European DataFlex users' conference - Edinburgh, Scotland

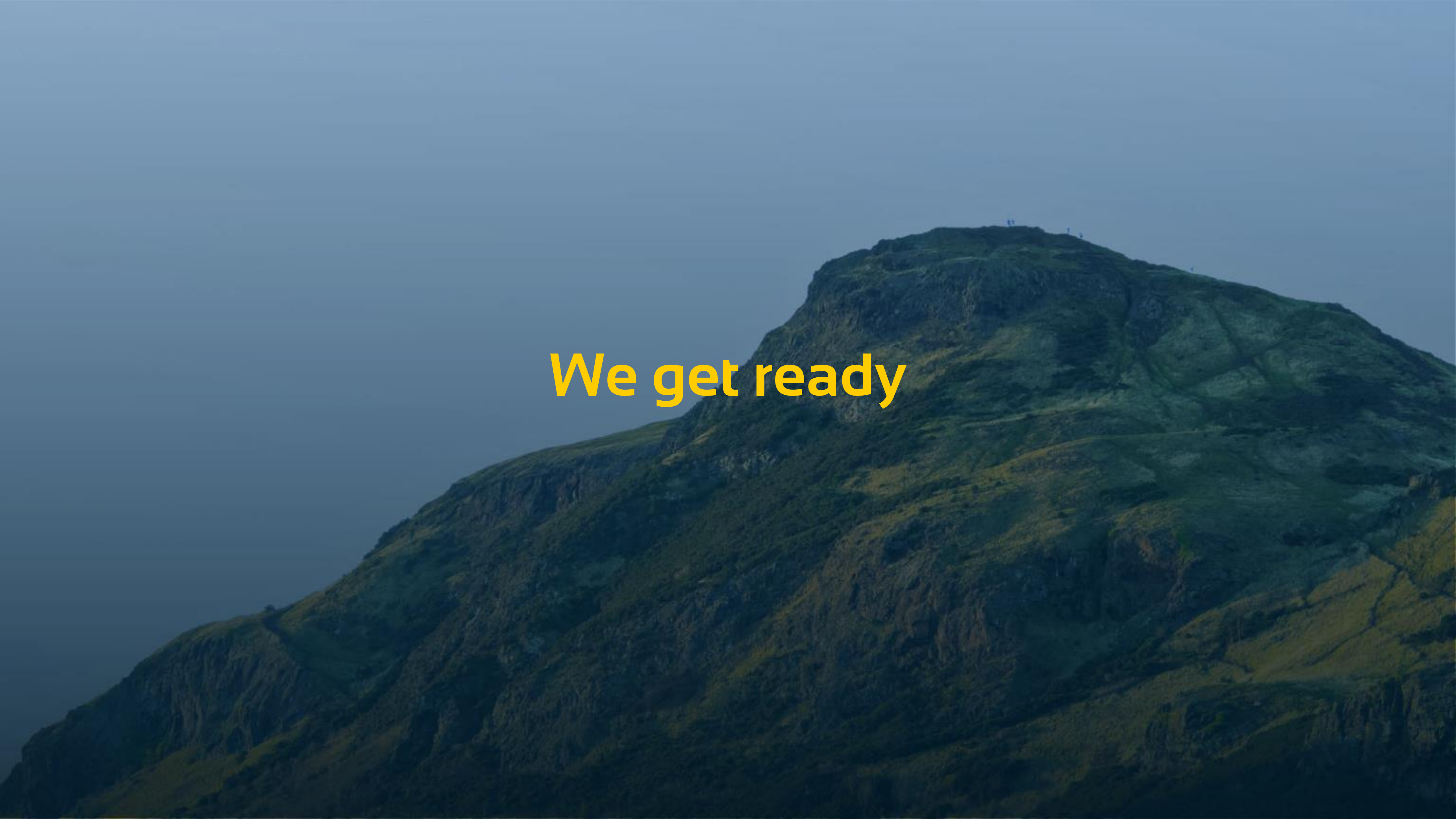
EDUC 2018

DataFlex to New Heights

Getting Your Applications Ready for DataFlex NextGen

John Tuohy

DataFlex NextGen Review



We get ready

Code cleanup project - Goals

- As the NextGen process progressed we've been looking at old code and techniques and asking:
  - What does this even do?
  - Does it even work?
  - Do people still use it?
  - Oh no, we still use it!
  - Can it be moved to DataFlex NextGen?
  - Should it be moved to DataFlex NextGen?

Code cleanup project - Goals

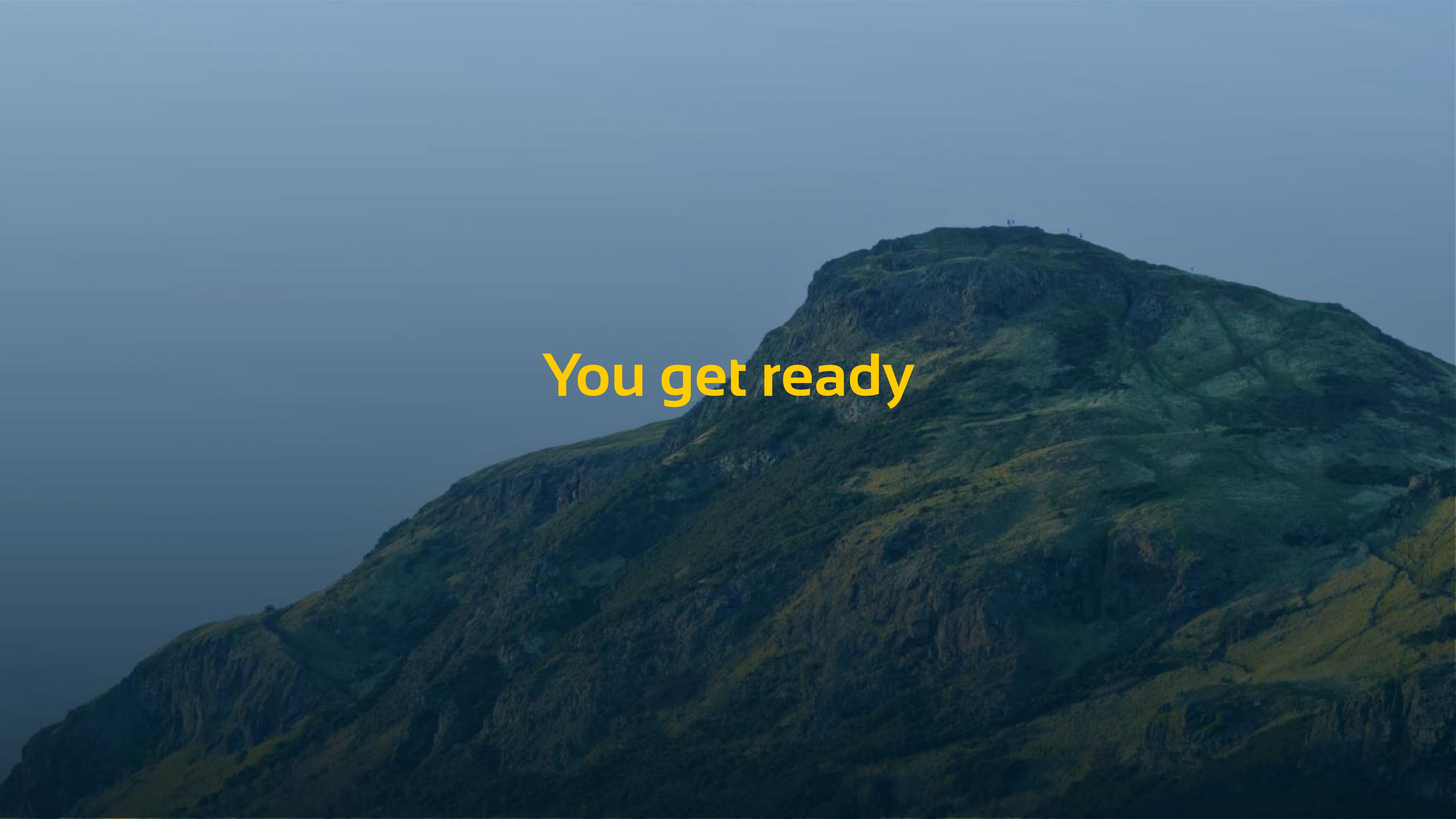
- We decided that we will migrate as much as we can but that we:
  - Need a way to identify and discourage obsolete use
  - We need to make sure we are not using these techniques in our public code
- We decided to start by cleaning our “public” code
  - packages and samples
- This has been something on our to-do for quite a while but has always been deferred
- We decided to start this process for DataFlex 19.1

Code cleanup project - How

- How we did it:
  - We went through our product and decided what things should be considered obsolete
  - We built an automated warning system to help us find those things
  - We modified the Studio to display warnings and make it easy to edit them
  - We added a compiler warnings throughout our code
  - We chose to be pretty strict about this. When in doubt issue a warning
- We cleaned up all warnings in our packages and samples
- While we were at it we cleaned up:
  - The formatting of all of our source code
  - The comments in our code

Code cleanup project - Results

- The results of this are in DataFlex 19.1
- Once we built the system, we still had to do a lot of tedious work to do
- The good news is that once identified, it’s pretty easy to improve the code
- The even better news is that upon completion, it feels really good to bring things up to date
- And… this provides mechanism to stay up to date on an ongoing basis



You get ready

Bringing code cleanup to you

- We felt that a robust compiler warning system will be equally welcomed by our developers
- You deserve the same strict warning system that we imposed on our own code … with the following caveats:
  - It can be disabled and enabled, so you can use it when you are ready for it
  - Your applications will run as before, despite the warnings
  - It is easy to use
- You can do this in DataFlex 19.1

How you can use compiler warnings

- Enable warnings for a project
- Compiler you application and see *all* the warnings
- You can choose to fix as many or as few of the warnings as you like
- Your application runs the same as ever
- You can even choose to use the #warning command yourself
  - Mostly we expect you to just use the warnings we provide

Compiler Warnings

How compiler warnings are implemented

- New compiler command - #Warning

```
#Warning DFERR_COMP_WARNING_OBSOLETE_PACKAGE "ArrayPut.pkg is obsolete"
```

- We added warnings throughout our packages and command definitions (fmac)
- We modified the Studio to display warnings
- Warnings can be enabled disabled at the project level (and more)
- You compile your application, you see warnings in the Studio

Warning Types

- We have warnings for the following:
  - Obsolete commands
  - String commands vs. String functions
  - Obsolete keywords (e.g. public, private, local)
  - Obsolete classes (when instantiated as an object)
  - Obsolete packages (when Used)
  - Obsolete global functions (when called)
  - Obsolete use of the old Type/End_Type structs
  - Use of indicators
  - “If” commands on a single line

Refining compiler warnings

- If you have suggestions for other warnings let us know
- If you disagree with our warnings let us know
- Limitations of compiler warnings
  - There are things we just can’t detect
  - Our loose data type casting can make it hard to detect bad data types at compile-item
  - Our late binding object message system impossible to detect obsolete object based methods
  - There are techniques that are too hard to catch

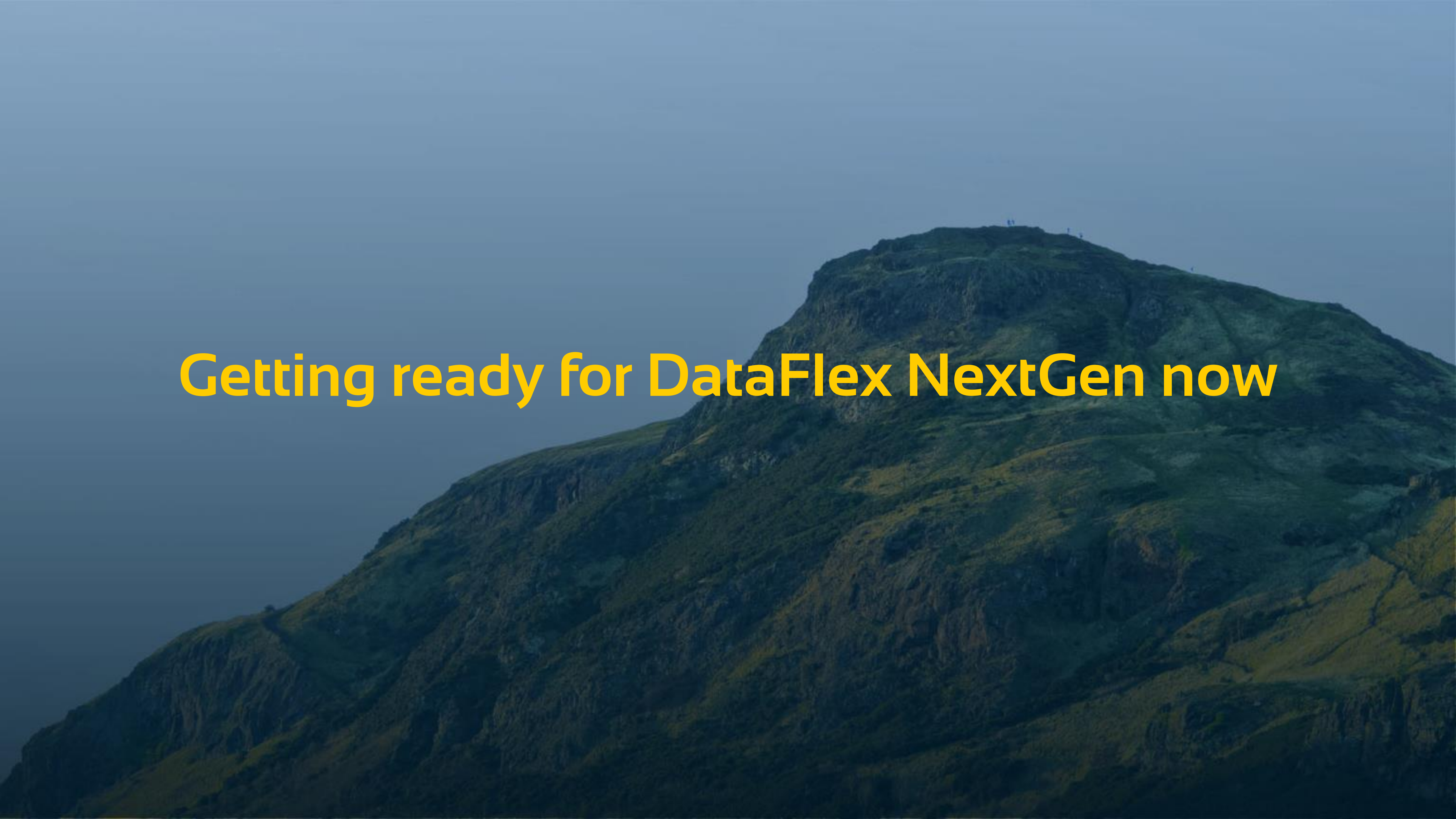
Additional changes that just might (temporarily) break your application

DFAllent and removed packages

- We have removed a number of obsolete packages from DFAllEnt.pkg
  - These contain classes that are obsolete and have been replaced with better alternatives.
  - If your application compiles, you don’t need them - congratulations
  - If you get compiler errors
    - Add them back with a “Use OldDfAllEnt.pkg”
    - If you think these classes still have value, let us know

Built in commands have been removed

- Some commands have been moved out of FMAC
  - These are commands that are so old, that no-one should be using them
  - Some don’t even work
  - If you are using these:
    - You will get a compiler error (command not found)
    - You can add them back with a “Use OldFmacCommands.pkg”
    - If you think these commands still have value, let us know



Getting ready for DataFlex NextGen now

Code Cleanup and DataFlex NextGen

- Our goal is that all of our code is up to date before moving to DataFlex NextGen
- We hope you will want to do the same with your code
- Most of your obsolete code will run fine in NextGen DataFlex
  - These obsolete items are not necessarily going away
  - Changes are going to be required when you move the NextGen
  - The more current your code, the easier this process will be
  - We are providing you with the tools to do that now

Integers and Pointers in NextGen

- Integers and Pointers
  - In 64-bit, integers will still be 32-bit
  - Pointers will be 64-bit or 32-Bit depending on platform
  - *You cannot treat Integers and Pointers as interchangeable*

- You need to review your code and make sure you use Pointer or Address when working with memory pointers.

Handles in NextGen

- Handles
  - In DataFlex the Handle type is used for:
    - DataFlex Objects
    - Windows Handles
- Handles in 32-bit
  - DataFlex Handles are 32 bits
  - Windows Handles are 32 bits
- Handles in 64-bit
  - DataFlex Handles are always 32 bits
  - Windows Handles are usually 32 bits in a 64 bit space (huh?)
- Check your code and make sure you are not using Handles for pointers.

Windows APIs in NextGen

- You must make sure your API definitions use the correct Windows datatypes
  - Windows DLL calls (External_Function)
  - Windows Notifications
  - Windows Structs
  - Windows Structs also have different padding rules for 32 and 64 bit application
- If you are using obsolete the Type / End_Type commands and its surrounding commands, we advise you switch over to Structs now.
- If you define additional Windows structs, you will need to double check them
- You need to change Windows notifications to use the right datatype - that's what LongPtr is for

Strings in NextGen

- Strings and Unicode
  - In DataFlex strings have been used to manage character strings and bytes of memory.
  - With Unicode this is not the same thing.
  - Our String function library is going to be extended and modified to handle string byte and character usage.

- If you are using obsolete string commands, we advise you to switch these to string functions now
- Check your code for string usage and start identifying places where you are using strings to manipulate memory.

All of this can be done now

- We’ve already made these changes in DataFlex 19.1
- You can start doing the same in your applications
- We will be providing specific changes and guidelines during the DataFlex 19.1 release phase
- We will get you there!

The virtues of being up to date

- There is a big overhead in constantly updating to the latest
  - Trust us on this one – we *feel* your pain
- Can you fall back to the "If it's not broke, don't fix it" strategy?
  - This strategy is no longer viable in the 21st Century
- Keep your DataFlex applications up to date
  - You get all the latest new features
  - We will keep your application working in an ever changing environment
  - We can't help you, if you won't help yourself
  - When 64-bit / Unicode DataFlex is here, will you be ready?

European DataFlex users' conference - Edinburgh, Scotland
EDUC 2018

DataFlex to New Heights

Thank you!

Are there any questions?