DogFoodCon 2019

By Andrew Hinkle



TODO: Intro

DogFood Metrics

Fusion Alliance Azure day camp overlap with DogFoodCon

# Session Categories

First off I do not envy the organizers in developing a schedule. Presenters have limited availability due to work, life, travel, other conferences, and a multitude of other concerns. Sessions by category need to be spread out across the time slots, so attendees of all different walks of life can find a session in each time slot that interests them. Compromises have to be made. Some overlap happens or the sessions may not happen at all. Of course [bowls of M&Ms must be supplied without the brown ones](https://www.insider.com/van-halen-brown-m-ms-contract-2016-9) or the presenters start flipping tables. At least I assume that's what they have stored in the presenter's only room. I tip my virtual hat in thanks and appreciation for all the efforts of the organizers to make an event of this size work so well.

DogFoodCon had sessions broken down by categories for the following approximate time slots. I can deduce that the categories definitely helped the organizers setup the schedule well. There's a fairly even distribution amongst the categories across the time slots with just a few outliers. [Modern Workplace] category was for MS Flow which doesn't flow well into any other category (pun intended). Usually, there's more than just one [Front End/UX] session, so I get the outlier here too.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Category | Thursday | | | | | Friday | | | | Total |
|  | 10AM | 11AM | 1PM | 2PM | 3PM | 10AM | 11AM | 1PM | 2PM |  |
| [AI/ML] | 1 |  | 1 | 1 |  | 1 | 1 | 1 |  | 6 |
| [AppDev] | 4 | 3 | 4 | 5 | 3 | 1 | 2 | 3 | 2 | 27 |
| [Azure/Cloud] | 1 | 2 | 1 |  | 1 | 2 | 1 |  |  | 8 |
| [DevOps] |  |  |  |  | 1 | 1 | 1 |  |  | 3 |
| [Front End/UX] |  |  |  |  |  |  |  | 1 |  | 1 |
| [Human Skills] |  | 2 |  | 1 | 2 |  | 1 | 1 | 2 | 9 |
| [Infrastructure/  Security] | 1 |  |  |  |  | 2 | 2 | 1 | 2 | 8 |
| [Modern Workplace] |  |  |  |  |  |  |  | 1 |  | 1 |
| [Office 365] | 1 | 1 | 1 | 1 | 2 |  | 1 |  |  | 7 |
| [SQL/BI] |  | 1 | 1 |  |  | 1 |  |  | 1 | 4 |
| Total | 8 | 9 | 8 | 8 | 9 | 8 | 9 | 8 | 7 |  |

Sometimes sessions could be in multiple categories and a decision must be made. [AppDev] had 3 times the number of sessions as the rest. Could this category be broken up?

A colleague of mine mentioned he would have attended some of the sessions I attended if he knew they were more on the Agile, Scrum, and other related project management topics. The sessions on "Example Mapping ", "Software Requirements", and "Iterative Development" may have been categorized as [Project Management] to break up [AppDev] some more and could have garnered his attention.

The session "Usability Starts With Accessibility" could have been in the [Front End/UX] category. The sessions on Vue could have been added to that category as well. I'm sure we could break it up more, but this is enough to understand this point of view.

Breaking up the [AppDev] category into more/other categories could have helped the organizers better plan the schedule. However, with a quick analysis of the sessions with my own makeshift adjustments to the categories the distribution is pretty good given all the other considerations mentioned at the beginning of the article.

My colleague didn't review the session descriptions based on the category name. Perhaps this is just anecdotal. Did you use the categories to determine if you should even read the session description?

# Keynote – Day 1 – Keith Elder

Keith delivered a passionate presentation that left me with choking back tears. I highly recommend listening to Keith speak especially on the ISMs and his experience at Quicken Loans.

He described how he started at [Quicken Loans](https://www.quickenloans.com) with the expectation of staying for just a few months. Overtime he came to love the culture based on ISM's and the people until he could throw away his resume because there was nowhere else he wanted to be. The ISM's promote an employee's happiness, stability, and work life balance. Here are just a few of them.

1. Numbers and money follow; they do not lead
   1. Develop your skills so you may create great products and innovations.
   2. Money will follow.
2. Yes before no
   1. It doesn't mean that every idea will be approved
   2. Be open to new ideas and give them a chance
   3. I love the "KNOW" before "NO" attitude
3. We'll eat our own dog food
   1. I thought this ISM was so appropriate given we were at [DogFoodCon](https://dogfoodcon.com/)
   2. Take advantage of all the opportunities within your company
   3. If your products are good enough for your clients, then they are good enough for you

Quicken Loans dedicates one week a quarter (every 10 weeks) to "Hack Week" dedicated to innovation and improving business. They treat it like a science fair. Some of these excursions have led to new companies forming. There is plenty to learn from this company's cultural ISMs that every business should incorporate.

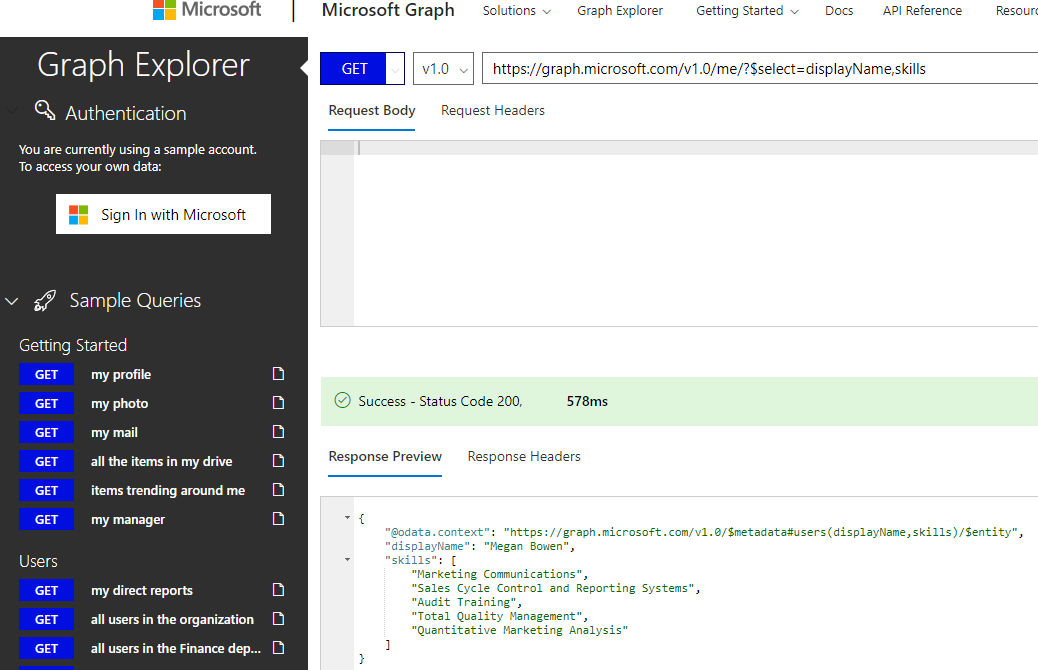
# [Office 365] Getting Up to Speed with Microsoft Graph Development – Brian T. Jackett

[Microsoft Graph](https://developer.microsoft.com/en-us/graph) is a unified REST API for O365, Windows, Azure AD, and more. You may [register your Enterprise applications](https://docs.microsoft.com/en-us/graph/auth-register-app-v2) to work with Microsoft Graph as well. It supports standard Http Verbs: GET | POST | PATCH |PUT | DELETE. The url format follows this format:

* /{Resource}/{Entity}/{Property}/{Query}
* /users/james/events?top=5

When you login to the [Graph Explorer](https://developer.microsoft.com/en-us/graph/graph-explorer) with your MSDN account you'll grant it permission to access a lot of your information. Once logged in you'll have access to plenty of sample queries to learn more about you and your company. Here's an example using the sample account.

* https://graph.microsoft.com/v1.0/me/?$select=displayName,skills



Microsoft Graph uses [Azure AD Identity](https://docs.microsoft.com/en-us/graph/azuread-identity-access-management-concept-overview). There are two types of permissions: [Delegated and Application permissions](https://docs.microsoft.com/en-us/graph/auth/auth-concepts#microsoft-graph-permissions). Be careful to only grant the appropriate level of permissions for the users and API. As an example and Application Permission follows the format: Resource.Operation.Constraint. If User.Read.All was applied to a user to access email then that user would have access to not just their emails, but everyone's email. [OAuth 2.0 authentication](https://docs.microsoft.com/en-us/azure/active-directory/develop/v1-protocols-oauth-code) is also supported.

Implement the Microsoft.Graph Nuget package in your application to assist making calls to the Microsoft Graph service. The Microsoft.Graph.Auth NuGet package (preview) supports the OAuth 2.0 functionality.

Learn more by reading an [Overview of Microsoft Graph](https://docs.microsoft.com/en-us/graph/overview).

Brian did a nice job reviewing the topic. He went over the concepts very well for an introductory session. I would have liked to have learned more about adding an Enterprise application to Graph to handle the authentication if it could be done in a few more minutes. I can see it being a follow-up session depending on its complexity. Good job!

# [AppDev] Clean Architecture with ASP.NET Core – Steve Smith (Ardalis)

Steve Smith has written an [Architecture eBook](https://ardalis.com/architecture-ebook) for Microsoft titled "[Architecting Modern Web Applications with ASP.NET Core and Microsoft Azure](https://www.microsoft.com/net/download/thank-you/aspnet-ebook)" available for download for free.

Steve ran through the standard [SOLID](https://en.wikipedia.org/wiki/SOLID) principles including [DRY](https://en.wikipedia.org/wiki/Don%27t_repeat_yourself) and [New is Glue](https://ardalis.com/new-is-glue). He went over previous architectures such as [Hexagonal Architecture](https://en.wikipedia.org/wiki/Hexagonal_architecture_(software)), [Onion Architecture](https://jeffreypalermo.com/2008/07/the-onion-architecture-part-1/), and [N-tier architecture](https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/n-tier). All of these concepts are integral to understanding Clean Architecture. I liked [Uncle Bob's book on Clean Architecture](https://www.amazon.com/Clean-Architecture-Craftsmans-Software-Structure/dp/0134494164) so I already had the basics coming into the session, but weren't necessary as Stev0e walked us through the concepts of [Domain Centric Design](https://enterprisecraftsmanship.com/posts/domain-centric-vs-data-centric-approaches/).

[The Core Project](https://github.com/ardalis/CleanArchitecture#the-core-project) contains your entities, aggregates, DTOs, interfaces, etc. This project should not depend on any of the other projects in the solution. On the other hand the other projects may depend on it. Any objects shared by multiple solutions should be isolated in a Shared Kernel and published/referenced as a NuGet package. From my experience the Shared Kernel should not change often, else you run into what my team calls the NuGet Churn.

Side note on this Anti-Pattern I've dubbed the NuGet Churn. The NuGet Churn occurs when you make a change to a NuGet package and now have to update the NuGet version on many applications dependent on the NuGet package or the application can't speak with some or all of its services. This is usually the result of a monolithic application that has been broken down into micro services or an attempt at following DDD where the dependencies have not yet been inverted. It's a maintenance nightmare that requires redeployment of all apps dependent on the NuGet package and running tests to validate the deployment. There's a longer and more harried experience on this Anti-Pattern that deserves its own article.

[The Infrastructure Project](https://github.com/ardalis/CleanArchitecture#the-infrastructure-project) contains most of the applications dependencies on external resources such as ORMs like Entity Framework, file access, external API calls, etc. Use dependency injection to these instances via constructor injection.

[The Web Project](https://github.com/ardalis/CleanArchitecture#the-web-project), UI, Console, etc. is the entry point into the application. It depends on your IoC container and the rest is injected as necessary.

[The Test Projects](https://github.com/ardalis/CleanArchitecture#the-test-projects) should correspond with each of the other projects to unit test your projects.

Check out his example [Clean Architecture](https://github.com/ardalis/CleanArchitecture) github for more details. I still prefer the Feature Project pattern, but that's another article on its own too.

I have always enjoyed attending Steve Smith's sessions. His presentation style is top notch which is why I consume his blogs, podcasts, and PluralSight videos. Great job as always!

# [AppDev] Example Mapping: The New Three Amigos – Thomas Haver

Creating a user story that everyone understands is difficult unless you include the [Three Amigos](https://www.agilealliance.org/glossary/three-amigos/) in the process. The Three Amigos represents the perspectives of Business, Development, and Testing. Usually this means one person per perspective though that's flexible the goal is to strike the balance between no collaboration and too much.

[Example Mapping](https://cucumber.io/blog/example-mapping-introduction/) builds upon the Three Amigos concept. Get stacks of index cards of four different colors. Yellow for User Stories, Blue for Rules, Green for Examples, and Red for Questions.

Start by defining a User Story using the pattern: As a [] I want [] So That []. Next fill out your Acceptance Criteria as Rules. Create Examples for each rule. Write any questions that nobody in the current group can answer.

1. User Story: As a Google User, I want to type my search string and click search so that I will get a list of links matching my search.
   1. Rule: Search results are empty
      1. Example: /////////////////////
         1. Question: Do we just tell the user no results were found or should we provide suggestions?
   2. Rule: Search results return a list
      1. Example: Dog Food Con
         1. A link to [DogFoodCon](https://dogfoodcon.com/) was returned

Context is king as was demonstrated in the session by asking us to exchange our examples with another team. We were tasked with trying to come up with additional examples that would work for their rules. Based on their response we had to write up what we thought their rules were. The reverse is easily true that rules may be worded in a way that the developer and tester could come away with a totally different understanding of the rule than the business person. Examples clarify the rules and thus the user story.

Here are a few final takeaways from the presenter. Sometimes features that describe technology updates on UI changes don't fit this model. Though you may be tempted to write examples using the standard Gherkin Given [] When [] Then [] pattern, the examples should be kept simple to illustrate the user stories. Don't just test the examples, test around the scenario as well. Teams may be resistant at first and may change as you make it a habit.

I liked how the session illustrated the concepts in a fun interactive way. There was slight confusion in how much information could be shared. We were asked to write all the examples for a single rule on a card. When asked to make our own examples based on those we did one card of examples to another. The team that did that for us instead put all examples of all rules mixed together which may have helped understand the user story confused what the actual rules were. Clarification on the steps may help the session go smoother, but really this was very minor as I enjoyed the session and took home lessons learned.

# [AppDev] Git Merge, Resets and Branches – Victor Pudelski

This is a session I was looking forward to. I've been using Azure DevOps (formerly TFS) primarily in my work and personal experience. I'm used to the concepts of branching and merging, but not for GitHub. I'm using a sample sandbox private repository I created a while ago to test what I've learned.

When I first decided to start creating and publishing articles like my [Tips](https://github.com/penblade/Tips), I wanted them exposed publicly and GitHub fit that bill nicely. My experience to date has simply been to use Visual Studio with the GitHub extension to clone a repository locally, make changes to the articles and code samples, commit changes, sync, and then verify the changes were in GitHub. This was my first time working with GitHub via command line and I was excited!

Victor started with a cloned repository, made a change that he knew would conflict with changes in his master branch, and then used git to add his changes and commit them.

1. git add --all
2. git commit
3. git pull "{repository}"
   1. [git pull is the equivalent of git fetch followed by a git merge](https://stackoverflow.com/questions/292357/what-is-the-difference-between-git-pull-and-git-fetch)

The merge failed due to conflicts as expected. He opened the file with the conflict and showed us how git changed the file.

1. Head
   1. The start of the conflict
   2. Changes you made in your local copy of the master repository
2. ===
   1. Changes they made in the master repository
3. {hash code}
   1. The end of the conflict

Resolve the conflict by accepting your changes between Head and ===, === to {hash code}, or manually merging the changes. Remove the Head, ===, and {hash code} lines. Test, code review, and then commit your changes. Pull the code again and merge to the master and it should work now. While you can merge straight to the master, my experience with Azure DevOps has me cautiously pull changes first and resolve conflicts locally before merging.

1. git reset --hard
   1. hard // lose changes, total undo
   2. mixed // changes in stage reset to working directory but nothing lost
   3. soft // reset to last commit, modified files set to staged (add)

Here's an example of some branching commands.

1. git checkout -b feat1
   1. This creates a branch.
   2. Recommendation is to use the feature or user story number in the name to reference what the branch was meant for
2. git branch
   1. Lists the branches
3. git add --all
4. git commit -m "comment"
5. git checkout master
   1. Switched the branch
6. git checkout feat1
7. git branch -d <branch>

Do not delete the repo and reclone. Do not rebase unless you know what you're doing.

There were plenty of commands relayed and additional discussions on branching strategies. I highly recommend reviewing Victor's slides. He did an excellent job at presenting the subject matter and paced through each section well. He answered questions throughout the session and left a great impression on me and others. Well done!

# [AppDev] Making Use of New C# Features – Brendan Enrick

Brendan illustrated many new C# 8 features along with some of his favorites introduced in recent versions that aren't as utilized as they could.

* The dreaded [Null Reference Exception](https://docs.microsoft.com/en-us/dotnet/api/system.nullreferenceexception?view=netframework-4.8) "Object reference not set to an instance of an object." is despised by all! There are many ways to handle it.
  + [Null Conditionals](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/member-access-operators#null-conditional-operators--and-) // return user?.Nickname;
  + [Null Coalescing](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/null-coalescing-operator) // return user ?? new User();
    - Does not stop an index out of range if the first element does not exist
    - // return this.Children?[0]?Name;
  + [Null Coalescing Assignment](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/null-coalescing-operator) // num ??= 42
  + [Nullable Reference Types](https://docs.microsoft.com/en-us/dotnet/csharp/nullable-references) // string?
    - AKA Nullable Annotation Contexts
    - Enforce the declaration of nullable types by project or by block of code
    - Enable or Disable
    - <nullable>enabled</nullable>
    - #nullable  
      // enabled  
      #nullable
* [Output Variables](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/out-parameter-modifier) // if (int.TryParse(rawInput, out int inputInts))
* [Discards](https://docs.microsoft.com/en-us/dotnet/csharp/discards) // if (int.TryParse(rawInput, out int \_))
* [Using Declaration Scope](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/proposals/csharp-8.0/using) // using var f1 = new FileStream("…");
* [String Interpolation](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/tokens/interpolated) // var fullName = $"{firstName} {lastName}";
* [nameof expressions](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/nameof) // var message = $"{nameof(id)} must be set.";
* [Tuples](https://docs.microsoft.com/en-us/dotnet/csharp/tuples) // (bool success, int number) SafeParse(string s) => (int.TryParse(s, out int n), n);
  + Check out my article on [Tuples](https://www.danylkoweb.com/Blog/developer-tips-c-tuples-LV)
  + Like the Null tips above there were plenty on Tuples
* [Pattern Matching](https://docs.microsoft.com/en-us/dotnet/csharp/pattern-matching) – [is expressions](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/is) // if (shape is Rectangle rec) return rec;
* [Pattern Matching](https://docs.microsoft.com/en-us/dotnet/csharp/pattern-matching) – [switch expressions](https://docs.microsoft.com/en-us/dotnet/csharp/pattern-matching#using-pattern-matching-switch-statements) // case Square s when s.Side == 0;
* Default Interface Methods
  + Interface IBotCommand  
    {  
     IList<string>Words { get; }  
     bool ShouldExecute(string word) => Words.Any(w => w == word);  
     Task Execute();  
    }

I skipped some of the features mentioned, but this gives you some good references. I highly recommend researching all of the new features added in C# 8 and the most recent versions to find new tools of the trade that may simplify or enhance your code.

This room was crowded and required additional chairs which were promptly brought in by conference center staff. Several in the crowd noted the session was in demand as one of the only .NET sessions in the time slot. Brendan did an excellent job in presenting. He knew the new C# 8 features well as he's been using them and his other tips extensively for a while. Great session!

# Keynote – Day 2 – Paul Kilroy

Paul started the Keynote by running their commercial [Huntington Bank – Looking Out for Each Other](https://www.youtube.com/watch?v=T45lII_j2xY) which also represents how their team members work together. Team members step out of their comfort zone to help customers.

When working with teams resistant to change they reach out to the team to learn and understand their circumstances. Based upon their analysis working with the team they develop an action plan. For one team perhaps they are not familiar with the new technology so they will be given access to training materials and other opportunities to learn and grow. They are more interested in raising skills up which we should all appreciate.

The Keynote took a different tact than most Keynotes I've attended. Paul wrapped up his presentation in the first half of his allotted time. He then opened the panel for questions for the remainder of his time. Below are just a few of the questions and responses.

1. Blockchain? Paul put forward that he thinks Blockchain has potential to lower costs. As an example Title searches take time and effort to verify information that can be done automatically by handling it with Blockchain. He did clarify that he was not including cryptocurrency in this discussion.
2. Automation? They have been investing in [RPA (Robotic Process Automation)](https://www.uipath.com/rpa/robotic-process-automation) to automate repetitive tasks that scales well while providing the agility to adjust as changes come up.
3. Migrate or Rewrite software? They follow the [strangler pattern](https://docs.microsoft.com/en-us/azure/architecture/patterns/strangler) to modernize systems. They separate concerns into microservices. They use git for their source control.
4. Build or Buy? Huntington prefers to build, not buy. When companies are purchased through acquisitions Huntington has their preferred stack and will import data into their system. If there is a product or feature on the acquiring company, then it is no longer offered to keep the acquisition clean. They may later implement the product or feature at a later date upon review, but it will be again done on their stack.

The Q&A format was interesting and provided insight into decisions that plagues many companies. I especially liked the "Migrate or Rewrite" and "Build or Buy" discussions. There were some compelling questions that came off the cuff that fed off earlier answers. The Q&A might have benefitted from some pre-submitted questions as the attendees were caught off guard for the first few minutes. However, the end result was an excellent wealth of knowledge in the inner workings of an enterprise level company that benefited all who attended.

# [SQL/BI] Demystifying NoSQL: The SQL developer's guide – Matthew Groves

Matt ran a "choose your own adventure" style presentation preceded by a short history lesson. In contrast to a session I attended a couple years ago that followed a similar format this session went smoother. There was a short history lesson intending to relate that [NoSQL](https://en.wikipedia.org/wiki/NoSQL) (Not Only SQL) concepts were avoided for a long time due concerns with hard drive space and performance. Most of those concerns are no longer valid with hardware improvements over the years.

My biggest issue with this format is the feeling that we may not have chosen the best path. What did we miss? Was there something important in one of the other paths? Overall, I felt like I got what I needed from the intro level session. While there were a couple advanced scenarios I think most followed along well enough.

Relational databases are not a one size fits all solution and [NoSQL](https://en.wikipedia.org/wiki/NoSQL) solutions address some of those areas. One situation is [impedance mismatch](https://en.wikipedia.org/wiki/Object-relational_impedance_mismatch) where the objects representing the data do not match the normalized database table structure.

NoSQL [document oriented stores](https://en.wikipedia.org/wiki/Document-oriented_database) can address this scenario in one fashion by representing the data as XML or JSON in a column allowing an ORM like Entity Framework to perform the mapping or use ADO.NET to read the data and use serialization/deserialization techniques to read the data.

NoSQL [graph databases](https://en.wikipedia.org/wiki/Graph_database) represents data based on nodes, edges, and properties.

NoSQL [wide column stores](https://en.wikipedia.org/wiki/Wide_column_store) use tables, rows, and columns. However, the names and columns may change per row in the same table more like a key-value pair.

Instead of listing all of the varied NoSQL variations I'll refer a site that lists all of the [NoSQL database systems](http://nosql-database.org/) for your reference with a description and a link.

Once your hands aren't strapped to the Relational Database model new options and challenges start to arise. See my article on [XML Shredding](https://www.danylkoweb.com/Blog/dude-lets-xml-shred-MT) for a taste of one challenge you could face with a document oriented store. Granted some of these challenges are addressed by using tools available from [Couchbase](https://www.couchbase.com/) and other vendors by using declarative queries. I found this article had an entertaining way to describe the difference between [imperative vs declarative query languages](https://neo4j.com/blog/imperative-vs-declarative-query-languages/).

# [AppDev] Supercharge your API Testing using Postman – Umang Nahata

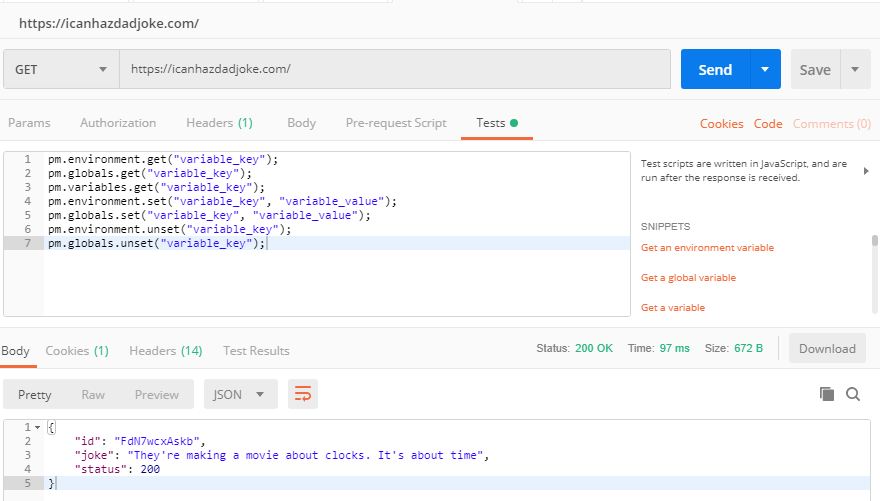
Umang appealed to my passion for telling dad jokes available through <https://icanhazdadjoke.com/api>. Each time he tested the API through [Postman](https://www.getpostman.com/) a new joke graced the screen to my delight. Ex. "They're making a movie about clocks. It's about time". Classic!

The steps/checklist to build your API tests:

1. Gather details about API: url, documentation, examples
2. Open Postman and get started
3. Optional: test ta, test runner

Quick tips

1. Postman offers many features, so Umang walked us through [navigating Postman](https://learning.getpostman.com/docs/postman/launching_postman/navigating_postman/)
2. Postman can open web pages by setting the request url to the web page url
3. View > show [Postman console](https://blog.getpostman.com/2016/08/26/the-postman-console/)
   1. [Debugging using the Postman Console](https://learning.getpostman.com/docs/postman/collection_runs/debugging_a_collection_run/#debugging-using-the-postman-console)
4. Tests tab – [Test examples](https://learning.getpostman.com/docs/postman/scripts/test_examples/) (snippets) on the right
   1. Tests are run after a response has been received
   2. Ex. Pm.test("Status code in 200", function()) console.log



Create an [environment](https://learning.getpostman.com/docs/postman/environments_and_globals/intro_to_environments_and_globals/) for DEV, QA, PROD, etc. Each environment usually contains the same list of [variables](https://learning.getpostman.com/docs/postman/environments_and_globals/variables/) with different values for each environment.

[Import](https://learning.getpostman.com/docs/postman/collections/data_formats/) settings, collections, environments, headers, and swagger

1. [Import swagger](https://stackoverflow.com/questions/39072216/how-to-import-swagger-apis-into-postman/39072519) by clicking Import > Paste Raw Text > Paste the JSON > click import > a collection is created matching the swagger input

[Runner](https://learning.getpostman.com/docs/postman/collection_runs/starting_a_collection_run/)

1. Runs tests for selected environment
2. Tests are run sequentially in the order they appear in the collection
   1. [Build workflows](https://learning.getpostman.com/docs/postman/collection_runs/building_workflows/) to run the tests in the order you want
   2. [Chaining requests](https://blog.getpostman.com/2014/01/27/extracting-data-from-responses-and-chaining-requests/) are done by running scripts before/after you receive a response from the server
3. [Random data function](https://vivrichards.co.uk/testing/postman-generating-random-post-data)
   1. pm.environment.set("RandomMonth", \_.random(1,12));

I'm not a big fan of using random data in my tests. They can be powerful when done right. I just haven't seen it done right in my experience. If I write a test I expect it to pass with the data I provided. If I use random data and I don't filter it properly I may get unexpected results return false negatives. That is not something I want to deal with during any deployment especially for a smoke test on a production deployment. I prefer stable tests so I avoid the random.

## Discussion on Room Requirements (wide vs. long)

Umang's presentation drew in a crowd requiring additional chairs to be brought in within the first 5 minutes of the session. Unfortunately, the Postman screen had a lot going on so the font size was small. I was near the back of the room and I could not read much of what was on the screen. It was fine for those in the front half. Others left due to this issue and some of us shifted closer.

This session would have been better in a wider room than the long room it was held. Umang took this in stride and accommodated by zooming in, more frequently by the end when it became more of a habit. I think this issue can be mitigated in the future by improving communications between the speakers and coordinators regarding room requirements.

Coordinators could ask if the presentation will use an IDE with small fonts like Postman, Visual Studio, etc. or perform a live demo. If yes, then ask if they would prefer a wider room so attendees will be closer to the screen. If a wider room is not available, then remind the presenter to zoom in frequently on anything important. The presenter of course should keep this in mind and let coordinators know up front. Of course this is all 20/20 hindsight and it’s a team effort on both sides to get this right.

# [AppDev] Iterative Development: The Paper Airplanes Game – Natalie Lukac and David Knighton

The Paper Airplanes Game is a fun interactive game. We played a variation of the [Paper Plane Game – Practitioners Agile Guidebook](https://agilefaq.wordpress.com/2018/11/25/paper-plane-game/). Teams of 3-4 were formed. Following a simplified Agile process each team went through 3 iterations. Each team member was only allowed to make a single fold before passing it to the next "developer". We only counted the number of planes that passed the test and ignored work in progress planes and those that failed. To pass the test the plane had to fly about 10 feet marked out on the floor with tape.

My team consisted of 3 people and for some reason we named ourselves after the [Princess Bride](https://en.wikipedia.org/wiki/The_Princess_Bride_(film)) characters. [My Name Is Inigo Montoya](https://en.wikipedia.org/wiki/The_Princess_Bride_(film))! Classic!

Iteration 1:

1. 3 minutes – Design/Planning
   1. As a team we discussed different paper airplane designs
   2. We settled on the standard [Dart](https://www.youtube.com/watch?v=V-rBmbBSGlA) design to have as few folds as possible
   3. We decided that we would perform an assembly line and as each of us finished a plane we would test the plane
   4. We estimated we could have 10 successful planes
2. 5 minutes – Build/Test/Deploy
   1. We tried to cut out a couple of the folds, but it caused the planes to crash and burn (still not sure how we managed that :P), so we adjusted midway to the standard dart folds
   2. Being a little crowded, some of our planes veered into developers on other teams so we called interference and tried again
   3. We completed/deployed 11 successful planes
3. 2 minutes – Retrospective
   1. Each developer stopping the assembly line to run a test slowed production significantly
   2. The adjustments mid sprint by adding all folds produced a better quality product

Iteration 2:

1. 3 minutes – Design/Planning
   1. We decided to designate one developer as the QA tester who would break off the assembly line once a minute to perform the tests
   2. While the QA person was testing the 2 remaining developers would pass the planes back and forth to each other
   3. We estimated we could have 18 successful planes
2. 5 minutes – Build/Test/Deploy
   1. The assembly line worked mostly as expected, but we would sometimes get stuck waiting for each other
   2. In the middle of the sprint when we found ourselves waiting we would start a new plane and instead of feeding the planes to each other we moved the planes into a backlog pile
   3. We completed/deployed 23 successful planes
3. 2 minutes – Retrospective
   1. We liked the concept of the backlog pile

Iteration 3:

1. 3 minutes – Design/Planning
   1. We decided to incorporate the backlog piles
   2. The QA person would feed his planes into whichever backlog was lowest and would pull from the highest to help balance
   3. We estimated we could have 25 successful planes
2. 5 minutes – Build/Test/Deploy
   1. The assembly worked mostly as expected and the backlogs worked well
   2. Unfortunately, we were getting tired at this point and fell into old habits of performing folds in a different order
   3. This caused bugs in the software to form as the next developer was expecting the code, I mean folds, in a specific stage/pattern
   4. This in turn caused delays to determine the next fold or we introduced a defect (technical debt)
   5. The DEV/QA person identified the defect before flying the plane and fixed the bug in some cases
   6. Even with the defects introduced, the improved process ensured that most of the planes were of good quality
   7. We completed/deployed 25 successful planes
3. 2 minutes – Retrospective
   1. We resolved to review the order of the folds to make the process smoother going forward
   2. Get some rest

The other teams weren't as successful. Some of their issues were as follows.

1. They struggled to develop a quality product that could pass the test
2. They didn't form an assembly line and went sequentially
3. Each team member performed every role

I really enjoyed this session and wish more attendees had joined in the fun. This exercise illustrated many of the benefits of [iterative development](https://en.wikipedia.org/wiki/Iterative_and_incremental_development): Design/Planning > Build/Test/Deploy > Retrospective > Repeat. Our team embraced the concept and building upon prior experiences learned and improved greatly from the beginning of the session up to the end. While the other teams struggled in some aspects they too improved by the end. I highly recommend participating in the Paper Airplanes Game as an entertaining way to learn iterative development.

# [AppDev] WebAssembly Live! – Mike Hand

# Conclusion