have a race who can read it faster.

I tend to have either only a theme per slide or one sentence (depending on the audience, which is the next section in this chapter) and I try to avoid bullet-lists at all costs - especially nested ones. These are old school presentation style and conjure up unpleasant memories of having to sit through two days of boring training sessions.

An agenda up front is a good idea if you really cover a lot of things but it also allows the audience to pick their faves and shut down in between which probably means they miss important parts of your talk. If you do a good job as a presenter the amount of slides is not a problem and neither is at which stage of the overall talk you are. People will be lead through it without realising it.

The overall amount of slides is only limited by your ability to go through them quickly. My rough estimate is a slide (which is one topic) a minute but I am also a very fast speaker.

## Consider the audience

One instance where I break my own rules of brevity is when the audience consists of people who do not speak English well and might have a harder time keeping up with my pace and the funny accent. When dealing with an audience like this, having a simple sentence per slide or even some bullet points and repeating them has quite an impact.

- **You keep things much more simple** and thus you don't make the audience feel inadequate or that they are missing important things.
- **You are forced to pace yourself** which is very important with an audience like that anyways.
- **You allow for better translation** in case you get transcribed afterwards or have live

translation at the conference.

In these scenarios (and especially in the Asian market where asking questions in front of a big group is just not normal) I also tend to keep my slides much more technical. Code is international and people can even repeat it easily and write essays about it in their own languages.

Another thing to remember when giving presentations in different cultures is that pop references and puns do not work. Don't expect the audience to know what you know and to be excited about what you are excited about.

**Example:** When I went to Sweden to give a talk I put in some slides about the Swedish Chef from the Muppet Show (here in my favourite sketch of all time - "Chocolate Moose"). Nobody got it as the chef is not called Swedish in the Swedish version of the show. I also added a "have a break, have a Kit-Kat" joke in there, and this was another ad only aired in the UK (I only knew it from "world's funniest TV ads" on German TV).

As said at the beginning, a lot of the tips here are for creating presentations for developer crowds. If you speak mostly to designers or management, other tactics have to be applied. All in all it is a good idea to question the classic way of presenting and slide design though.

## Corporate and conference templates

During your job as a developer evangelist you will be asked to use conference or corporate slide templates. Try to avoid doing that. The reason is that these templates are almost all the time targeted to the classic presentation style of one heading and 20 nested bullet points followed by a copyright line nobody cares about and other legalese things.

The reason is that using a corporate or conference template is good for the conference and the company but distracts you as a presenter - it is just not you. You and only you should own and run the presentation as it is your integrity on the line. Once a conference or company asked you to be a speaker for them they already trust you to do things right and there is no need to keep the corporate hat on and do a song and dance.

That said, there is a benefit to using these templates. In the corporate case you show a consistent look and feel to the world and align yourself with other publications. The question is if you want that. In my case, not looking like the slide decks of my company battles a lot of prejudices developers have as developers do not trust big brands. This is for you to decide. In the case of a conference looking the same as the others makes your deck more findable later on but at the price of looking the same as everybody else and having a distracting logo on each slide.

**The solution is to meet half way.** If you make the first page of your deck align with the others and then switch to your own style everybody wins. The cover sheet of your deck is only important in two cases:

- To fill the screen until your talk starts and
- as an eye-catcher when you later on send out the deck and show it in a blog post.

Other than that the slides should take the backseat and aid your presentation.

## Don't reuse without personalising

Another very common thing that will happen to you once you become a speaker for your organisation is that you will be handed presentation decks to present. "This has been done by Stephen from the US office and has been signed off by PR. Stephen can't come to the conference, so we want you to fill in for him. Here are his slides, good luck." is a sentence you will hear a lot.

**If that happens to you, be firm and say that this is not how it works**. You are a presenter - not a parrot. If the slides are not in your language, mirror your approach to a certain topic or talk about technology or products you are not firm in or have no control over you are treading on very thin ice. It is you on the line as the speaker and the success of your talk stands and falls with how you come across. If you can't be you, then don't do it.

**Example: ** This happened to me when Yahoo came out with the open strategy and our chief technologist couldn't come over for the Future of Web Apps London. Here are the slides where I refer to Neil's slides and the video of the talk. As you can see, there was no way I can present his deck as our styles are totally different.

That said, nothing stops you from using the information on the slide deck and translate it to your "language". Instead of flat out refusing to use the deck say that you are happy to take over and use the information but that you want to have a chat and information hand-over from the original author. Every speaker has extra information that makes slides make more sense and become more appealing and you cannot guess these things - you need to hear them "from the horse's mouth".

It boils down to this: giving the talk is one half of the whole show. You will have to answer questions and you will have to be able to explain the practical implementation of a certain technology or product. This you can only do when you played with it yourself and verified your findings with an expert.

Slide decks that get re-used without being challenged and changed become stale. For a company it makes more sense to keep a repository of facts and ways to explain a certain product than full slide decks. That way your information doesn't become stale. Wikis are perfect for that.

## Share and enjoy

Once you're done with your slides and you are happy with them don't forget that sharing is caring. Upload the deck to your blog, make it available as a download or - even better - upload it to Slideshare.

Slideshare is a great tool to get your slides distributed. People can comment on them, share them with

friends, embed your slides in conference blog posts or as a resource for a certain subject and many things more. It is what Flickr is to photos and YouTube is to videos.

**Example:** All my slides are available on Slideshare: http://www.slideshare.net/cheilmann/presentations and a lot of people re-used them in part in their own presentations.

Sharing your slides is what a lot of people in the audience will ask you to do and it will get you known as a speaker. People may stumble upon your decks somewhere else and learn about you that way.

# Record your output

Making recordings of your talks and in general your work is a good idea because of several reasons:

- **People who couldn't attend your session get the same information** - a presentation to me is much more than the slide deck - the deck is actually just the table of contents of the talk.
- **You have the chance to check how you come across to the audience** - I normally check my talks on my iPod whilst cycling in the gym - good use of the time and I can see where I need to improve.
- **People who prefer audio or video get interested in what you do** - this also includes people who need audio and video because of their condition - for example people suffering from dyslexia.
- **It allows you to publish in other channels** than just your blog or site or conference archive (more on that later in the web section).

Recording things is dead easy nowadays - you just need to know what to use and where to put it. All of the following tools are available for Mac, some are open source and also available for other systems. If you don't have a Mac - get one and thank me later.

## Record the audio of your talks

Most conferences will give you a lapel mike and do some proper audio recording but if that isn't the case or you simply want to have your own copy I found that the built-in microphone of a MacBook Pro is perfectly capable of making a good recording of your talk if you don't walk around too much.
The other option of course is to get a small external microphone.

For recording, I use Audacity which is a free, open source sound editing tool that has all the capabilities you need (record, cut, convert). Once edited, I put the file into iTunes to tag it and add the images for the coverflow for iPods.

For storage of audio files, I use archive.org which has a pretty nice uploading tool and comes with a good search functionality and in-built player.

Audio recordings are great as they are comparatively small and people can put them on their portable music players and listen to them on the train to work. In addition to that SlideShare also allows you to add sound to your presentations to make them slidecasts. The editor is easy to use and it gives presentations an extra zing.

**Example:** You can check my audio recordings on archive.org and see a slidecast presentation on

SlideShare.

## Shoot video

Having a video of your presentation or interviews is very nice, much for the same reason of having an audio recording. As there is audio and video it does give people the full experience of seeing you talk and the success of The Yahoo Developer Network Theater shows that people love to get videos and use them as training materials.

You can get small cameras with amazing recording quality these days and even some mobile phones allow you to record. Editing is a bit more complex than editing audio, but I found that using mpegstreamclip makes it easy to do simple cutting and conversion into almost any format.

Hosting is a bit harder - archive.org is again an option but the real power of online video comes from hosting it where people are used to look for videos and can embed them in their own blogs. YouTube is of course a main candidate (also because of their great annotation tool), but I also love using Vimeo. Uploading takes a lot longer than the smaller audio files so it is pretty tough to have a really quick turnaround.

## Screencasts and screenshots

Another really powerful tool to show people what you are doing are screencasts and screenshots. Sometimes a picture explains what you want to achieve much easier than a bunch of instructions. Step-by-step instructions how to use a certain interface (for example how to sign up for a developer key) are very easy to show as a screencast. Describing the interface with words is much harder - just try to explain people on the phone how to install Windows for example...

Personally I like to keep screencasts very small by just filming me going through some interface but you can jazz them up with voiceover or embedding your webcam, too.

By far the easiest screencast software I found on Mac is ishowu which is not free but dirt cheap. If you're on PC and have some more money Camtasia was another tool I used before.

For screenshots I found Skitch to be very powerful, especially as it allows you to annotate, highlight, add arrows and immediately upload it to the web.

## Link collections

Another great way to record what you have done is using social bookmarking (like delicious) to collect links for a certain event or talk that you've given. Instead of people having to remember all the links you've used in a certain presentation all they have to memorise is a single URL with a tag. That way you can easily find interesting URLs you talked about later on - simply get the link collection with the presentation tag.

presentation tag.

The other benefit of this is that people can tag and add notes to your links in the social bookmarking system, thus making them even more findable. Which brings us to a very important part of your job as a developer evangelist: using and knowing the web and the social web.

# Know and use the (social) web

The web is your biggest weapon as a developer evangelist. It is a world-wide, 24-7 information and communication channel and allows you to get your message out.

By understanding and using the social web you will also find that other people become relays for your great stories and that people will even translate and publish in their own markets for you.

Social media is an amazingly strong buzz word these days and a lot of money is being made by giving "social media consulting" or even "Twitter workshops". While the tricks told in most of these will give you instant success, this success will also fizz out faster than you can say "hype".

A lot of the "social media experts" sound a lot like the "search engine optimizers" of the recent past and use the same dirty tricks (for a humourous look at the subject check the complete social media douchebag).

Be aware of that - your job is to use the web as a communication and distribution channel - not to make a mint in a week selling snake oil. This would kill your reputation and as stated before your reputation, integrity and honesty is what makes you a developer evangelist. Losing these will make it almost impossible for you to get listened to in the future which renders you useless for your company and gives the general idea of developer evangelism a bad name.

## Find great web content

As the web is where you want to publish, you also need to be interested in it and find great things to tell people about. As explained in the Brand and Competition chapter of this handbook you cannot exclusively talk about your products.

The most time-efficient way to find content is to collect yourself some good RSS feeds in an RSS reader. Personally I use Google Reader and here are my subscriptions in OPML format.

Other great resources are Del.icio.us (as people tag and describe technical content really well), topical mailing lists and forums. Don't just find things - make sure that what you find was ratified by a human or humans you trust. Then you can safely re-distribute it.

## Redistribute web content

Once you found great content, re-distribute it. The reason is that you have a different network than other people have and you should never assume that people already know about the things you find

other people have and you should never assume that people already know about the things you find.

**Example:** The other day I found out about YouTube's TV interface and realised just how much more useful this is to elderly people or people with visual or motor impairments. I put it on Twitter and had an avalanche of re-tweets from the accessibility community and other developers. Mashable had covered this much earlier (and thus most likely TechCrunch and ReadWriteWeb, too) but there were still a lot of people who didn't know it - people that can show this interface to the people that really benefit from using it.

You can re-distribute web content in several ways:

- You can **blog** about it.
- You can **add it to social bookmarking sites** and add a good description and tags.
- You can **use it in a presentation**.
- You can **quote it in a mailing list or forum** to add more relevance to your post or mail.
- You can **Twitter** about it.

In any case, the most important thing is that you **attribute the content to the originator** by name and resource. The reason is simple: Clever web users track what people do with their content, so if you blog about a subject, your blog will show up on their radar. The same works for re-tweeting. In other words, you get known to them which could be the start of an interesting two- way communication.

## Be known on the web

If your job is to bring interesting news and explanations about web products to the web then it should be pretty obvious that you should not be a stranger on the web.

Sign up to mailing lists, post on forums, use Twitter, poke around IRC channels, leave comments on interesting news articles of tech magazines and online magazines - simply don't be shy to give your point of view or real advice whenever you can.

Be aware of new and upcoming networks and social apps and sign up for them as soon as you get the chance (more on this in the next section).

Being visible is especially important when you work for a large company. Tech news portals love to bring news about big companies - especially bad news. It is also stunning how many times these news items contain distorted if not outright wrong information.

As the marketing and PR departments of your company are most likely not aware of these tech publications and hardly ever would step in to rectify the mistakes in their reasoning this is a good chance for you to act. Stick to technical, real information and show proof of your points. Of course some commenters will side with the misinformation as it is just cool to stick it to the man and fight large organisations but the lesser vocal majority will at least get the real story straight from the horses' mouth. **Be very sure about your facts if you do that though!**

**Fact:** When you work for a large company people will automatically try to disagree with you and try to "out" you as a corporate drone. This always annoyed me to no end. If what we call best practices and standards are not used by large companies they'll never spread across the whole market - so fighting the big players is actually hurting the cause. I call this prison tactics: a new inmate will make sure to beat up the largest and toughest guy to make sure the others leave him in peace. Maybe successful there but annoying and pointless in the IT business.

Social media experts and entrepreneurs will tell you that it is terribly important to use your real name and have a domain with your real name and make it all very personal indeed.

I am living proof that that does not necessarily apply if your driver is to educate and help people rather than building a personal brand and make a living by giving the same workshop over and over again.

Sometimes a handle that is more technical or geeky actually gives you more credibility and makes people listen more - especially when you used it for years in other circles like old mailing lists, usenet or IRC.

## Use powerful social web sites and products

The social web is evolving almost monthly and especially right now there are a lot of companies being shut down or others emerging in weekly cycles. This is OK - it is an evolution after all. For you as an evangelist this can be very interesting:

- **Write about new social media products** - have a play with them, note your first impression.
- **Be on the lookout of new products** - if you sign up quickly you normally get invite codes (a very common practice in launches) - and having those as one of the first and saying so on Twitter is way cool.

In addition to the new kids on the block of the media space, it is very important for you to be aware of great resources that are already established, have a network of specialists and allow you to easily store and access content:

- Flickr is a photo sharing web site that allows you to store photos and screenshots. A lot of tools can talk to it via its API and it comes with a bulk uploader. Flickr is also a great resource for Creative Commons licensed photos for your presentation. Users can leave comments, tag your photos (add keywords to ease finding) and leave notes on the photo itself. Flickr also stores short videos (long photos as they call it). It is a massive community and a great way to interact with people.
- YouTube is the #1 video sharing web site in the world, run by Google. It is very easy to store video on YouTube and embed the videos back into your blog or web sites. Users can comment and tag. YouTube also has a great API to use the video and create your own custom players if you want to. The support for annotations and captions makes it very accessible and allows you to easily add extra information to video content.

- Vimeo is another video sharing site, which does the same as YouTube but has higher quality content and much better usability. It does take a while to convert video though.
- Archive.org is the Internet Archive which allows you to easily store videos, audio and pictures that you want to release to the public.
- Dopplr is a social travel site. You can store there which trips you make and find and notify people thus when you are in their area. You can also find out when people are in your city and for example invite them to the office for a talk or interview.
- Del.icio.us is a social bookmarking site - users can add notes and tags to your bookmarks and share them across networks.
- GitHub is a social code sharing network. Git is a version control system and GitHub allows you to store code there, have a Wiki to explain the information and make it easy for other developers to fork and watch your code. GitHub also comes with a nice code displayer and automatically creates archives of your code for people to download so you don't need to zip it up after each change.
- Google Code is another way to store your code for people to download. Google App Engine even allows you to store and execute your apps on Google's servers.
- SlideShare and SlideSix allow you to store presentations and offer them as Flash embeds to the world. Both allow for tagging and commenting and SlideShare even allows to add a sound file to the slides to turn them into a slidecast.
- Google Reader is not only a great RSS reader but it also allows for sharing with networks, adding notes to each other's subscriptions and tag content.
- LinkedIn is a professional network where you can find other evangelists and key people in companies you want to reach.
- Facebook I am sure I don't have to explain. Good for event organisation and contacting people quickly, less useful for photo storage because of their terms and conditions.
- Upcoming and Meetup both are social networks revolving around real-life meetings and events.
- Twitter has taken over the online world by storm. It is still re-inventing itself daily trying to define what it is, but one thing is for sure: it is an amazingly easy way to spread short information very far and very fast and a good way to stay in contact with people when you are on the go.

These are only a few of the sites that make a lot of sense to use as a developer evangelist. I will keep collecting them on delicious.

**Tip:** I found that every successful social network does not have the network as its main core idea but revolves around a thing people get emotionally attached to - pictures, videos, travel, music and so on. Pure networks hardly ever keep my attention for long.

As with anything, these resources are as useful as you make them. Good title writing, describing, annotating and tagging will make them more useful to the world and easier for you to find what you've put up there weeks ago.

## Use the web for storage, distribution and cross-promotion

The web is a network of products, documents, data and sites and to use it to its full advantage make sure to spread your content around it. Having a product in one place is great, but people have to find it

sure to spread your content around it. Having a product in one place is great, but people have to find it. If you put parts of the product in different places which were designed to host a certain content you give web surfers a lot more opportunities to find your content.

**Tip:** don't forget that social networks can be very insular - not everybody has the time and dedication to be active on lots of them. Therefore having information useful to the nature of a certain network pointing to a full product elsewhere makes you a part of the network but also gets people interested in following your link and checking out what you're talking about.

Spreading your content across different platforms has a lot of benefits:

- **Multiple points of storage** - even if a very successful blog post would kick your server off the grid (slashdot or digg effect), the information on the other platforms is still available.
- **Multiple feedback channels** - people can comment or enquire where they are happy to hang out - may it be YouTube, Flickr or Slideshare. All of these web sites spend a lot of time building close-knit communities which means more expert feedback for you.
- **Automatic conversion and hosting** - Flickr specialises in hosting and converting photos and short videos, YouTube does the same for longer videos and other specialised sites know how to tweak their servers how to convert and send their specialist data over the wire the fastest - something you don't need to think about.

**Example:** Say you have a new code solution. You can write a blog post explaining it, put screenshots on Flickr, a screencast of how to use the interface (or install the solution) on YouTube, have a presentation about the solution on SlideShare and host the code on GitHub. This quintuples the potential audience just by using all these systems in the way they were meant to be used.

Spreading the content is one part of the solution that will make this a success for you. What you need to make sure is to link all of these bits of information back to the main product. Write great descriptions for videos and screenshots, use informative tags (one of which can be the product name) and track the feedback on all the channels to be able to answer people's questions immediately. There is not much point in a multimedia resource on the web when nobody knows what it belongs to.

## Hint, tease and preview

Social media is a lot about exploration and showing off just how much better you are in finding new information and learning about new products than anybody else.

**Fact:** This is a general thing about the web - people have gigabytes of information already downloaded but rather than consuming this information we spend most of our nights hunting for more and adding to the big pile of what we already have. It is a human thing and deep in our psyche - ever since we started hogging food for the long, cold winter months that we had to stay in the cave.

You can use this to your advantage by previewing, hinting and teasing about upcoming releases. I do this a lot - not consciously to manipulate, but because I am too excited about things and prone to

release information before it can be publicly available. Instead of releasing too early - which is disastrous - I started doing some of the following:

- **Upload screenshots of upcoming products to Flickr** - this results in pretty cool comments and people tagging the photo with keywords you may not have thought of adding as content to the product docs.
- **Upload screencasts to video sharing sites** - same reason, but bigger impact. However, remember that watching a screencast expects more buy-in from the consumer than just looking at a screenshot.
- **Hint about cool stuff coming up on Twitter** - this will result in a lot of your followers directly messaging you asking for more details - and you can give out previews and insights for them to be the first to talk about it.
- **Flat out ask for beta testers** - people love poking at things before they are public. Feedback I got this way helped me on several occasions to spot sources of confusion I wasn't aware of and allowed me to fix them before release.

Once the product is out, don't forget to add the real URL to the previews you already spread on the web - that way all the late-comers will know where to go.

## Track your impact

There is not much point in putting your content out on the web without knowing what your impact is. You can track the effects of your publications in several ways:

- **Add a page counter** - the thing to read there is not really the amount of hits and get high on that but to look through the referrers. I found a lot of cool blogs in my stats. I am using StatCounter and Urchin, which comes with my MediaTemple server.
- **Search the web and social media for keywords** - Twitter search, Bloglines and Google Blog search are great for that. Personally I have some Yahoo Pipes and a few YQL scripts that aggregate several sources and publish them as an RSS feed for my reader.
- **Subscribe to comment feeds** of posts you've written or commented on.

None of this is rocket science, but it can be very powerful and teach you how successful different styles of publication are.

## Build a network

As the web is much more social these days than it used to be, it is easy for you to connect to other people. It is very easy to get up-to-date information from people on Twitter, Facebook and LinkedIn and you can give information to the right people with the right networks almost instantly.

You achieve this by contacting people, re-tweeting cool things they talk about, report problems you found with systems on Twitter and other, similar tricks that really are not rocket science.

**Example:** Sometimes being at the right time in the right mindset is everything. For example when Microsoft released their Bing API, I played with it and complained on Twitter about how hard it is to read the docs (MSDN-style frame monster until you find the “low bandwidth” version). A few seconds later I got a Twitter message from the API lead of Bing who told me what I can do to work around that issue. Since then we had a view short conversations of what I liked and disliked directly with the people that can make a change.

People who really grasp the social web are happy to answer to you once you’ve proven that you have interesting things to say and that you find and re-tweet, bookmark or blog about great content. All of this is to make our research time shorter and our lives easier. Other evangelists are by definition social people, so don’t be afraid of talking to people and pointing them to things that get you excited.

Once you’ve successfully done that you’ll find that you get good information earlier and earlier and before you know it you are the one with the invite codes to private betas and know and talk about cool new stuff that is in the making.

# Work with the conference buzz

Speaking at conferences is a great thing to do, especially as there is a massive opportunity to network with other speakers and get to know what they are up to. A lot of misunderstandings I had for example with Microsoft technology became a lot clearer after a few beers with people of the IE8 team.

Conferences also create a lot of online buzz and are a great channel to get your information out to lots and lots of people. This is to a degree dependent on the size of the conference. Smaller ones have less buzz but something like SXSW might have too much and your contribution will get lost in an avalanche of tweets of people trying to find and meet at a certain bar in Austin.

The trick is once again to come from a different point of view. In addition to following the normal marketing procedure of conferences, try to find the extra “what is in it for me?”.

## Be a part of the conference you talk at

Organizing conferences is quite a tough job so a nice thing to do is support the conference you speak at. Twitter about it, tell people you’ll be there, maybe organize a small informal breakfast or dinner meetup in the days and hours around the conference.

The main benefit of going to conferences - regardless of being a speaker or attendee - is to mingle with others and exchange thoughts, ideas and information during the breaks and before and after the event. Don’t just show up for your talk and leave - you’d miss out on most of the fun.

## Release immediately

During a conference and in the days to follow the web is a-buzz with tweets, blog posts, photos, links and all other kind of goodies with the tag of the conference. Conference organizers also start to show twitter updates live on the big screen and collect web content tagged appropriately to list on the main conference web site.

This is a great opportunity for you to get your stuff out as far and fast as possible. Have your slide deck ready on SlideShare with the right tag and put it live immediately after your talk. Twitter about it using the hashtag and add the conference tag to the links you put on social bookmarking sites and you’ll be part of the first wave of information.

The same goes for your photos. Upload them to Flickr or Facebook, tag them appropriately and people will find them as everybody checks the conference photos. Make sure to tag photos with the name of the people in them to make searching even easier.

the people in them to make searching even easier.

## Write about the conferences

Another good way to give back to the conference is to cover it in your communication channels. Write a small blog post about your session at the conference (of course), but also a general post about the conference and what you liked about it. I also give personal feedback via email to the organizers after each conference and got a lot of thanks for that.

# Additional presentation tips

The following are some tips that worked very well in the past for me. As you are not me, you might want to tweak them a bit. So here's what I do when I give presentations.

## Introduce yourself

Introducing yourself - however briefly - breaks down an initial barrier. You are not any longer this unreachable person on stage or at the head of the table - you are a normal person.

Explain why you are competent to talk about the matter at hand. Then put the ego away - people came for information, not to see you sing and dance.

## Use humour

Humour is important to keep a long presentation interesting. I like to put in things that people just don't expect - to keep both me and them on the ball.

Humour also makes things more approachable. We tend to use humour to deal with things that scare us. Furthermore humour allows for a memorable moment - it is a way of structuring and providing landmarks in your presentation.

## Build bridges to the real world

I like to bring up real world examples and comparisons. The rationale is that they make very theoretical and hard to grasp data more easy to consume for humans. Real world comparisons also allow for emotion - and emotional responses are very powerful and make us remember.

**Example:** If you for example talk about code standards and re-use of code without a proper review a good case to mention is the Ariane 5 disaster. This rocket self-destructed because it veered off its intended path 37 seconds after lift-off. The reason was re-use of the code used to launch Ariane 4, which had different flight specs. 370 million dollars were lost due to this error.

## Pace yourself

Speaking at the right pace makes you easy to understand. If you appear rushed listeners will feel uneasy.

Trying to keep up is a terrible feeling and makes us feel inadequate. So speak slowly with meaning and concentrate on pronouncing things thoroughly. **Pauses are good**. They allow listeners to take information in and digest it in the way they know best.

## Avoid "Hello World"

"Hello World" examples are easy to show. They are also useless, as they teach a syntax, but not the concept of a language or solution.

There is no personal value in "Hello World". We should teach how to solve issues and fulfill tasks. I yet have to be asked in a professional product to produce "Hello World".

It is much better to have a real production example to build upon:

- "This is what we had to create – here are the specs"
- "This is the final outcome"
- "Here's what we used to deliver this job"
- "… and here is how you can do it yourself!"

Build on top of what people are asked to do, not what you expect them to do for you.

## Be fresh

I always try to deliver fresh material. I hate re-using presentations and training material. The least I do is to bring some new, fresh angle. Check what is hot at the moment, research it and add it to the talk.

That way you show that your content is not only good but also very relevant at this moment in time.

It also means for seasoned conference attendees that you don't bore them with something you told them before and as seasoned conference attendees are also adamant bloggers and Twitter users this can only be a good thing.

# Thanks!

Thanks for reading this and hopefully I managed to give you some ideas how to approach developer evangelism.

I've been working as a full-time developer evangelist for about a year now and I have to say that although I put in about 50 hours a week it never feels like work. It is a great feeling to have about your job, wouldn't you agree?

I have to thank some people and institutions who got me on the way to be what I am today:

- The Yahoo Developer Network for giving me free reign in becoming a full-time evangelist.
- My colleagues at Yahoo for building great stuff to play with and asking me to explain things to them.
- Matrix 42 and their amazing Exceptional Learning Facilitator (ELF) training programme which made me a much better trainer.
- Paris Web, The Ajax Experience, Ajax World, Web Directions, Future of Web Apps, Flask Forum Konferenz, Accessibility 2.0, Braillenet Paris, Highland Fling, European Accessibility Forum, Fronteers, Webmaster Jam Session and all the other conferences that invited me as a speaker.
- Evolt.org, A List Apart, Digital Web, Think Vitamin, Sitepoint and all the other online magazines I got published at.
- WebAxe, Boagworld, Technikwuerze and other podcasts I was featured in.
- .net, Practical Web Design, Internet Magazin and the other print magazines that published interviews and articles.
- thelist, CSS Discuss, microformats, Google Page Speed, Twitter development, Webdesign-L and all the other mailing lists I had lots of conversations and confrontations on over the years.
- Apress, Friends of Ed, O'Reilly, Sitepoint, Packt and other publishers that either published my books or had me as a technical editor.
- Far too many individuals to mention - you know who you are, we chat on Twitter, IM and emails all the time.

If I forgot somebody or some important institution, I am sorry, but it is late and I am tired :)

Chris Heilmann