Introduction to DQMH

The World’s Most Popular 3rd-Party Framework for LabVIEW

Darren Nattinger

Chief TSE, CLA

NI

Before we get started

All of my presentations (including this one) are available at:

dnatt.org

(slides, demos, and links to video recordings)

This presentation's link: http://bit.ly/dnattdqmhintro

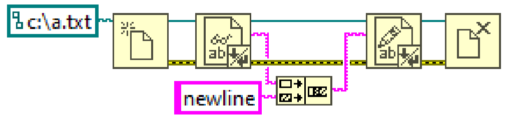
[Download link for ZoomIt]()

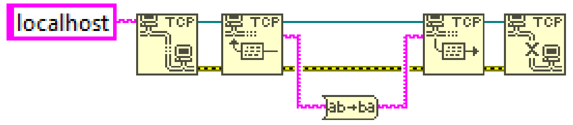
Glossary

(As defined by me)

General Glossary

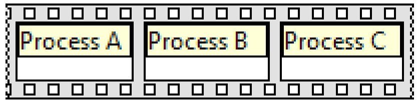
- **Process** – continuously-running code
  - Almost always a VI with one or more **while loops**
- **Reentrancy** – the ability for multiple instances of the same VI to run simultaneously
- **API** – application programming interface
  - Group of related functions organized in a logical manner
- **Business Logic** – application-specific code
  - Code that is *not* part of the framework being used
  - Written in pre-defined, documented places in the overall code



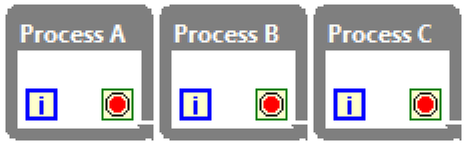


General Glossary

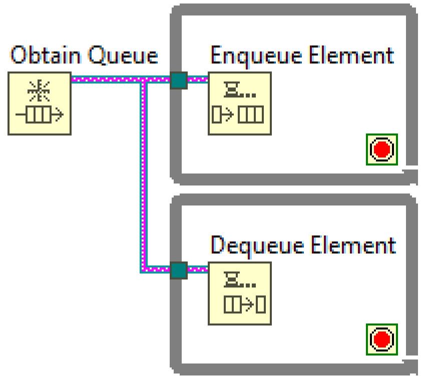
- **Synchronous Process** – ordered operation, dependent on completion of another process



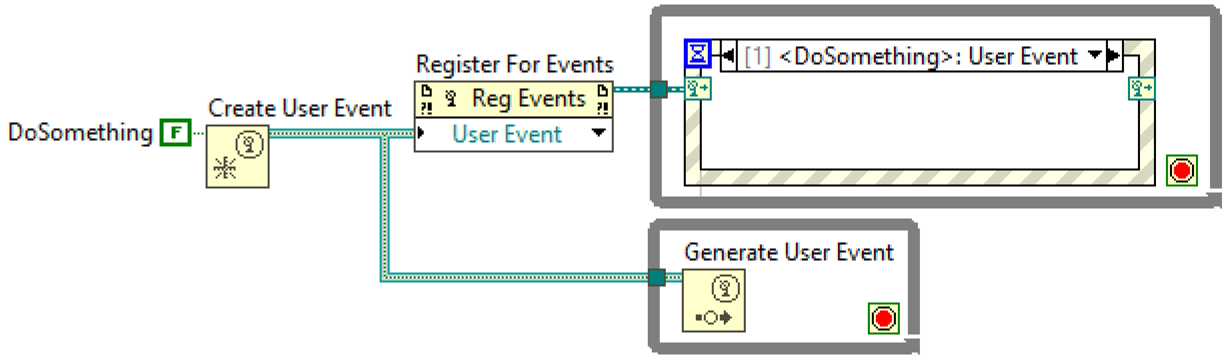
- **Asynchronous Process** – independent operation, runs in parallel with other processes



- **Queue** – LabVIEW API for passing data within or between processes



- **Event** – LabVIEW API for passing data within or between processes



General Glossary

- **Design Pattern -** Theoretical mechanism to execute synchronous or asynchronous code.
  - Examples: state machine, queued state machine, producer/consumer, **queued message handler**

- **Architecture -** Real-world implementation of one or more design patterns that facilitates execution of asynchronous code.
  - Templatized approach to implementing business logic
  - Examples: JKI State Machine, TLB', Messenger

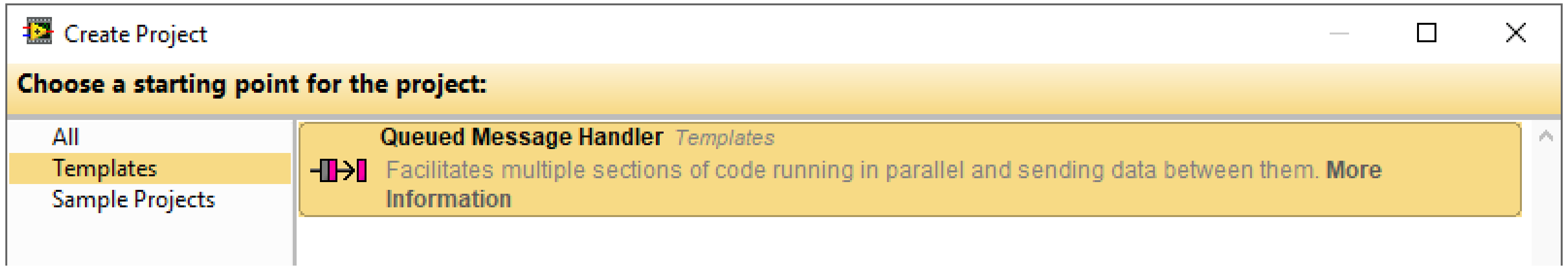
- **Framework -** Consumer-grade architecture, with **documentation** and **tooling** to improve developer experience.
  - Examples: DQMH, Actor Framework, Workers, JKI State Machine Objects

NI QMH: A Brief Discussion

QMH: Queued Message Handler

NI QMH

- Project Template introduced in LabVIEW 2012



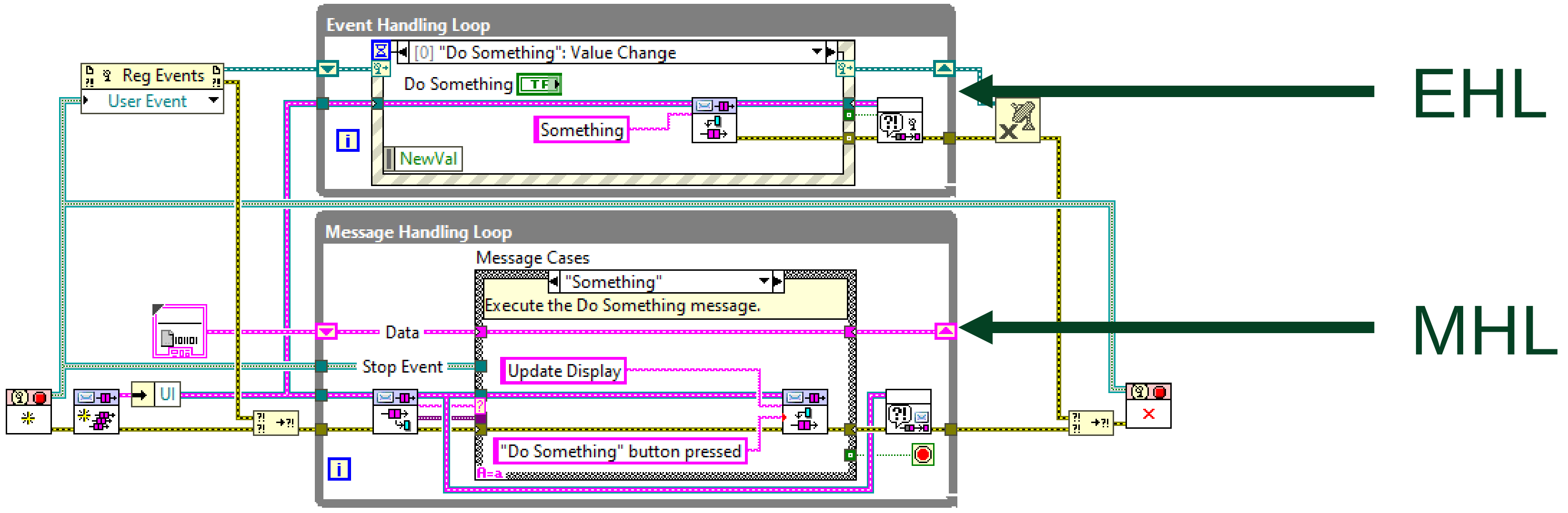
Create Project

Choose a starting point for the project:

All
Templates
Sample Projects

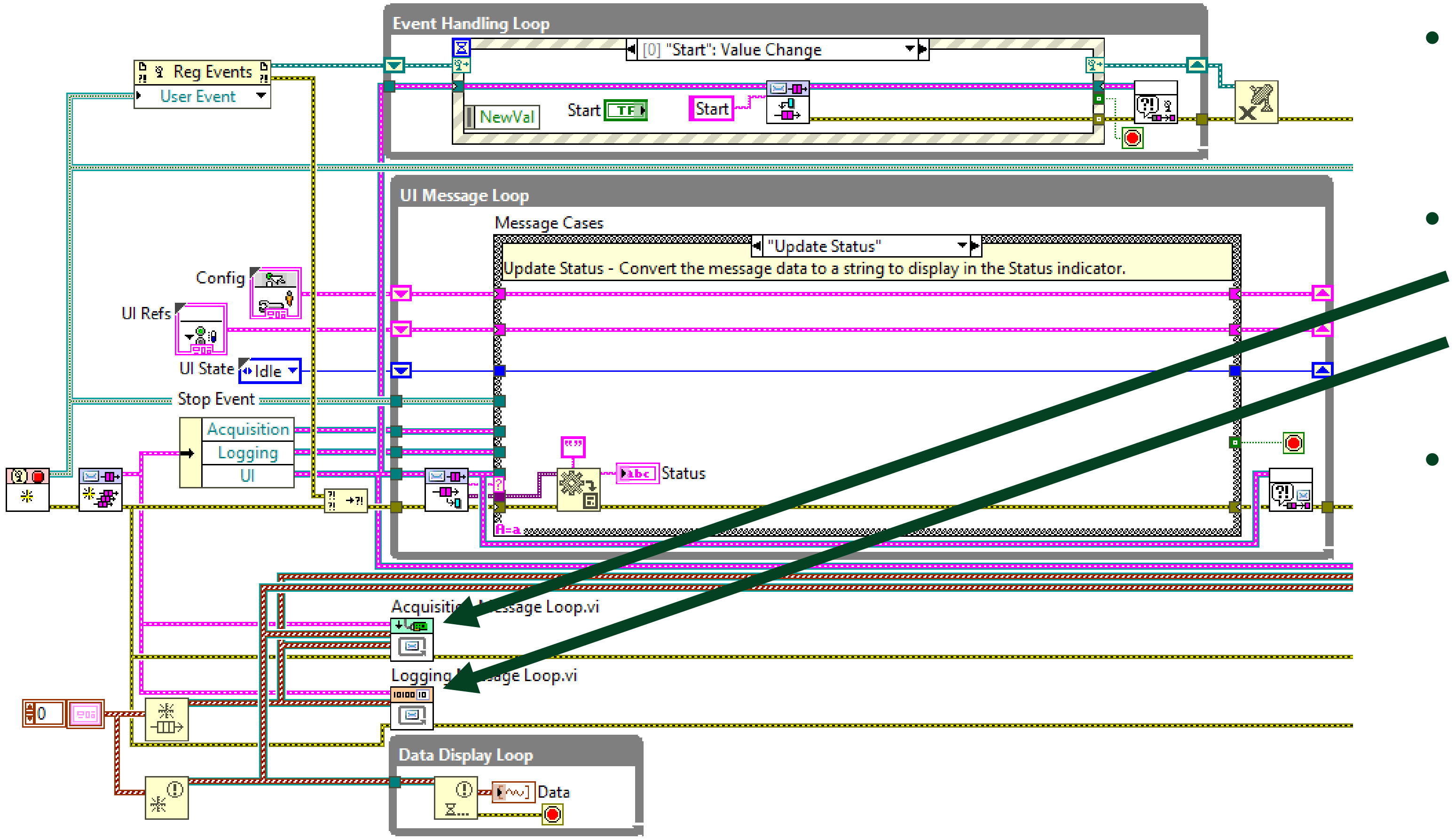
Queued Message Handler *Templates*
Facilitates multiple sections of code running in parallel and sending data between them. **More Information**

- The official NI template for using the QMH design pattern



NI QMH

- Continuous Measurement and Logging sample project built on NI QMH
  - NI-DAQmx installer includes a version that uses DAQmx API calls



- Same initial structure of NI QMH template
- Additional MHLs
  - Acquisition
  - Logging
- Queue references passed into subVIs that need access to them

Problems with NI QMH

- Communicating between processes is not straightforward
  - Wire more queue references into subVI connector panes?
  - Not scalable for systems with many modules
  - Difficult to debug issues when all queues are available to all MHLs

- Limited reusability of process VIs
  - For example, the Acquisition MHL VI has a reference to the UI Queue on its connector pane

- Difficulty in supporting reentrancy
  - No built-in mechanism to support multiple instances of the QMH VI running in parallel

- “Architecture”-level changes must be made manually
  - Adding frames to EHL and MHL can be tedious and error-prone

ni.com

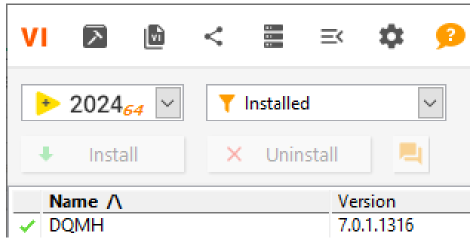
DQMH Basics

DQMH History

- 2015 – First public release
  - Originally developed by **D**elacor, an NI Alliance Partner
    - Chief Architect – Fabiola De la Cueva
- 2016 – LabVIEW Tools Network Product of the Year
- 2018 – DQMH Trusted Advisors Program
- 2021 – Formation of DQMH Consortium
  - http://www.dqmh.org
- 2024 (present day) – Latest product release – **DQMH 7.0.1**

DQMH Basics

- Free to download and use

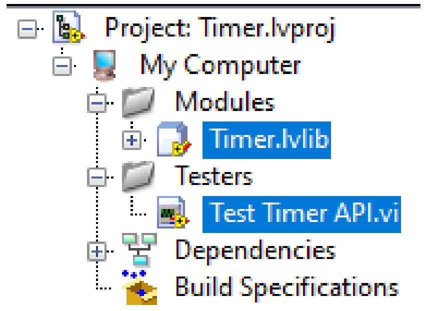


Name /\	Version
✓ DQMH	7.0.1.1316

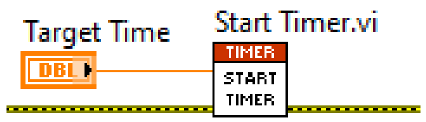
- Framework for LabVIEW to facilitate large application development
- Most popular 3rd-party framework for LabVIEW in the world
- DQMH Framework certification badge available from NI
- Same intra-process communication scheme as NI QMH (queues)
- Adds an inter-process communication scheme (user events)
- Designed to be accessible to **CLAD/CLD-level LabVIEW programmers**
    - Minimal use of LabVIEW classes out of the box

DQMH Glossary

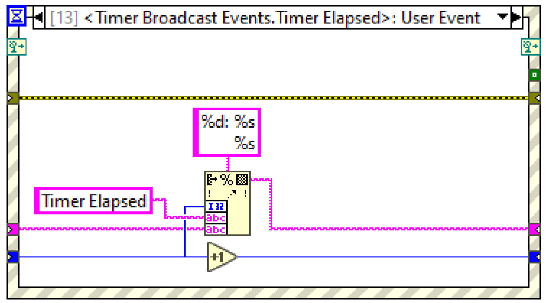
- **DQMH Module** – The basic building block of the DQMH framework
  - Asynchronous process with a well-defined **Public API**
  - A LabVIEW Library (.lvlib)
  - Can be **singleton** (non-reentrant) or **cloneable** (reentrant)



- **Request** – A way for the external world to ask the DQMH Module to do something
  - Communication mechanism is a User Event
  - Implemented as a VI in the module's Public API
  - Can optionally include a reply for whoever makes the request



- **Broadcast** – A way for the DQMH Module to tell something to the external world
  - Communication mechanism is a User Event
  - Zero or more external event structures might be registered for this event
    - (the module doesn't care)



DQMH Glossary

- **Main VI** – The main QMH VI of your module

- **API Tester** – A VI that lets you “test” the Public API of your DQMH Module
  - One of the most useful parts of DQMH

- **Scripting Tools** – What makes DQMH Framework a framework
  - **Eliminate the need for manual changes to framework-level code**
  - Create/rename/validate module
  - Create/rename/remove/convert request and broadcast events
  - Workflow encourages best practices
    - Enter documentation when creating an event
    - Tester VI diagram shown after scripting, encouraging tester maintenance

Demo

Let’s create a simple DQMH module

Creating a DQMH Module

- Let's create a **Timer** DQMH Module
- Features:
  - Start a timer with a specified time duration
    - This will be a **request** event
  - Stop the timer
    - This will be a **request** event
  - Be notified when the timer elapses
    - This will be a **broadcast** event

- **DEMO** – Creating the timer module
- **DEMO** – Integrating the timer into a larger application

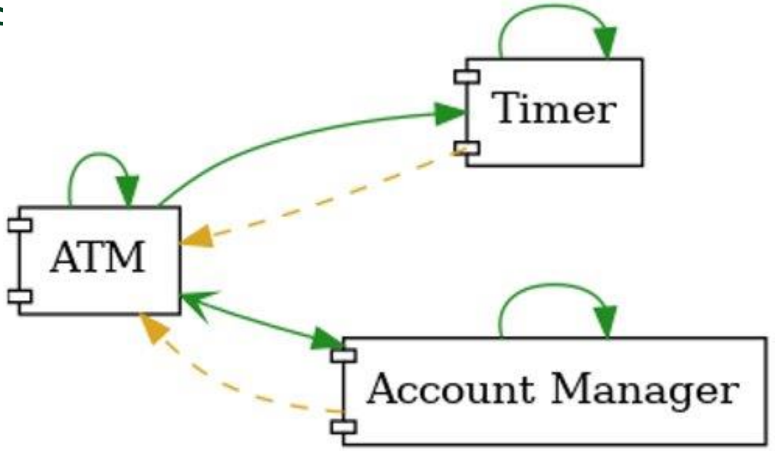
Developer Roles

How are you interacting with DQMH?

Are you a DQMH Module **developer**?

- …then you are *creating* the **Requests** and **Broadcasts** for a DQMH module.
- …and you are writing the business logic code in the **MHL** of a DQMH module Main VI.

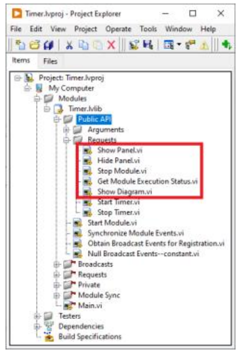
- **Keep your API tester up to date!**
- Follow DQMH Best Practices
- Use the scripting tools for all framework-level changes
  - Don't manually update framework VIs!
- Document **everything**
  - Module description
  - Event descriptions
  - Use Antidoc to auto-generate project documentation
- Create **module templates** to save time



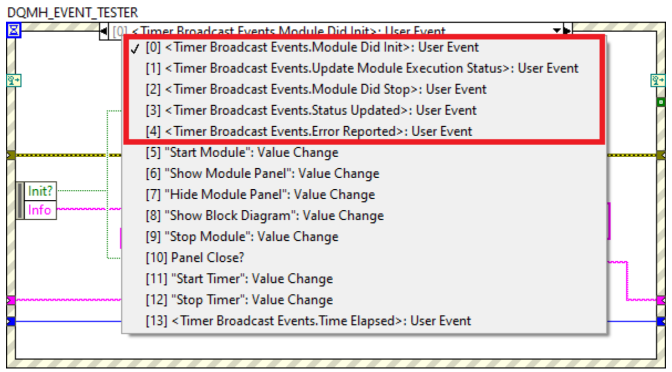
Are you a DQMH Module **user**?

- ...then you are *using* the **Public API (Requests)** and **Broadcasts** of a DQMH module.
- …and you are *not* looking at any of the MHL business logic in the DQMH Main VI.

- If the module was installed via VIPM, it should have a palette
- If not, find what you need:
  - Requests: in the **Public API** folder in the module .lvlib
  - Broadcasts: in the event list of the Event Structure
- DQMH modules include several default requests and broadcasts:



```
Timer.lvproj - Project Explorer
File  Edit  View  Project  Operate  Tools  Window  Help
Items  Files
Project: Timer.lvproj
  My Computer
    Modules
      Timer.lvlib
        Public API
          Arguments
          Requests
            Show Panel.vi
            Hide Panel.vi
            Stop Module.vi
            Get Module Execution Status.vi
            Show Diagram.vi
            Start Timer.vi
            Stop Timer.vi
          Start Module.vi
          Synchronize Module Events.vi
          Obtain Broadcast Events for Registration.vi
          Null Broadcast Events--constant.vi
        Broadcasts
        Requests
        Private
        Module Sync
        Main.vi
    Testers
    Dependencies
    Build Specifications
```



```
DQMH_EVENT_TESTER
[0] <Timer Broadcast Events.Module Did Init>: User Event
✓ [0] <Timer Broadcast Events.Module Did Init>: User Event
  [1] <Timer Broadcast Events.Update Module Execution Status>: User Event
  [2] <Timer Broadcast Events.Module Did Stop>: User Event
  [3] <Timer Broadcast Events.Status Updated>: User Event
  [4] <Timer Broadcast Events.Error Reported>: User Event
  [5] "Start Module": Value Change
  [6] "Show Module Panel": Value Change
  [7] "Hide Module Panel": Value Change
  [8] "Show Block Diagram": Value Change
  [9] "Stop Module": Value Change
  [10] Panel Close?
  [11] "Start Timer": Value Change
  [12] "Stop Timer": Value Change
  [13] <Timer Broadcast Events.Time Elapsed>: User Event
```

Look at the tester VI to see examples of how to use the module’s Public API and broadcasts

- ...and to verify the module is running correctly

Benefits of DQMH

Benefits of DQMH

- **Scripting Tools** automate framework-level changes
  - New module, new request, new broadcast, etc.
  - Module Validation: automatic application of bug fixes when upgrading!
  - Continued maintenance of API Tester during development
- QMH is a **familiar pattern** for CLAD/CLD-level LabVIEW programmers
- Handles **fundamental aspects of asynchronous LabVIEW programming** (with no additional effort on the part of the developer):
  - Starts when you tell it to start
  - Stops when you tell it to stop
  - Basic error management
  - Basic panel management
  - Built-in debugging
- Active DQMH **developer community**
- DQMH Consortium responsive to **feature requests**
- Many more benefits listed on the HSE site here

HAMPEL SOFTWARE ENGINEERING

How to Learn DQMH

DQMH topics that I didn’t cover today...

- Request and Wait for Reply events
- Cloneable Modules
- Helper Loops
- Private Events
- Local Instance Events
- Module “hierarchy”
- Using DQMH with Hardware Abstraction Layers
- Best Practices
- Additional Tools like Antidoc and Panther Dashboard
- ...

How to Learn DQMH

- DQMH Documentation: How to Learn DQMH
- Tom’s LabVIEW Adventure videos
- DQMH Consortium videos
- Shipping example
    - *[LabVIEW 20xx]\examples\DQMH Consortium\DQMH Fundamentals - Thermal Chamber*
- Official DQMH Training Course

How to Learn the **Basics** of DQMH **in 1 day**

http://bit.ly/dqmhlearningpath

- Step-by-step quick learning path that covers the **most common** DQMH concepts
- Step 1 is watching this presentation ☺
- References multiple existing resources (examples, sample projects, videos, etc.)
- (not a substitute for the official DQMH training course)

Thanks for attending!

bit.ly/dnattdqmhintro

dnatt.org