
NEW ORLEANS
DRUPALCON 2016
The Path to Becoming an Accidental Architect

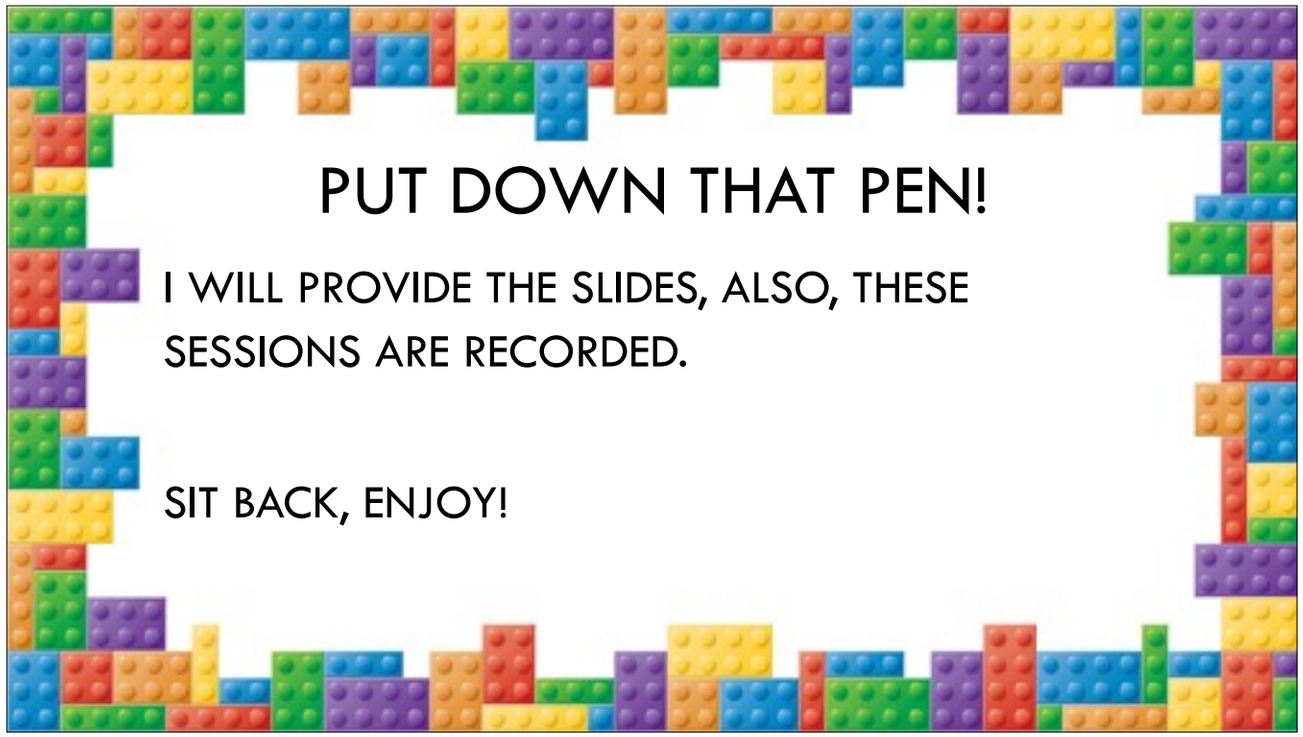
Heh, yeah, they tell you "Only if you submit them. Get cracking!"
Annnnnd, here we are :) (and a very special Thank you to Larry Garfield that friend who prompted this journey).

In case you walked into the wrong room, let me get the introductions out of the way. This session is "The Path to Becoming an Accidental Architect,"

AND I am Patrick Teglia, and I am an employee of a company called PowTec contracted as the Drupal Developer for the office of the CIO for the US Forest Service.

A little story about the title of this session. In October of 2014, I had my second interview with the Forest Service. During the interview, they had all kinds of questions, that apparently I answered to their satisfaction, which were all related to enterprise Drupal architecture. They even mentioned that they were already satisfied with my developer skills via the pre-interview steps and my first interview, and were only interested in my ideas around "enterprise architecture" topics.

I call this accidental because the position I was in at the time for NBCNews.com was not Drupal Architect, it was more of a senior Drupal Developer, and the company that was hiring me was not actually advertising Drupal Architect, they were hiring for Drupal Developer, they didn’t mention it in the job description or the first couple of calls. However certainly the Forest Service was very much interested in Drupal Architecture, and my job since then has been one filled with Drupal Architecture work and decisions.



This is probably a good time to tell you that this entire session is going to be put on my session page as a slide deck with notes, and I believe we are being recorded as well. You don’t need to take notes, just enjoy the session!

Also, I will be referring to my notes pretty often, as I have been really optimizing this session for a couple of weeks now, and it has changed enough that I really didn’t have time to memorize it.

Anyhow, let's get this started. Let me ask you a question first: “Why are we all here?”

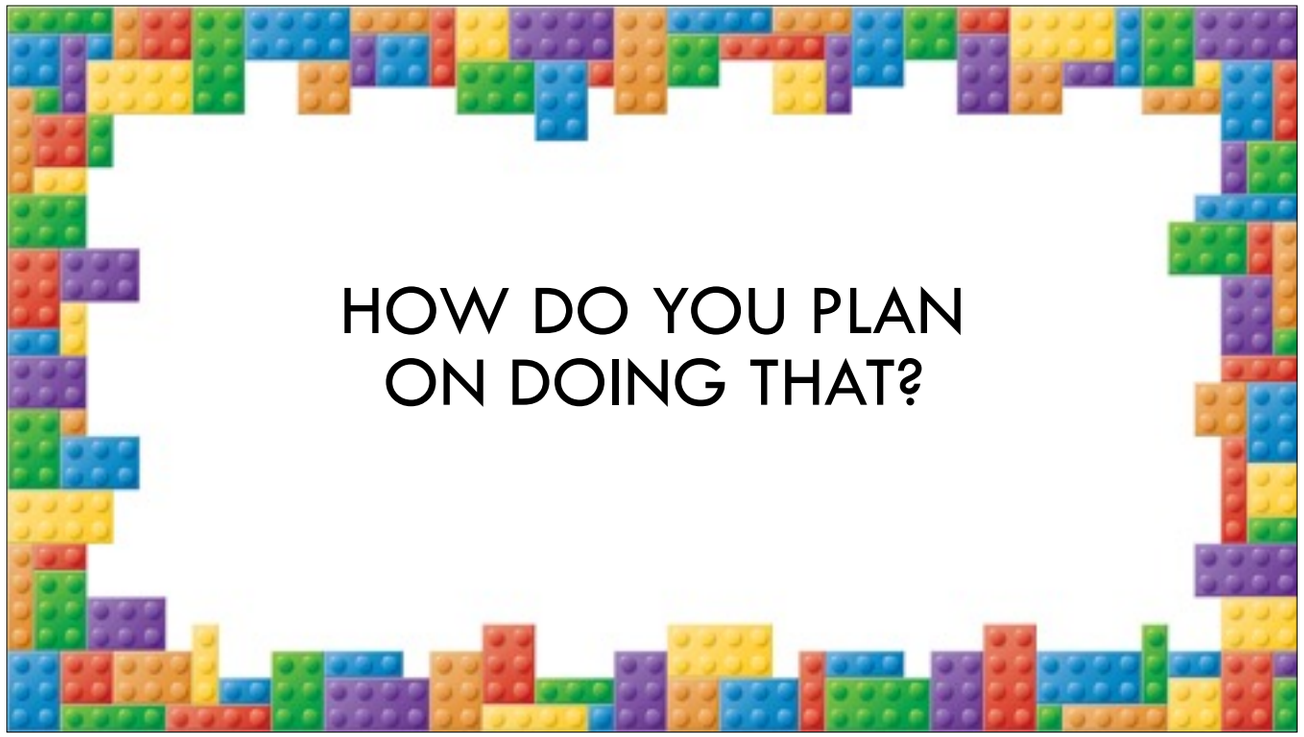
I can think of a couple of reasons.

- You are currently a developer, engineer, or lead, and you want to move into the position of architect, and you are here to figure out how.
- Maybe you are trying to figure out if you already are an architect?
- Ooorrr, you are one of my friends giving me some support or a bit of friendly heckling :)

Really though, I am thinking that most of the people here to learn how to become an architect, making the purpose of this session:

To help you build a better you so you can help your customers better.

To help you build a better you so you can help your customers better.



That sounds like a pretty stratospheric goal, to build a you that is better when you leave than when you came in. How?

We are going to do it by following the path from one of my favorite books, “The Success Principles” by Jack Canfield. His book is all about getting from where you are to where you want to be. His methodology is one that I have applied to several places in my life to pretty major success. We will use that methodology here.

So let’s break it down into a few steps:

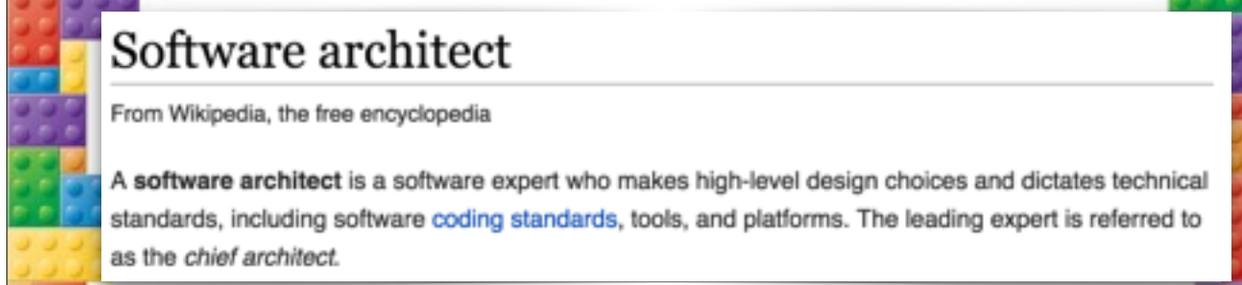
1. We'll talk first about the DEFINITIONS of "architecture" and "Drupal architecture", about what they are, how they are similar and different. This is going to cover the "Where you want to be" part.
2. We will talk about WHERE YOU ARE right now on this path, which is of course the "Where you are" part.
3. NEXT will be the largest part of the talk, the "how to get there" part. I will talk about some of the traits that make up an architect, or really, anyone who wants to truly be better at building solutions.
4. Then, we will cover some resources that I have been looking at.
5. At the end, I will let you in on some of the things that other Drupal architects have recently told me,
6. Give you an opportunity to ask me questions, and hope that I have some answers for them :)
7. If we have time left after that, maybe we will get out a little early!

We are going to start by setting up some definitions. We want to define two things, Software Architect, and Drupal Architect, so that we can compare and contrast them, and to give us a good idea of our final destination. This is the “Where you want to go” part.

The first thing to define is software architect.

This is hard to pin down. It's vague. But, as it says on Wikipedia, there are some common traits to software architecture. Let's ask Wikipedia for a definition and then separate out those traits.

WHAT IS A SOFTWARE ARCHITECT?



Software architect

From Wikipedia, the free encyclopedia

A **software architect** is a software expert who makes high-level design choices and dictates technical standards, including software coding standards, tools, and platforms. The leading expert is referred to as the *chief architect*.

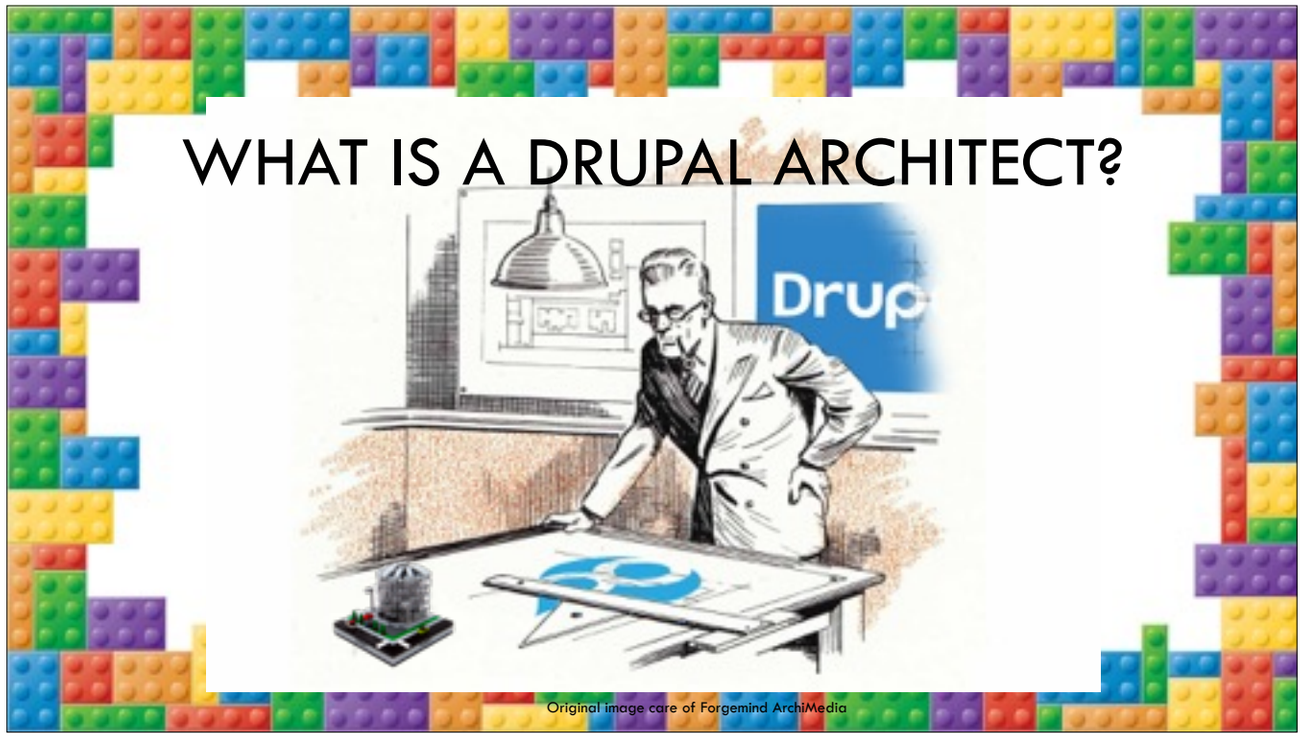
The definition they give is:
"A software architect is a software expert who makes high-level design choices and dictates technical standards, including software coding standards, tools, and platforms. The leading expert is referred to as the chief architect."

We can definitely break this down into a few key points:

- Software EXPERT - so, lots of experience
- Makes high level choices
- Dictates technical standards
- And we can infer that they need to know how to communicate all of the above

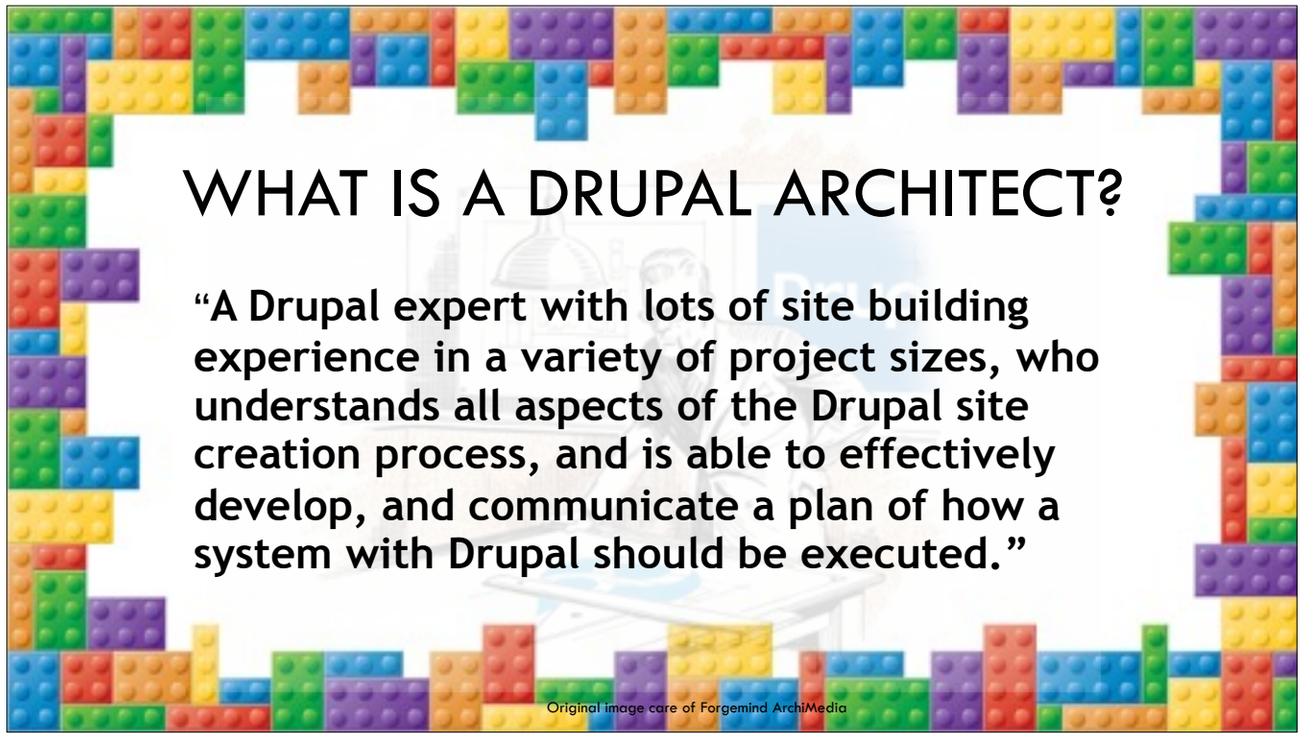
That doesn't seem too bad. We can work with this.

Obviously, beyond this definition, there is a lot to software architecture. We could have many discussions about ability to understand different architectures and patterns, understanding large codebases, reporting tools, external services and how they interact, etc.
Seems pretty obvious that the world of software architecture is a fairly broad and flexible one.

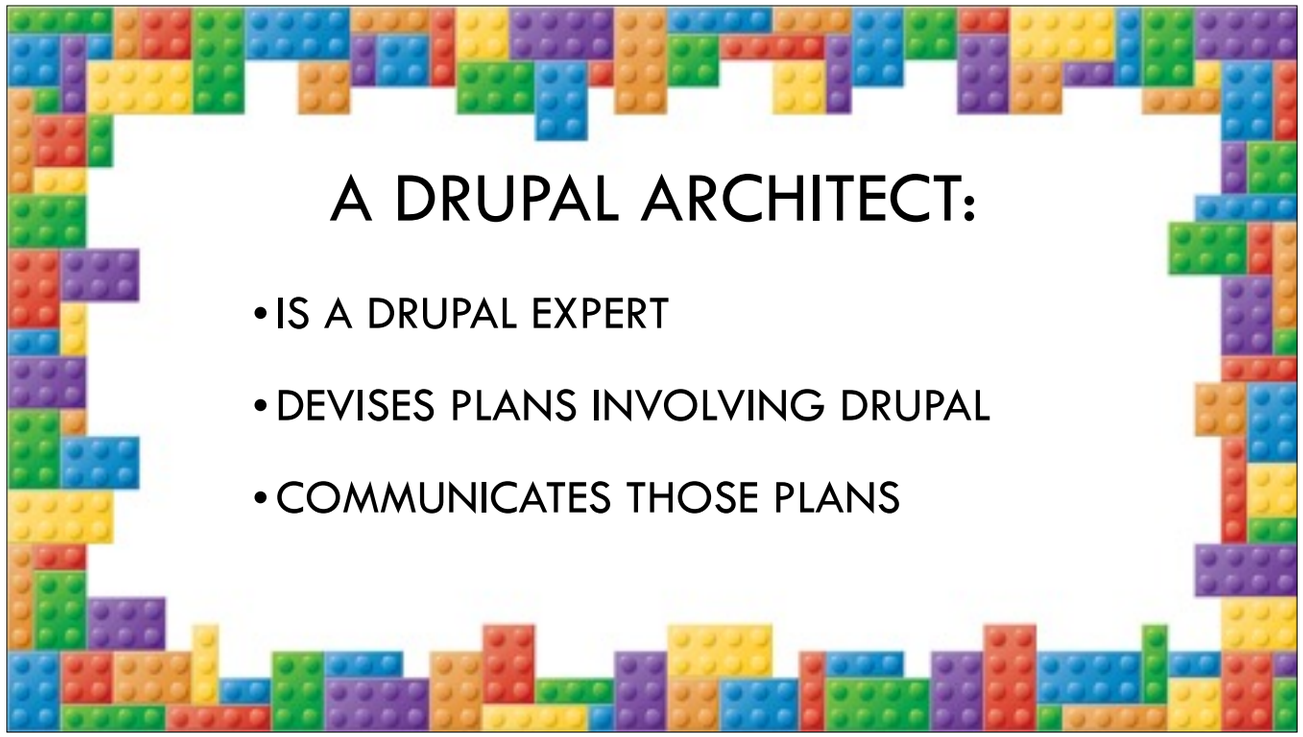


Now, let's look at our neck of the woods. What is a Drupal Architect?

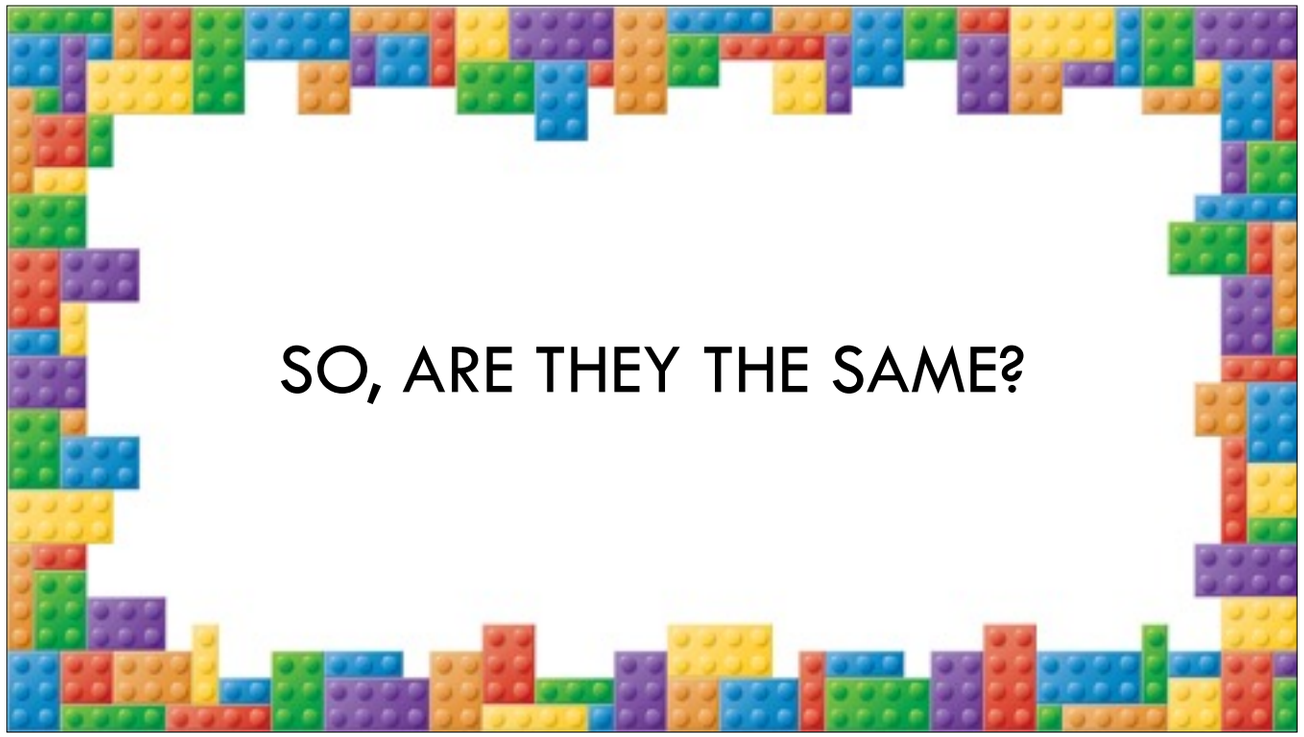
It has been pretty challenging to find the exact definition for this. I looked at quite a few job postings for Drupal Architect openings, looked all over Google, and what I have found is that this is even less defined than software architect, so I made up my own, and ran it by some friends. They didn’t object, so let’s run with it. It is:



“A Drupal expert with lots of site building experience in a variety of project sizes, who understands all aspects of the Drupal site creation process, and is able to effectively develop, and communicate a plan of how a system with Drupal should be executed.”

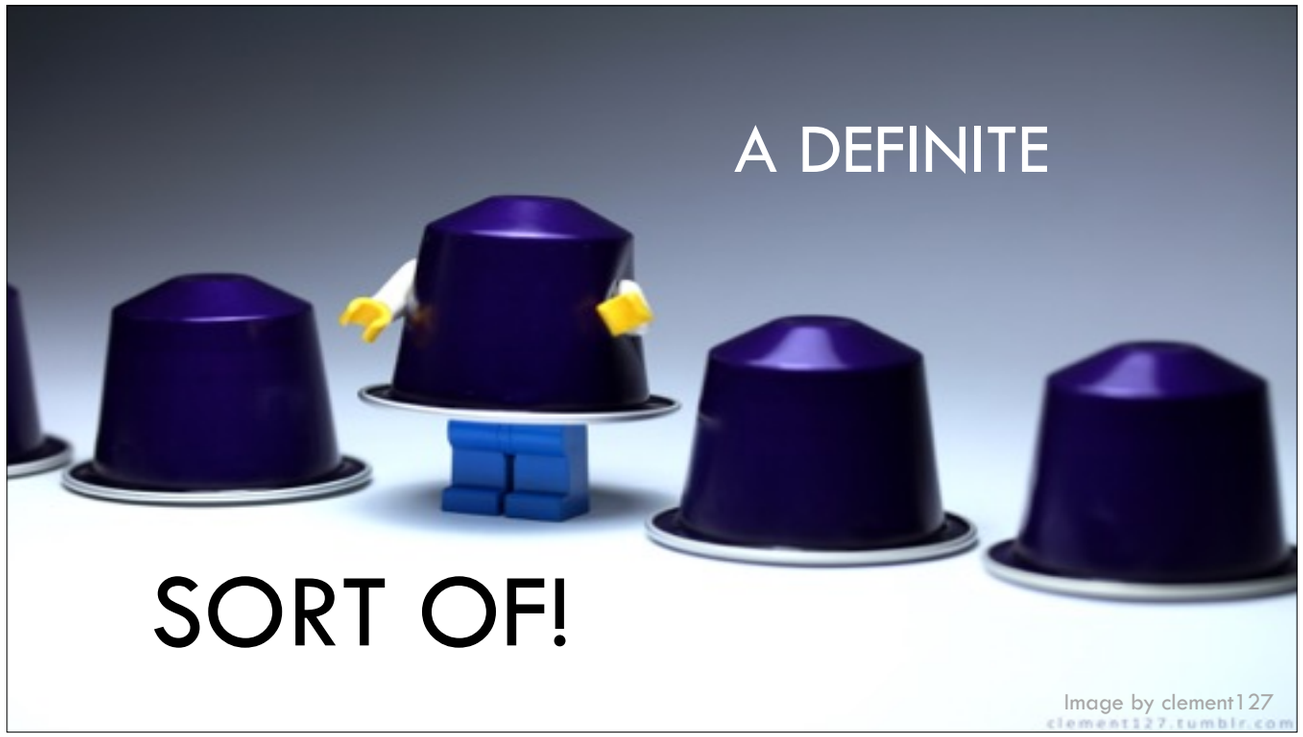


We can break down our definition into a few points as well:
Is a Drupal EXPERT.
Not just part of Drupal, like the front-end, or the back-end or making modules. They **understand** all of it. Not necessarily THE expert in ALL parts, just that they UNDERSTAND all parts.
In other words they have a significant breadth of knowledge, not necessarily an extreme technical depth of particular parts of Drupal.
Can devise a plan of things involving Drupal.
Note this doesn't say ONLY Drupal. It's things involving Drupal, so it can be Drupal PLUS other things.
Can communicates the plan they devise.



So, the big question:

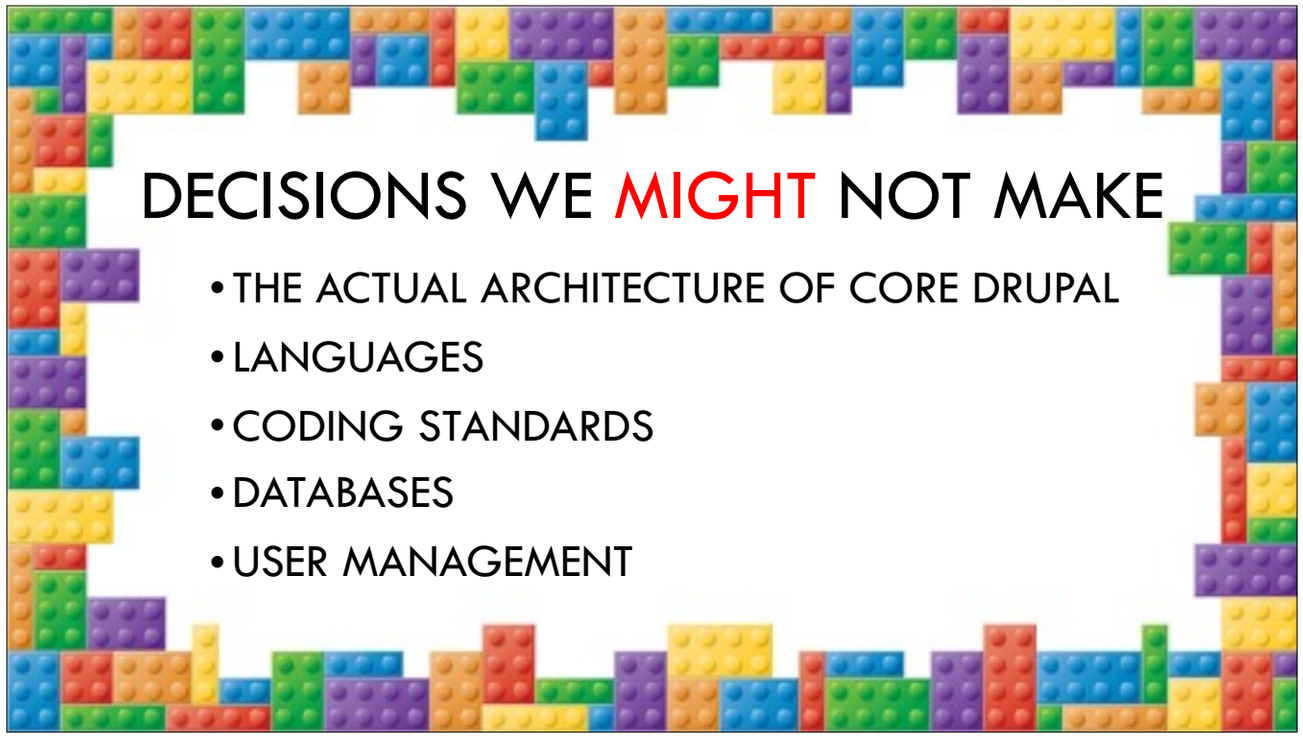
“Are they the same?”



Sure, there are tons of similarities but kind of on a more micro-scale.

- Both require expertise, experience. This one is the same.
- Both Make high level choices: In Drupal I called this "Develop a plan of a system involving Drupal”
- "Sometimes dictates technical standards”
  - In Drupal I didn’t include this in the definition, but I would say that Drupal Architects definitely have to dictate project standards, just not usually the same kind.
- "Know how to communicate all of the above”: Definitely lines up with "And communicate a comprehensive plan”

Also, the business side of things is quite similar. Architects need to get buy-in from necessary parties, have to navigate office politics, have to be able to talk business, code, systems, and translate between all of the above. They need to deal with budget and human resource constraints, and more.



Looking more closely though, we can see many differences. One of the biggest ones is the fact that many of these decisions have already been made. Often we don't have to pick language, or security type, or architecture. They're already in Drupal.

This is because Drupal has many good answers for very common questions on the topic of developing a CMS, and provided you are using it as a CMS it's going to do ok right from the start.

Though, like with so much of what I have found in architecture, there are no absolutes, so you can totally flip this around. Every one of these things is something YOU CAN override. Like, databases. A Drupal Architect want the speed of Mongo, so MySQL goes out the door.

Really though, the biggest difference I see between the two is that a Software Architect would determine the technology after a lot of investigation into the problem. In a Drupal Architect's case, we start from there, as we know that the need is "CMS" and the answer is Drupal.

NOT SO FAST THOUGH, that doesn't mean we get off scot-free, because even though Drupal makes lots of sane choices for us, we all know that we still have to think about a bunch things that software architects don't, things like:

```
Site structure (as in multi-site, domains module, etc.)                         Asset management
Search engine solutions, Solr, maybe Google Search Appliance. Commerce choices.
How to structure Content structure, entities, taxonomies.                       Will it be headless?
Panels vs. no panels                                                                    Development workflow.
Will we use features?                                                                                 Editorial workflow
What modules to be used?
How about install profiles?                                                             Caching and so so much more.
```

I think you see this list could be endless.

Architects are also going to often have to be weighing all of these options against the manpower resources they can muster and against the budgetary limits.

Taking an earlier example, possibly our Drupal Architect wanted the speed of Mongo, but it doesn't make sense to decide on a MongoDB solution when they only have MySQL knowledge in-house.

On every journey, you need at least three things; You need the place you are starting from, the place you are going to, and the mechanism that will take you from one to the other.

Having defined the Drupal architect, we now have the place we are going to. In the section after this, we talk about some of the things that get you there, but for now, let’s spend a little time on the starting point.

Where are you right now? Are you pretty green, or do you maybe have the experience and skills to be a Drupal Architect already?

I think every journey of self-improvement requires really looking at where you are so you can build a good plan for what it will take to get you to the finish line.

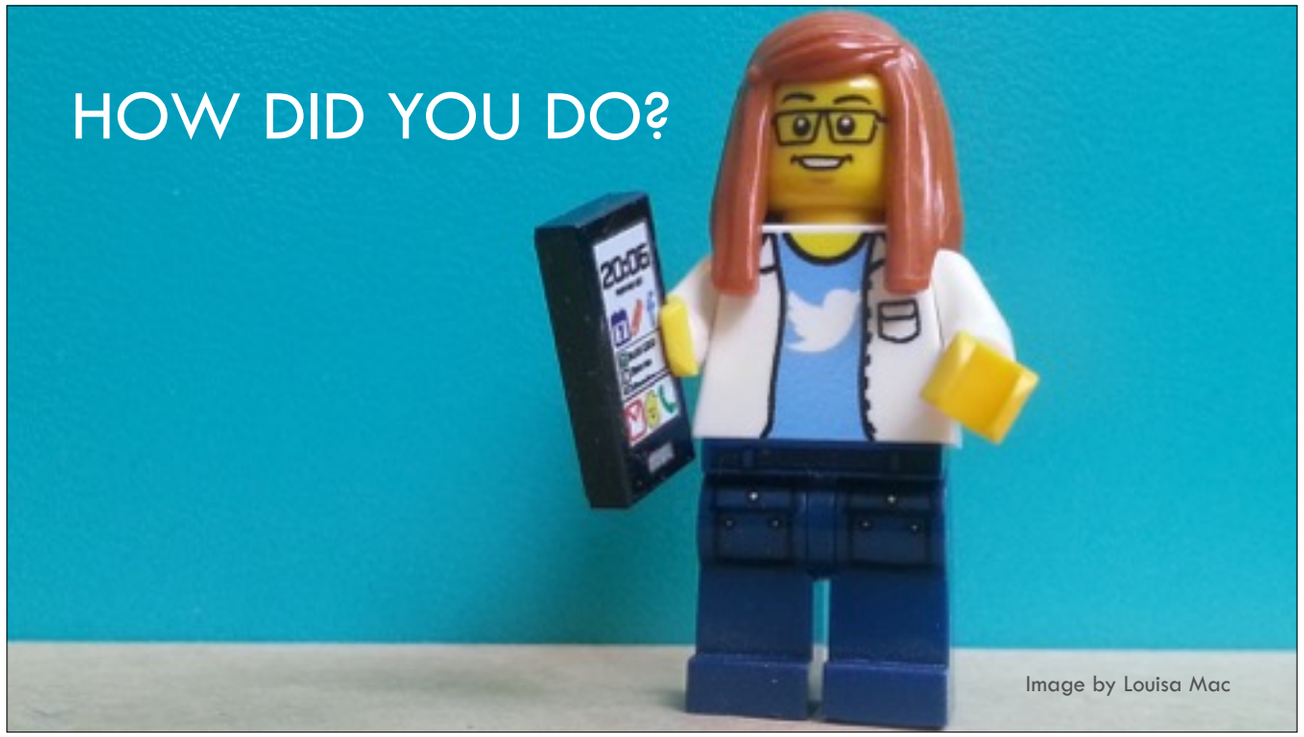
Let’s take a little test. Answer these Drupal expertise questions to yourself honestly:

1. Have you been doing this for quite some time?
2. Do you have a particular convention for organizing your Features, OR
3. Do you dislike using Features in your development workflow?
4. Do you have a preference on building themes from scratch or using base themes?
5. Have you provided core patches, and been in a long Drupal.org issue discussion?
6. Have you helped others on Drupal.org or in the #Drupal IRC channel?
7. Do others come to you first on Drupal related topics?
8. Are you starting to forget some of your old projects?
9. Have you ever said "Man I LOVE Drush!"
10. Do you wish you had a module that would remind you to clear the darned cache?

Chances are that if you relate to most of these questions, you are probably up there on your Drupal expertise. That’s half the battle right there.

These next bunch of questions are related to the softer side of things. Once again, answer these to yourself.

1. Do you have the ability to talk the "business" side of things with stakeholders?
2. Have you had to do much estimating?
3. Do you know how to work with MORE than one style of project management?
4. Are you able to easily come up with many suggestions for Drupal problems?
5. Do you feel confident in your suggestions?
6. Do your suggestions often get implemented?
7. Do you get complimented for making a technical concepts easy to understand?
8. Do you automatically multiply the amount of time you think something will take by a factor of X?
9. Are you comfortable communicating with important people?
10. Have you had projects fail? Spectacularly?
11. What did you learn? What did YOU do to cause it to fail?
12. How many times have you recommended that Drupal was NOT the right solution?



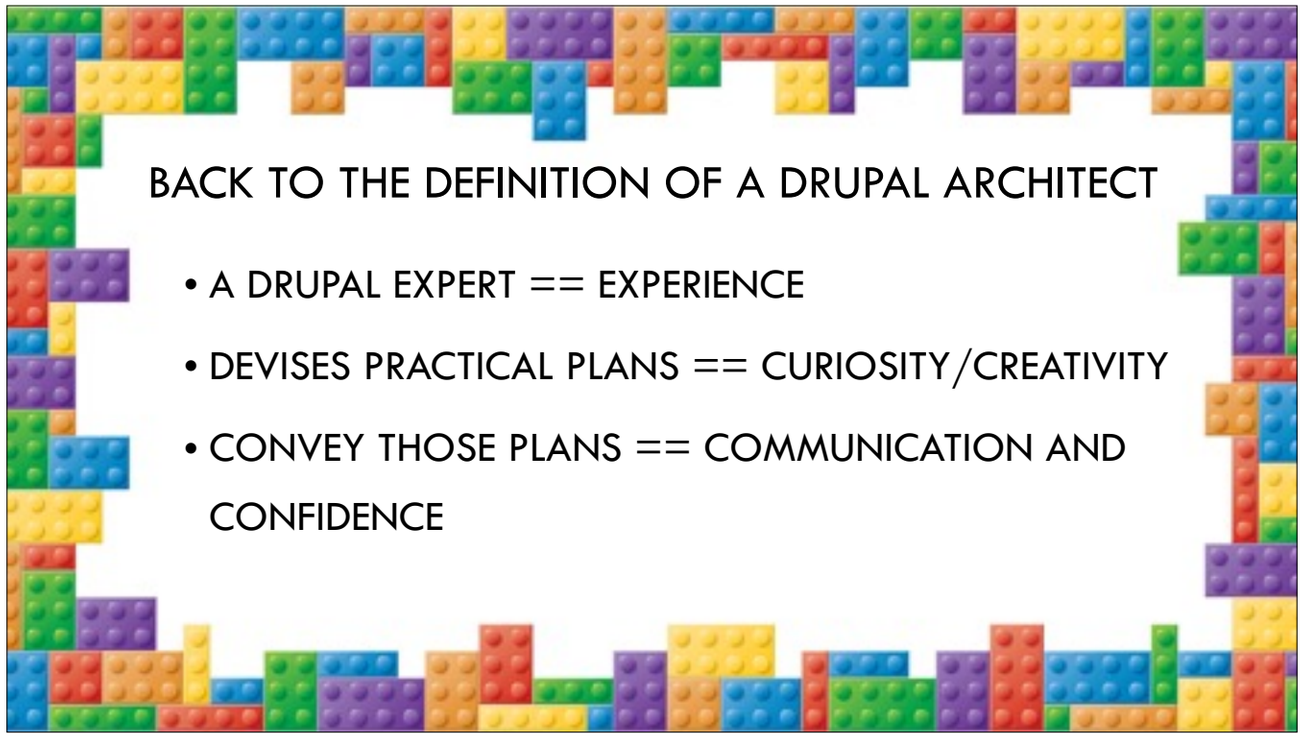
That’s it. How are you feeling about your answers?

If you were nodding your head a lot, or answering yes pretty often, there’s a good chance you probably have adopted many of the soft-skill traits needed to be a Drupal architect.

If not, PERFECT! That’s why we are here!

Now, we have talked about the goal of where you want to be, and we have talked about you, and where you are on the path to this. So what are the next steps for continuing down that path to the goal?

I think that how this works is via a mental transformation. That’s how it worked for me. If you can incorporate certain traits into your mental being, you WILL be the type of person that, given enough experience, is set to be an architect.



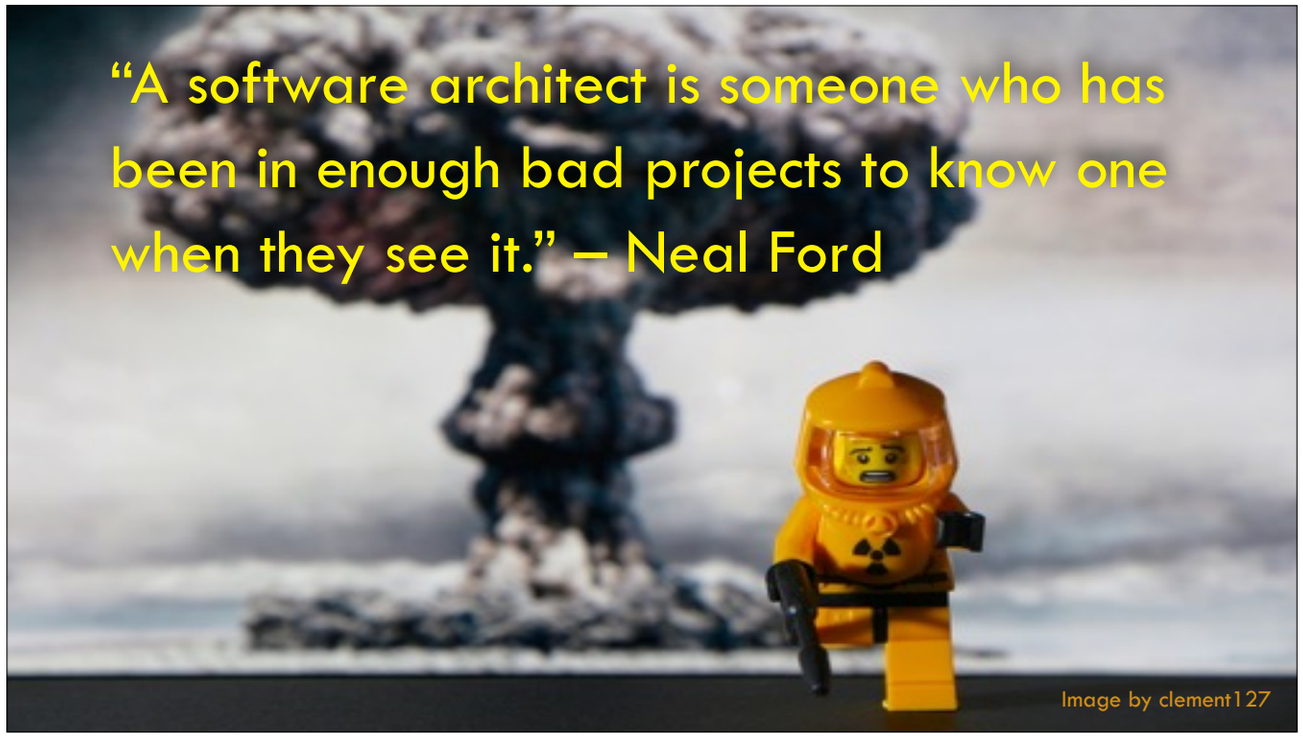
So what then are the traits of a Drupal Architect? Let's look at the definition we came up with earlier:

- Is a Drupal EXPERT. To me this says one thing, a person with a bunch of experience.
- Can devise a practical plan. To do this, I feel someone has to be fairly creative. We will talk about why I put curiosity up there in a minute.
- Can communicates the plan they devise. Pretty obviously this means they need to be able to communicate.

Outside of experience, which covers the first point, I see soft-skills being all the rest. You need to make practical yet innovative and creative plans that are good for your client's particular needs. You need to dig to find out what those needs are. After that you need to communicate the plan, often you need to confidently sell that plan.

So, the road involves creativity, curiosity, confidence, communication, combined with lots of experience

There's much more that you CAN learn, industry specific knowledge, general software architecture knowledge, etc. but this will be a very good start.



EXPERIENCE, Now, It isn’t a personality trait or a soft-skill like the rest I will talk about, but IS a requirement. In fact, barring all else, this one would do more for you than ALL the others put together.

In one of the videos I was watching on software architecture, Neal Ford said “A software architect is someone who has been in enough bad projects to know one when they see it.” It’s funny but you know it’s true, right?

Seriously though, if you really want to be a Drupal architect, experience is definitely key.

And because it is KEY, you want to get it right.

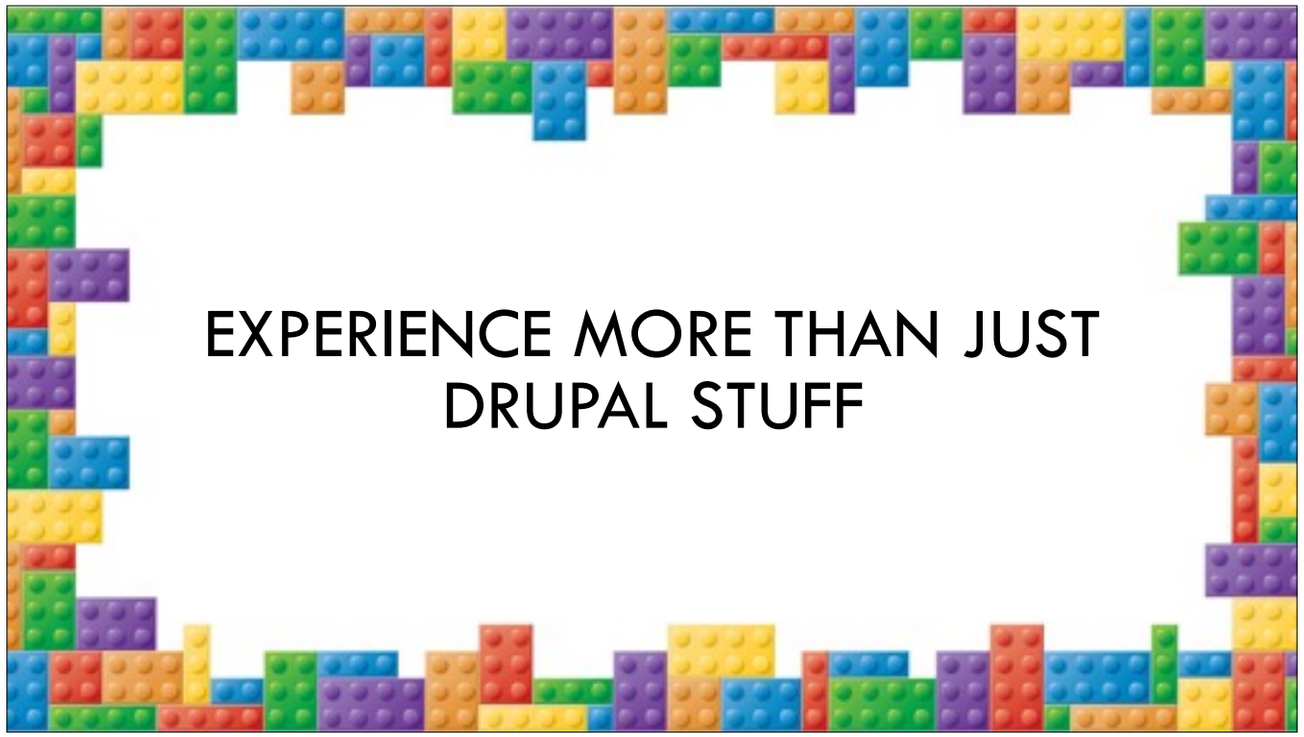
My suggestions for experience are:
1. First, wear lots of different hats. Work on every part of Drupal projects that you can. Back, front, core, contrib, everything you can. It really helps you stop seeing “every problem as a nail”.
2. Next, if you get the opportunity, work on bigger sites. On smaller site builds, there often isn’t a need for an architect. Complex business needs drive more complex technical solutions. Pretty much you wont see those needs unless you work on bigger sites.

Personally, I’ve had the opportunity to work on many projects for lots of companies and clients. The diversity was great because it gave me experience in different industries and on different sized projects.

THEN I got a fantastic opportunity to work at NBCNews.com thanks to Seth Brown at lullabot. While I was there, I got to take part in the construction of a multi-level profile build that became a basis for a couple of different sites. This thing was also headless, depended on a complicated queuing architecture as well as external media management, caching, a distributed team and more.

It was exciting, and pretty darned complicated, and the architect for that site, Chris Pucci, was really super smart. I learned so much working on that project, and from working on that team. It taught me bundles about working with people as well. It was probably the best employment experience I have ever had. I grew so much in that job.

Basically I touched on lots of things in that job, and it was big, really big, so I got to see many complicated things that needed solving.



Beyond the Drupal side of things, it pays to get to know the business side. I have two really key suggestions for this; take any opportunity you can to get documentation tasks, and if you can, get some experience estimating.

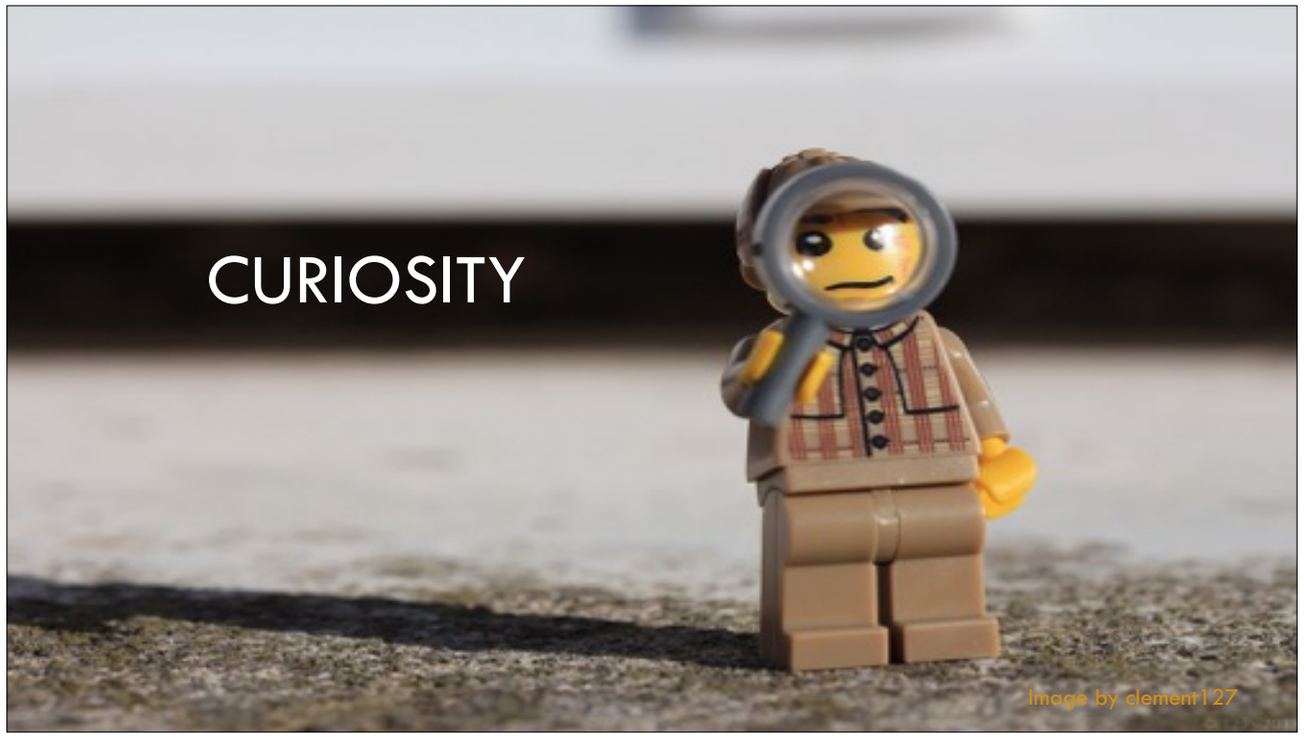
On the first one documentation, You will NOT be a good architect if you can't get your ideas out of your head and into a document that can be easily understood. It's all well and good to be able to whiteboard, or to be able to speak to people, but your ideas will need to be referred to, and you will not always be there for people to ask questions to.

On estimation, I will say that while I was at the fantastic Drupal shop Palantir, estimating was the task helped me understand how many little important details there are in a project, and how to come up with sane assumptions for many of those details, and how to write out those assumptions. There is probably no better tool I can think of to really understand the breadth of a Drupal project that having to estimate it out.

Trying to plan the structure of a site, down to the field level, so that you can estimate how much time you will spend on that project, is very good practice for the budding architect.

By the way, Palantir has always been very transparent and giving with how they manage projects, so if you are interested in learning a VERY good way to estimate projects, I can recommend none better than to look up their methods and give them a try.

Oh, lastly, almost forgot, but it doesn't hurt to get to know the current major styles of project management. If you get an opportunity, fill in the for the PM when they are sick, or have a schedule conflict, see what it's like to run a scrum if the scrum master gives you the chance (you don't have to be perfect at it). Knowing a bit more here can't hurt and certainly helps you be a better team member, when you get to know how that world works.



An important thing architects need to do is to keep up with trends in technology. I will explain more in the next section, but the gist is that you can be more creative with your solutions if you have a larger base of knowledge to draw from. Curiosity can help you expand that base of knowledge.

The thing about curiosity is that once you have developed the trait of curiosity you will find that it helps all aspects of your life, not just for helping you be more creative. If you become curious you will have a more open mind, you will find yourself becoming more observant.

Well that's great, you say, but how do you make yourself more curious? There are a couple of different things you can do.

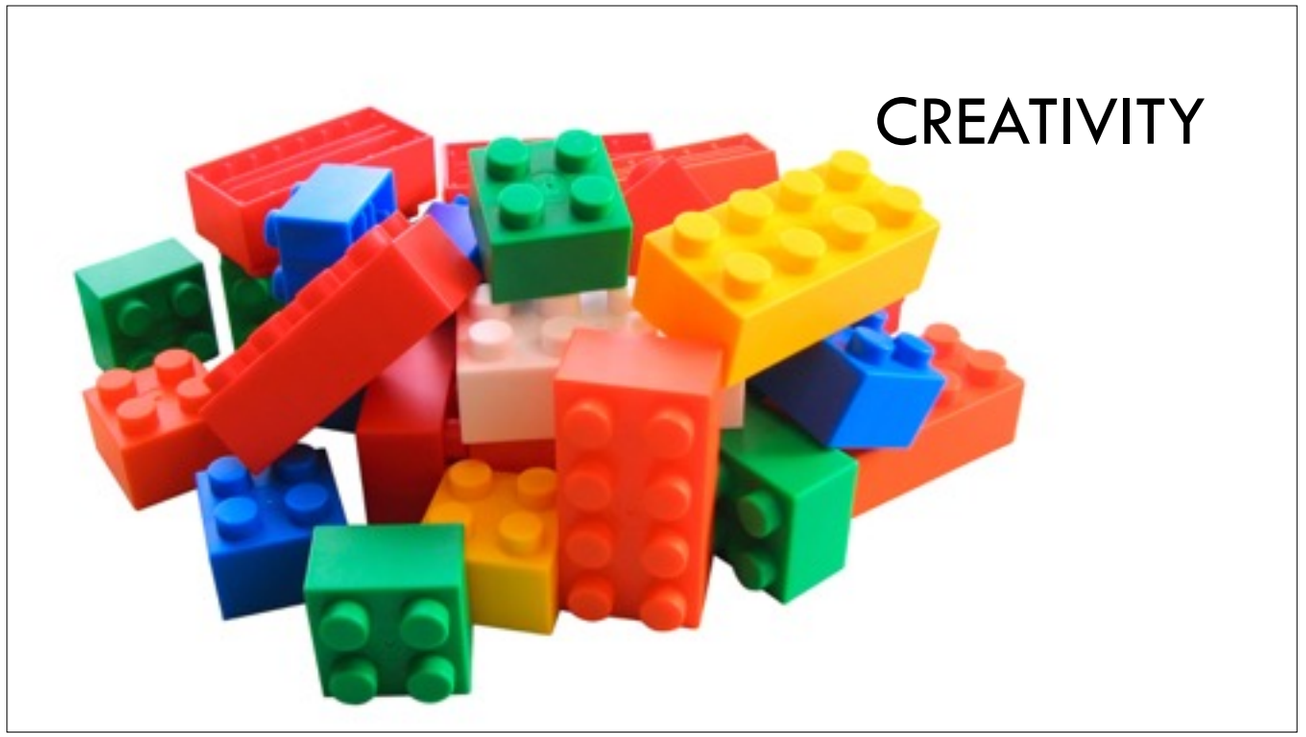
One is to follow your interests. Let yourself go off on tangents following what interests you, just to see where things take you. For example, open one of these newsletters we are always trying to unsubscribe from, and actually click a link. Or go browse slashdot, or Science Daily or some other similar place, and find the thing on the page that sounds really interesting, and go read it!

When you do this, you open new avenues. New avenues open up new ways of thinking, and help bring a fresh perspective to things.

Next, see things as fun. By this I mean try to look at learning as a fun prospect, instead of a burden. Every day there is something new to be discovered, and that's pretty exciting!

This helps to reduce the amount of stuff that you don't know you don't know. And turn it into stuff you know that you don't know, as well as sometime into stuff you actually know. Every bit of this expansion is truly gold for when you need to come up with good ideas for solutions.

Don't ever stop being curious!



A few minutes ago I spoke about how the definition translated into traits. In the definition, Curiosity, the part we just finished, wasn't mentioned at all, however, this section, Creativity, would be SO much less without it.

Creativity is the ability to imagine solutions, or the skill of thinking up new ideas.

It's a process that is similar to Legos, really. Follow along here. You have a bin of Legos, maybe you have 4 or 5 different types of pieces, you have a really basic set. You have some 4 dot bricks, some 6 dot bricks, some 8 dot bricks, maybe a couple of platforms, and some roof tile looking bricks.

What are you going to make with that?

CREATIVITY OF LIMITED CURIOSITY

Did any of you think, “I don’t know, maybe a house? Some sort of a building?”
Right! Seems pretty obvious, you are working within your constraints, you are thinking of things you can do with what you have.

BIGGER BIN OF IDEAS

Now, imagine you have a set with lots of wheels, and cool looking angled things, nifty windshield items, lots of cool sporty pieces, etc.
Now what are you thinking of building?

BIGGER BIN OF IDEAS

Many of you probably thought of a car, or perhaps a plane. See how the available materials help shape the ideas?

It’s that way with creativity. The more materials you have to choose from, the more varied the ideas are that you can come up with. This is why it pays to be curious.

When you are always out digging around in the giant Lego box of technology, you are seeing all the cool new JS libraries, looking at the latest module that Lullabot does on Module Monday, seeing what nifty development practice the Weekly Drop newsletter is highlighting, chasing down new things on Slashdot, you are DEFINITELY expanding the size of your idea Lego bin.

So, to continue the analogy just a bit further, now that you have a bigger bin because you allowed curiosity to fill it, how do you get better at coming up with better ideas of what to build with all those new pieces? You don’t want to just keep making the same airplane or house, you have WAY more things to use now!

Curiosity got you to this point, but how do you go further.

I will tell you one way that I improved my creativity, allowing me to use more of what I learn.

There is an author named James Altucher, and he recommends a pretty simple program that he calls “exercising your idea muscle”.

Yes, he calls it a skill. It’s not some talent bestowed upon special individuals, it’s something you can learn. And with diligent practice, like any skill, it can be improved.

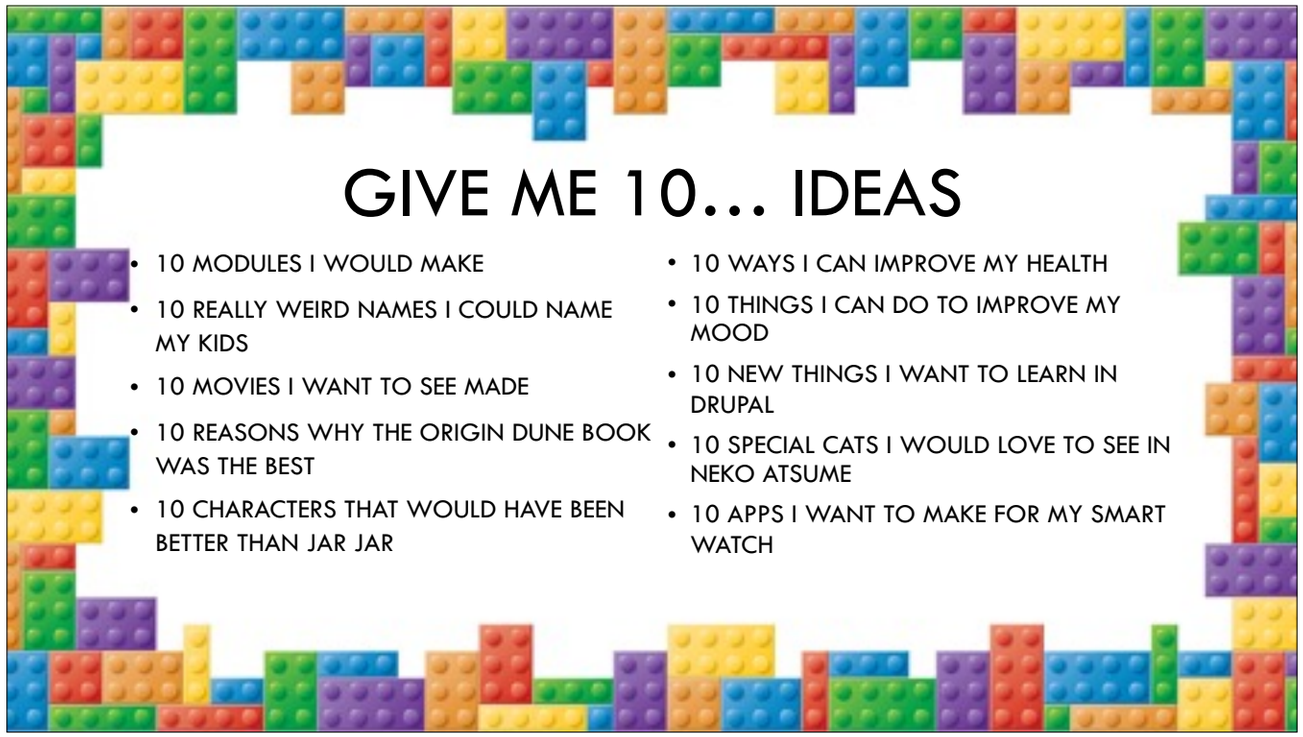
The program is seriously simple: Basically it can be boiled down to **“Exercise your creativity every day.”**

Doodle a doodle in your notepad, take a picture, re-work an old picture, color in a coloring book, do chalk art with your kids on the sidewalk, try to whistle a new tune, come up with a new way to do an old thing, optimize a small bit of code until you think it is beautiful.

Do ANYTHING creative, and YOU decide whether you think it’s creative, but I would recommend letting yourself off easy on anything you do at first. The idea is to build a habit of doing something new and creative daily.

The method he puts forward that works for me is to simply write a list of 10 ideas every day.

That’s it. The ideas DON’T have to be good, in fact he completely encourages you to write bad ones, because these aren’t supposed to be your keepers. It is simply practice for when you need to come up with ideas that matter.



Here’s an example 10 list you could start with if you are having a hard time getting started:

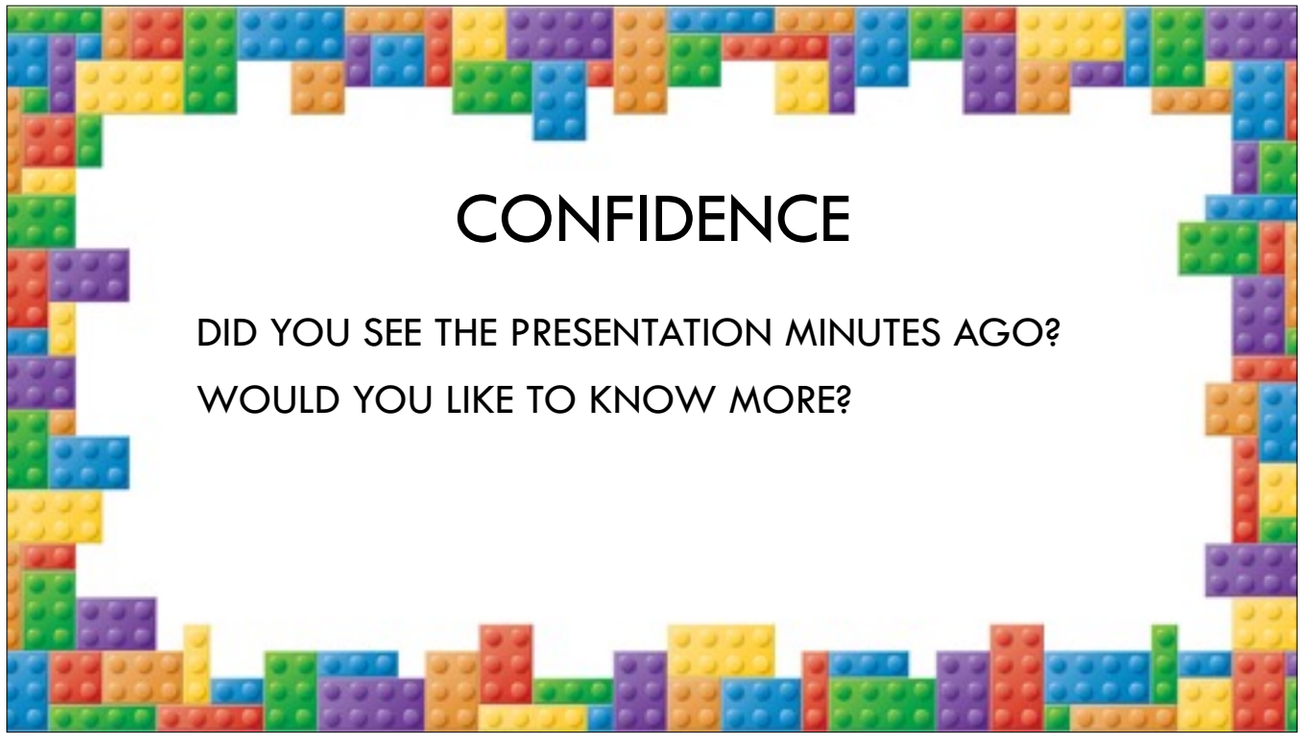
10 modules I would make
10 really weird names I could name my kids
10 movies I want to see made
10 reasons why the origin Dune book was the best
10 characters that would have been better than Jar Jar
10 ways I can improve my health
10 things I can do to improve my mood
10 new things I want to learn in Drupal
10 special cats I would love to see in Neko Atsume
10 apps I want to make for my smart watch

It’s “creativity” practice. But practice only works if you do it. Consistency is key on this. James says that you have to do it every day or your creativity muscle will atrophy, and I am with him on this. You will get better at this the longer you do it.

Go super simple on this at first, give yourself permission to REALLY suck. But don’t stop, and you will find that after time you will get very good at coming up with ideas in a pinch. It will be as easy as breathing. That’s an EXTREMELY useful skill.

(breathe)

After getting some experience, and developing curiosity and creativity, you should have a pretty good sized Lego bin from which to build your solutions, AND you will be able to think up better solutions more easily.



Ok, you’re able to come up with brilliant, pragmatic, appropriate solutions for your client. You now need to give it to them. You need to get in front of them and explain it and seem like you believe in it. Confidence helps with this.

Do you sometimes feel like you should not speak up, thinking that everyone else is smarter than you, or that it isn’t your place to contribute?

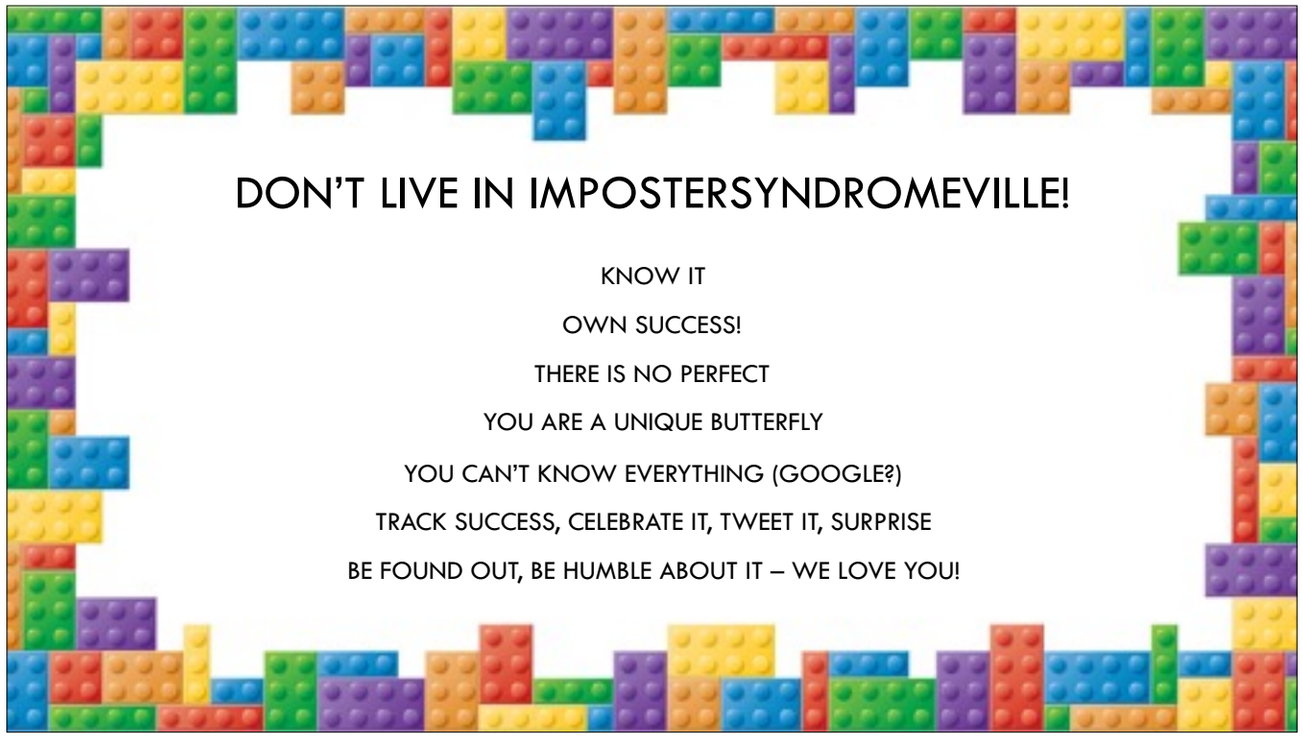
If you really don’t know your stuff, then refer back to Experience, and keep on that road for a while, but if you do know your stuff, then you have a fairly common problem known as Imposter Syndrome.

A statistic I read says over 70% of people experience this.

Before I go on here, can I get a show of hands of people who attended Dan Linn’s session prior to this on Imposter Syndrome? Show of hands of people who would like me to at least briefly go over this part?

All right then, we’ll go on!

As I said, pretty common, so don’t worry, if you are experiencing this, now is always a good time to own it, and use some tools to get past it.



Briefly then, because I know you can find the previous session on YouTube by now, here are some pretty good methods to conquer Imposter Syndrome:

- Know that you are experiencing this thing.
- Own your success. When you do something cool, great, or notable, celebrate, Know that you did that! Don’t diminish it!
- Stop thinking that there is a “Perfect” out there, and stop comparing yourself to that unattainable goal. Perfect doesn’t exist, and comparing yourself to something unattainable is a recipe for disaster.
- NO ONE is perfect. Other people may know more on some things than you, but you are the only person that knows the special blend of what you know. Understand that you passed, they HIRED you, and unless you lied your butt off when you were interviewed, you BELONG there.
- Know that you don’t know everything, accept it, understand your knowledge gap, and admit it when it comes up. Remember, you are a curious person who is constantly trying to increase your knowledge of what’s out there. That’s awesome, really, it’s ok, remember there is no perfect. Kind of the whole premise of Google.
- Keep track of the successes, publish them, blog about them, tweet them. When you find out a cool thing because you are always expanding your brain, or you figure out a new way to do something, tweet about that too! You will find that people who you thought were so much more amazingly smarter than you will say “Oh cool, I didn’t know that!” in response to your tweets.
- Quit being afraid of being found out. In fact, it is far better to be humble about your lacks than acting like you know it all. Especially in Drupal, there are so many who have been right there, and we LOVE new people here!

Outside of imposter syndrome, I have some other tips related to confidence in general.

The first of these is fitness. This is a huge one to me. Seems to be working for HIM too, he is still doing pushups!

Anyhow, I have had shoulder pain for a few years now, most likely because of the football I played in school, as well as the fact that I had gotten way out of shape, and moved from out of shape to obese. My muscles had just disappeared over the last couple of decades of computer work. This wasn't helping my mental state or emotions either.

Beyond that, I realized that my computer life wasn't helping me live a long life, and I would really like to be around for my 3 daughters and my wife for many many more years, embarrassing them into their old age.

What's more, as my wife pointed out while I was writing this, we both figured out that we weren't getting any more time every day. 3 kids, two careers, and hobbies eat up the time, so if we wanted to accomplish more, we were going to have to make more of our selves.

So, I decided to make an audacious goal; I decided to compete in the 40 and over bodybuilding competition in my home state of Idaho this time next year.

If any of you know anything about this, you probably know the work in front of me, but for those of you that don't, it means I am going to be working out 4-5 times a week in a gym for 1 to 1.5 hours a day, with the goal of converting this (point at self) into something that looks like Arnold in the first Terminator. Then, I am going to get up on a stage in a bikini bottom and do that embarrassing of my kids I was talking about ☺️

Actually, just the thought of doing that, makes getting up here seem not nearly as scary as it did.

Anyhow, from this process of working out and changing my diet, my mood, attitude, attention, and confidence have all skyrocketed.

AND My brain is reacting to this change. It has left me feeling fantastic, pretty much all the time, with the exception of leg day, and WAY more productive.

Now, I am not telling all of you to go out and start bodybuilding, **I really don't need that kind of competition**, but I do entirely think that every one of us can get more exercise and improve our diets, because we ALL know that there are a thousand studies telling you how good it is for every part of your life. I am going to stand here and agree with those studies, it _CHANGES_ your life.

We are all busy, but you CAN fit it in, and once you start, the improvement to your life is really noticeable.

Please note I am not saying diet. Don't do diets, they rebound, don't work, and make you feel worse. Instead, increase your activity level. IF it takes setting an audacious goal, do it. I know that if I don't end up in the competition in a year, I am still going to be in wayyy better shape than I was 2 months ago. I already am, so that's easy to quantify 😌 I'm already MAKING GAINZZZZZ!

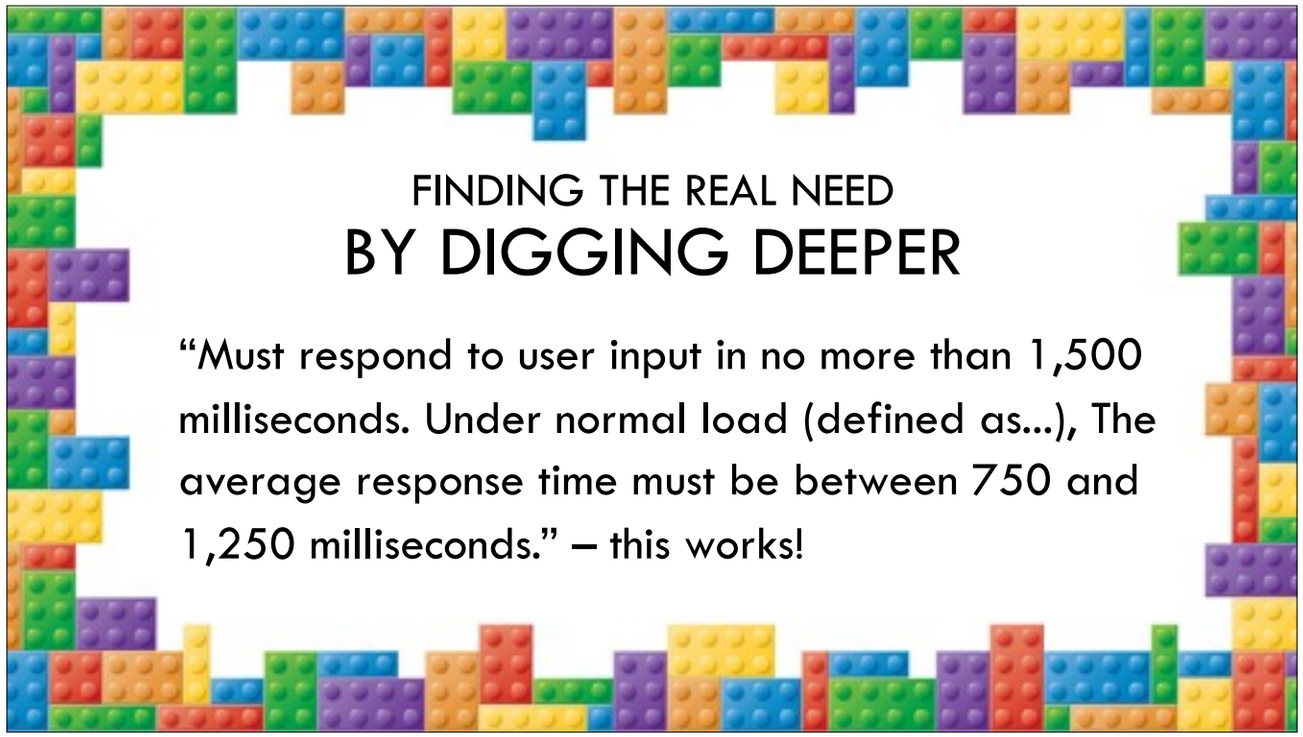
Ok, what if you need what fitness gives you, but you need it like RIGHT NOW!

You can get that! Just stand up, and assume a power pose for two minutes, you get some of these same benefits, increased endorphins, decreased cortisol, and an increased feeling of self-confidence.

Try it when you are feeling down, or before an important meeting. It will probably be the last thing you want to do, but give it a try, it works. I did it just before coming out here! It’s science. There is a great TED talk about the science of this by Amy Cuddy. Look it up, it’s fascinating.

Let’s get on to the final trait I am going to talk about today, communication. This one is sort of a combination of things: technical translation, diplomacy and salesmanship.

Nowadays, in my position at the Forest Service, I get to work with this one more and more. When there are inter-department relationships to navigate, many politically hot topics that I need to skirt around, and plenty of stakeholders that I need to convince of this or that.



Actually, before I go further on those particular parts of communication, I want to discuss something that I think may be the most important thing to an architect’s job, finding the actual need.

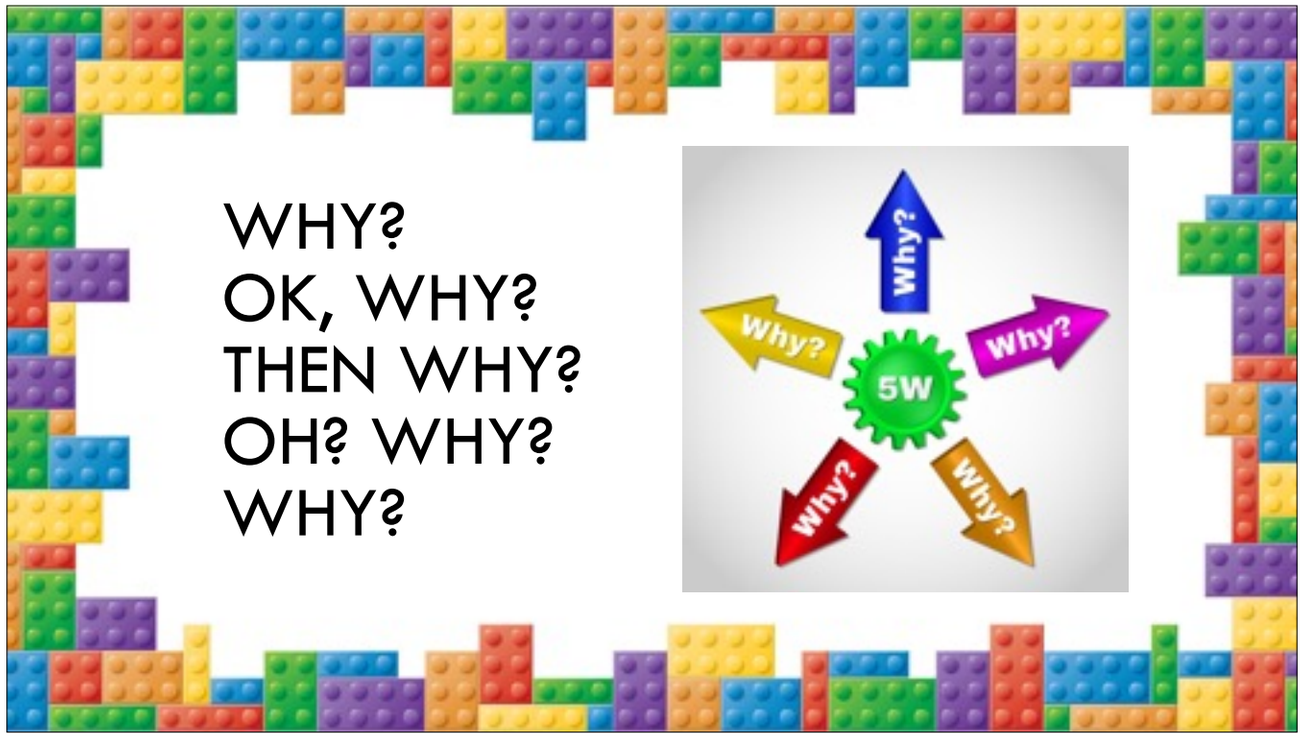
Customers often do not know the real need they have when we are initially discussing things. They sometimes want something, saw something cool somewhere else, or know they need something but can’t articulate it. This is a place where you can really be the most helpful.

First, find out if they know what they want. Can you help them describe it as a user story, with actual numbers? Can they Quantify? Too often the customer will give you vague feely sorts of words around their needs. They will say things like “flexible”, “maintainable”, “fast”, “inexpensive”.

If you leave your requirements at that, you developers won’t have what they need to build the solutions, and you will not have enough to go by to know when you have accomplished these goals.

A fantastic example of quantifying is one by Keith Braithwaite that I read in “97 Things Every Software Architect Should Know” was this:
“Must respond to user input in no more than 1,500 milliseconds. Under normal load (defined as...), the average response time must be between 750 and 1,250 milliseconds. Response times less than 500 milliseconds can’t be distinguished by the user, so we won’t pay to go below that.” *Now that’s a requirement*.

To get to the point where you get a user story that is quantifiable, you ask questions. You ask “how many”, “how fast”, “what cost”, until you get numbers on the things that matter.



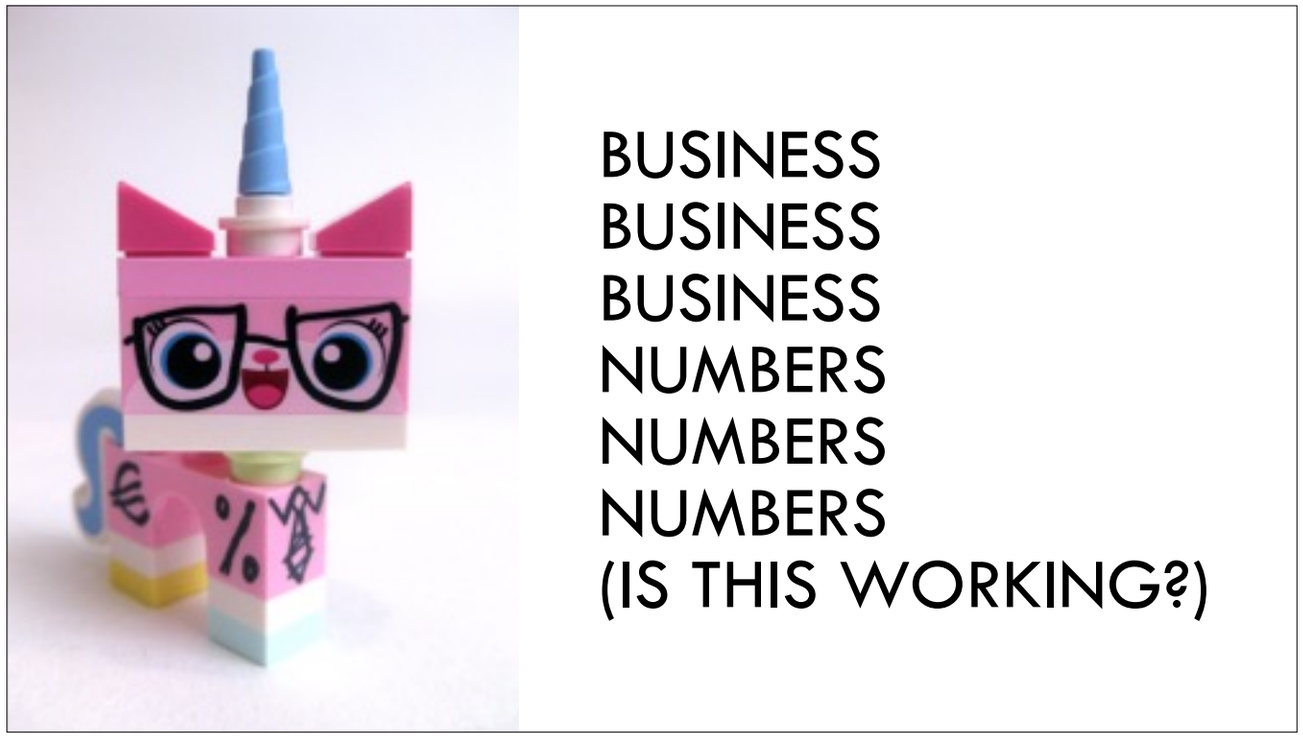
If your customer is having trouble, though, in just figuring out their needs, but they know they have a problem, perhaps the best tool to use is the tool of the 5 Whys.

The 5 Whys methodology was created by Toyota during the evolution of its manufacturing techniques and it is very useful in getting to the bottom of why there is a problem. An example listed on Wikipedia goes like this:

The vehicle will not start. (the vague problem)
Why? - The battery is dead. (first why)
Why? - The alternator is not functioning. (second why)
Why? - The alternator belt has broken. (third why)
Why? - The alternator belt was well beyond its useful service life and not replaced. (fourth why)
Why? - The vehicle was not maintained according to the recommended service schedule. (fifth why, a root cause)

You can use this pattern with pretty much any problem you come across, and you will nearly always find the root before you get to the fifth why.

I bet you can imagine plenty of scenarios in Drupal where the third or fourth why was “Because I didn’t clear cache” but really, it can be used to get to the base of why a user can’t do something, and that basis can then become the kernel of a user story to quantify and build upon.



Now let’s get back to communications.

In most of my previous roles, I had been very technically minded, I talked Drupal to people, and they talked Drupal to me.

Now, In my current role, I speak much less technically. I talk about how our department can help to support and facilitate other departments, sell others on how necessary we are, and convince them how our product can reduce drains on web budgets while simultaneously helping to increase their time utilization for content entry.

Sure, in reality what I am talking about is that I crafted a solution that utilizes a base Drupal profile and a base Bootstrap theme to allow a common look and feel across all Forest Service sites, and a selection of contrib modules that my group would support, and select custom modules and theme modifications that we had implemented on that base profile to make it easy to fire up new FS sites.

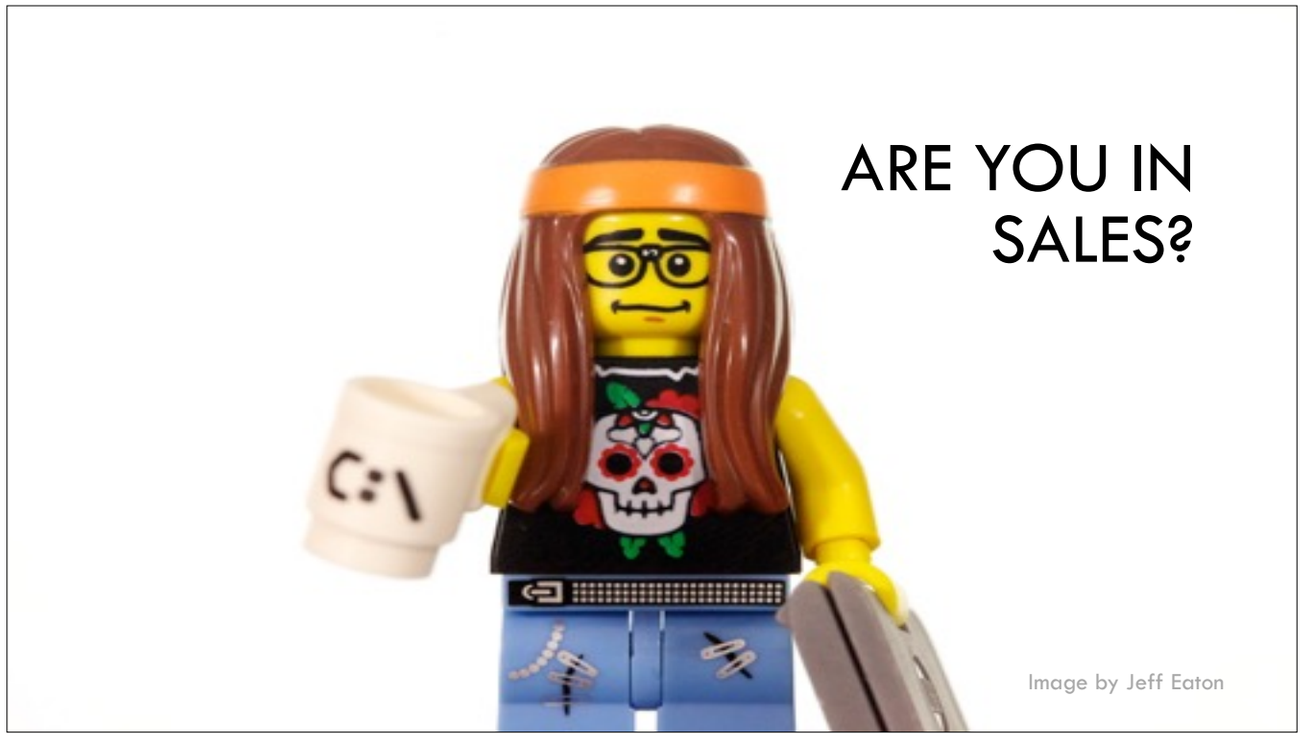
I just really don’t get to say that sort of thing as often. The people I talk to, from my awesome product stakeholder on the main FS site, to the lead of content entry, to the couple of new customers for this base profile product, even my project manager, they all don’t talk like this.

They talk about needs, desires, help, budget, use cases, sometimes about support artifacts, and documents, training, goals, project management, and estimates.

When I start talking about this module or that solution, everyone goes quiet, I know I have miss-stepped and I need to start over in the right language. I might as well been speaking Klingon.

It is a change that you must prepare for when moving from your very technical job of developer to your new job of language translator between business and development.

For this particular problem, the best advice I can give is once again become well versed I become in the business side of that world. You need to talk their language. They should not have to ever learn yours.



Once you do start talking their language, you may often need to do so in a persuasive manner. This is what I call salesmanship, because often, you have to sell your product, your solution, or even yourself as a solution provider, many things.

I just want to make one point in this section, and that is, you need to know that people buy benefits not features.

So when you are going on and on in a meeting about all the things this thing does or has, you are going down the wrong path. You should be talking about how this thing solves that problem they had, or how it decreases overhead, or how it decreases load times. These benefits are what people are looking for, not how you used Akamai or the cool things Workflow module does or whatever.

If you can express to them how this solution that you KNOW solves their need in a way they understand, by expressing it’s benefits, and not getting lost in technical jargon, you will be doing them a service, and you will be doing your job right.

Next I want to talk a little about being a bigger person. So, you are moving into this lofty position of architect. Mmmm kay? You probably feel like you might want to try go around asking for those TPS reports.

Stop right there. Yes, it’s true. You have a higher profile position, and the stakes are higher and you need things to go right. But let me tell you, the best boss isn’t the one who gets the TPS reports on time by TELLING people to go ahead and come in on Sunday.

The smart leader is the one who listens to, and trusts the people they work with.

An architect can accomplish nothing without their team, and quite frankly, you MUST leave a lot of the thinking to them. The best architects are going to be the ones trying to bring in the really super intelligent people to surround themselves with, and then not wasting that talent by micromanaging it to nothing.

Trust that they are experts in what they do, they have great big brains and can help you understand how things may work better. The bonus after all of that is that It’s sort of like getting to play with all their Legos too!

Anyhow, Yes, come up with a plan, but ALWAYS listen to the smart people on your team to make it a great plan.

Then, when your team helps you, give them credit! They deserve it, and it shows the customer that you have actual teamwork, which is far stronger than a group of individual workers.

That wraps up the body of my session.

Here, now, are some of the resources I have been into or through in the last several months.
O'Reilly's Software Architecture Fundamentals Video series
Amy Cuddy's 20–Minute TED Talk on body hacks and confidence
97 things Every Software Architect Should Know – o'reilly
Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives - Rozanski, Nick
Software Architecture in Practice (SEI Series in Software Engineering) - Len Bass, Paul Clements, Rick Kazman
The Blog of James Altucher
Success Principles, How to Get From Where You Are to Where You Want To Be – Jack Canfield

The last one, Jack Canfield's book, is one that honestly changed my life. I truly believe using those principles I could accomplish nearly anything.

EXPERT WISDOM…

Here are a few comments from some of my Drupal friends who responded to my plea for advice, straight from their tweets to your eyes.

EXPERT WISDOM…

LARRY GARFIELD @CRELL APR 10

@CRASHTEST_ ARCHITECTURE IS WHERE SOFT SKILLS AND TECH SKILLS MEET, FORNICATE, AND PRODUCE A BOUNCING BABY PRODUCT.

EXPERT WISDOM…

ANDREA RENE SOPER @ZENDOODLES MAR 24

@CRASHTEST_ REMEMBER THE SCOTTY PRINCIPLE WHEN ESTIMATING.

"MR. SCOTT, HAVE YOU ALWAYS MULTIPLIED YOUR ESTIMATES BY A FACTOR OF 4?"

https://www.youtube.com/watch?v=t9SVhg6ZENw

Mr. Scott, have you always multiplied your estimates by a factor of 4?

EXPERT WISDOM…

JBRAUER @JBRAUER MAR 24

@CRASHTEST_ 'SEEK FIRST TO UNDERSTAND' … THE 'PROBLEM' YOU'RE ASKED TO SOLVE IS RARELY THE PROBLEM THE CLIENT WANTS TO SOLVE.


EXPERT WISDOM…
LARRY GARFIELD @CRELL MAR 24
@CRASHTEST_ YOUR JOB IS TO FIGURE OUT HOW TO MAKE YOUR DEV TEAM BORED, BY NOT NEEDING ANY CUSTOM CODE.


EXPERT WISDOM…
ANDREA RENE SOPER @ZENDOODLES MAR 24
@CRASHTEST_ WHEN DEFINING REQUIREMENTS, NEGOTIATE FOR THE 95% SOLUTION YOU CAN DO W/ 5% OF THE EFFORT.

EXPERT WISDOM…

LARRY GARFIELD @CRELL MAR 24

@CRASHTEST_ SOMETIMES A SMALL CHANGE TO THE DESIGN CAN SAVE 40+ HOURS OF WORK. YOUR JOB IS TO HELP THE CLIENT MAKE AN INFORMED CALL ON THAT.

EXPERT WISDOM...

EARL MILES @MERLINOFCHAOS MAR 24

@CRASHTEST_ LEARN THE #DRUPAL WAY BUT DON'T BE SLAVISH TO IT. IT'S NOT ALWAYS RIGHT, SOMETIMES IT'S JUST HISTORY.

EXPERT WISDOM...

LARRY GARFIELD @CRELL MAR 24

@CRASHTEST_ GO WITH THE GRAIN. DRUPAL HAS A VERY STRONG GRAIN. IT'S RIGHT MORE OFTEN THAN YOU'D THINK (BUT NOT ALWAYS).

EXPERT WISDOM…

DAMIEN MCKENNA @DAMIENMCKENNA MAR 24

@CRASHTEST_ WORK WITH PMS TO GET A BETTER HANDLE ON ESTIMATING.

EXPERT WISDOM…

KAROLY NEGYESI (CHX) MAY 1

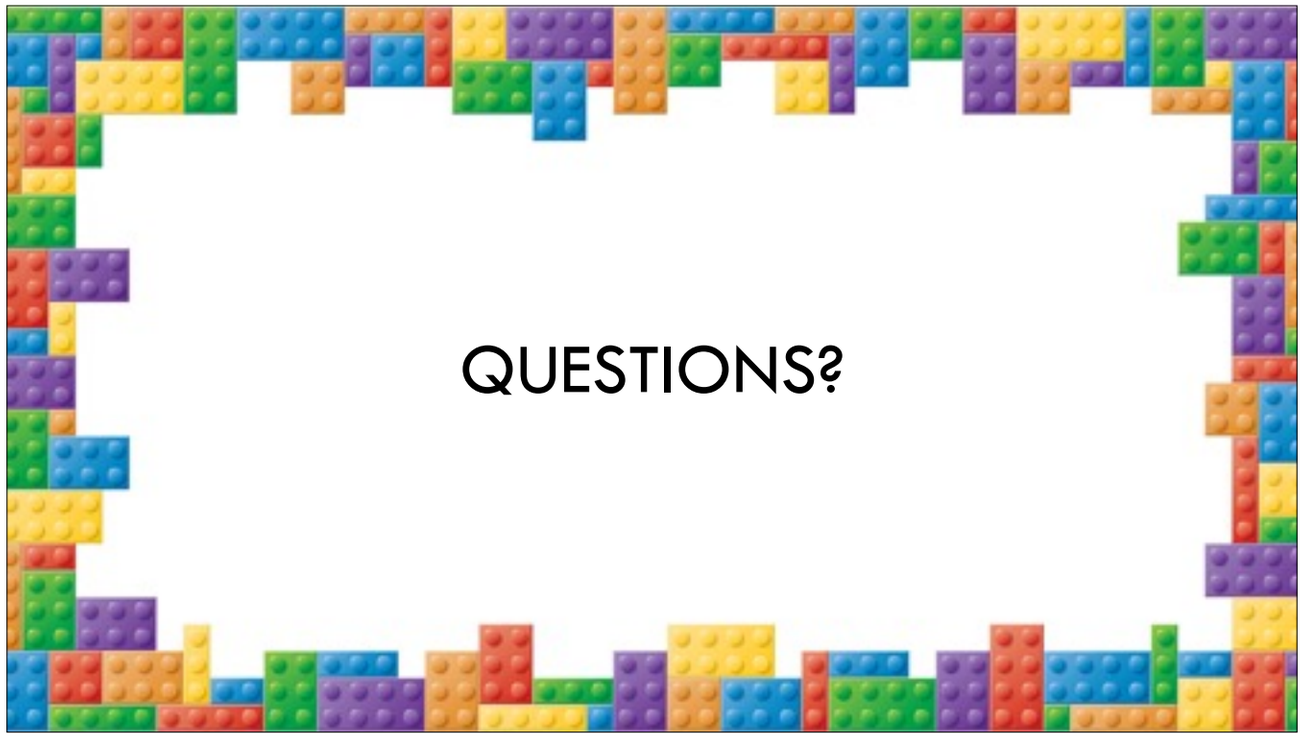
“EXPERIENCE IS I HAVE ALREADY MADE THE MISTAKE YOU ARE ABOUT TO MAKE :D”

EXPERT WISDOM…

JOSH BRAUER ON “HIRING PEOPLE FAR MORE QUALIFIED THAN ONE’S SELF”

“MY GOAL IS ALWAYS TO BE THE DUMBEST GUY IN THE ROOM ☺️”

“(usually not hard in my case but we’ll skip that for the moment) … only by surrounding myself with much stronger developers can one put together the best solutions (along the creativity section lines IMO)”



So, it looks like we have a couple of minutes for questions, and then we will all get out of here!

Awesome!
I want to Thank you all for coming, and please remember to go out and vote on your sessions. Please leave comments so that I know what parts need improving or extending or removing!

I appreciate all your time today, have a great CON!