Interop

Simon Robinson
Software Developer

@TechieSimon www.SimonRobinson.com

Overview

Interoperating with

- The native (unmanaged) Windows API
- Managed VB and F# code

Windows API Functions

- Enable more features in apps
- No longer confined to BCL capabilities

VB and F#

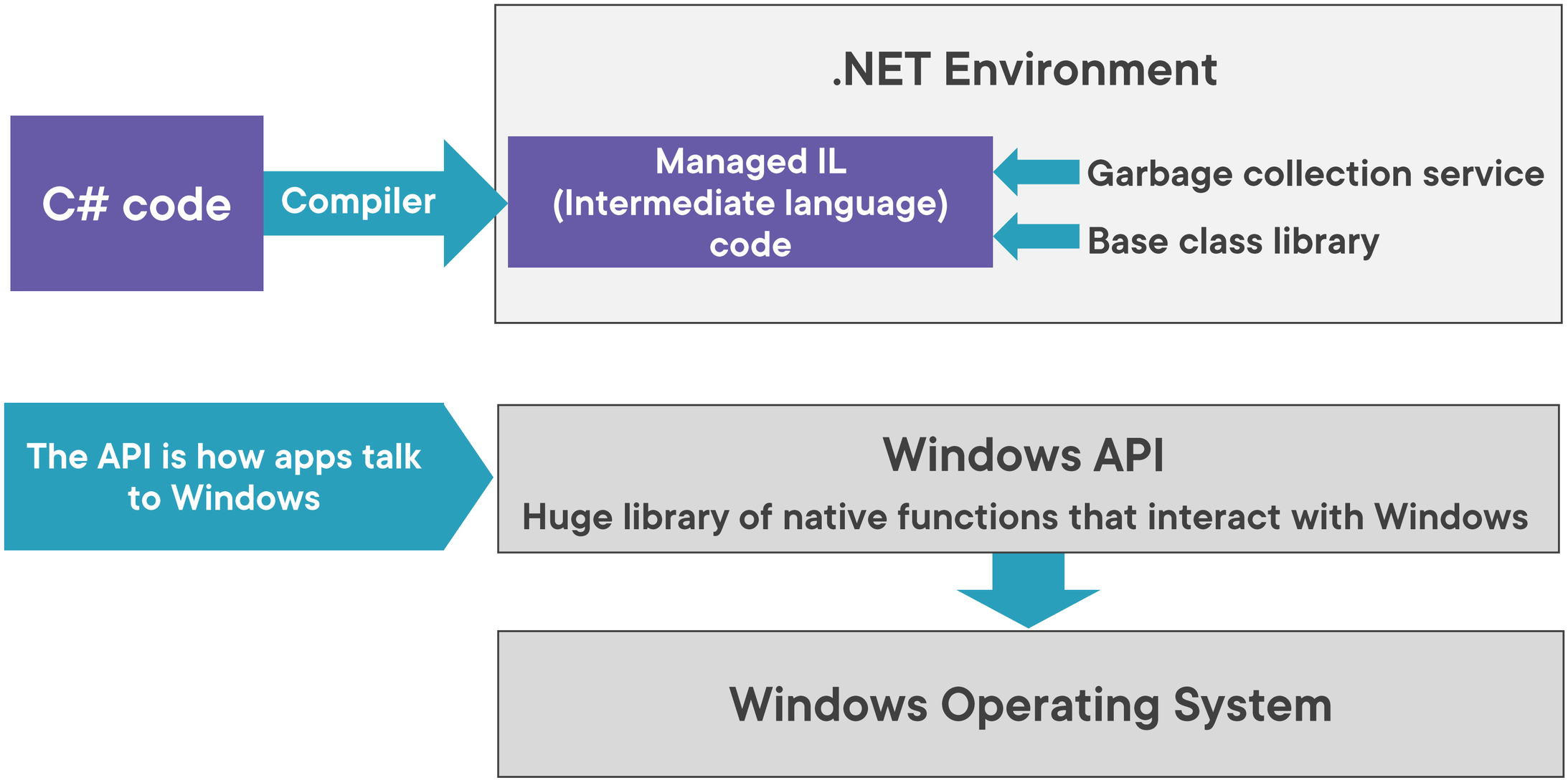
- Use code produced by teams working in those languages

Version Check

Windows API is applicable to

- MS Windows
- The principles of calling native code are cross-platform, but the code you'll see is Windows specific

Windows API


.NET Environment
C# code
Compiler
Managed IL
(Intermediate language)
code
Garbage collection service
Base class library
The API is how apps talk
to Windows
Windows API
Huge library of native functions that interact with Windows
Windows Operating System

Windows API

Base class library only provides subset of Windows API features

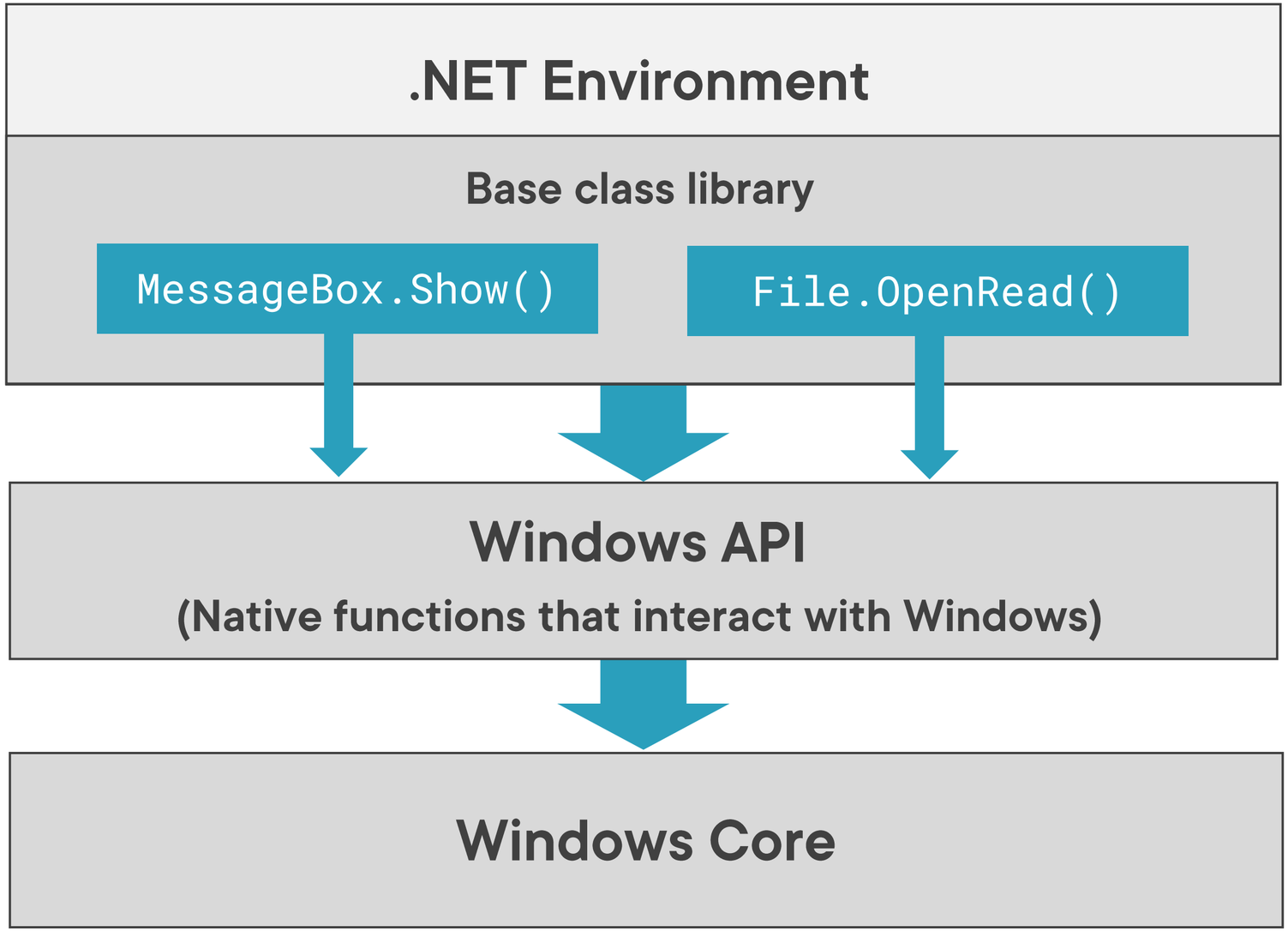
Call Windows API directly for everything else

For example:

Hardware access

Windows Shell advanced features

DirectX graphics



Calling Windows API Functions

MessageBox function (winuser.h)

Article • 10/13/2021 • 7 minutes to read

Displays a modal dialog box that contains a system icon, a set of buttons, and a brief application-specific message, such as status or error information. The message box returns an integer value that indicates which button the user clicked.

Syntax

C++ Copy

Demo

Calling `MessageBox()` API function

- P/Invoke layer
- Matching managed and unmanaged types

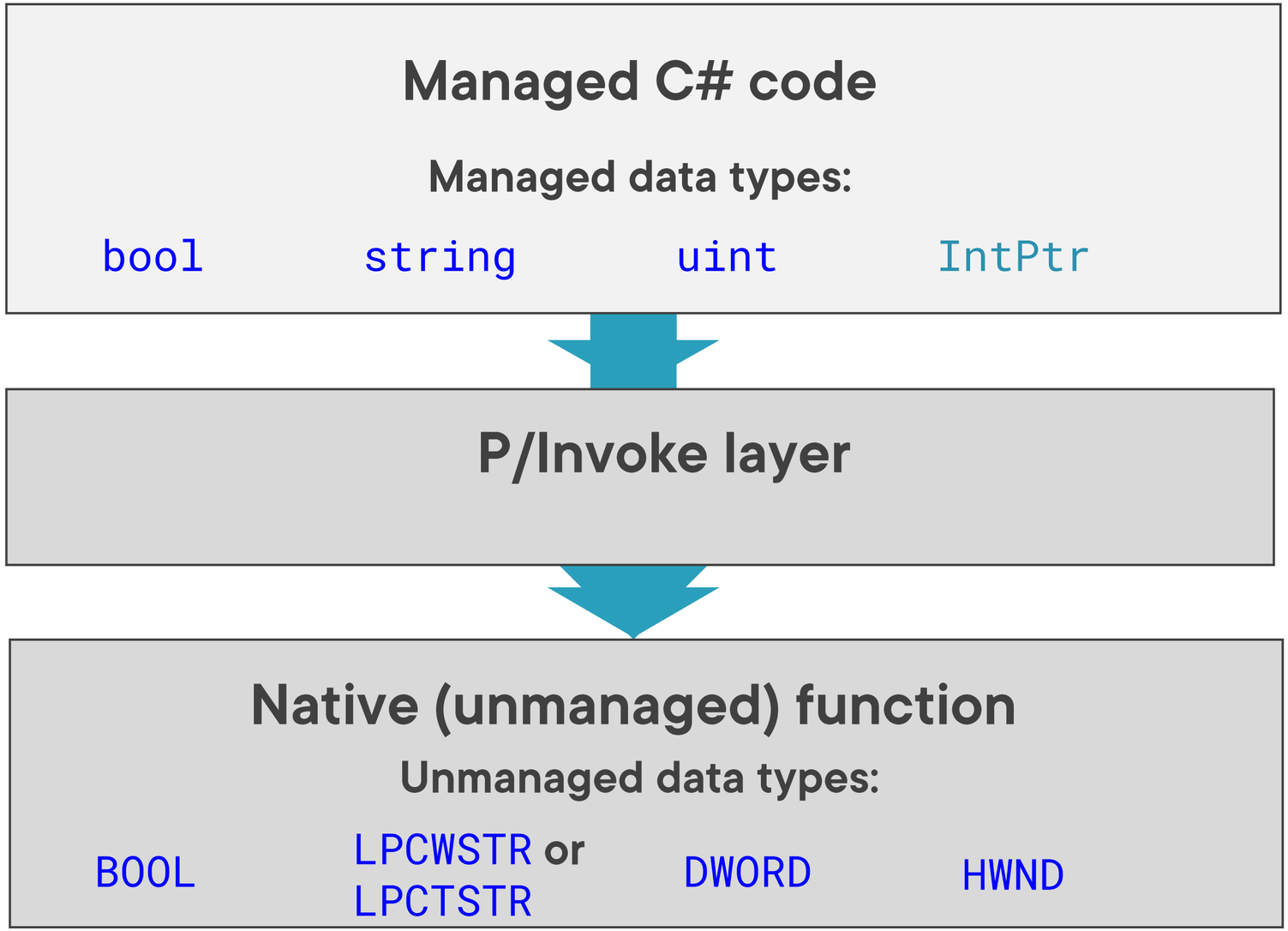
The Platform Invoke (P/Invoke) Layer

P/Invoke converts between managed and unmanaged types

This is called marshalling

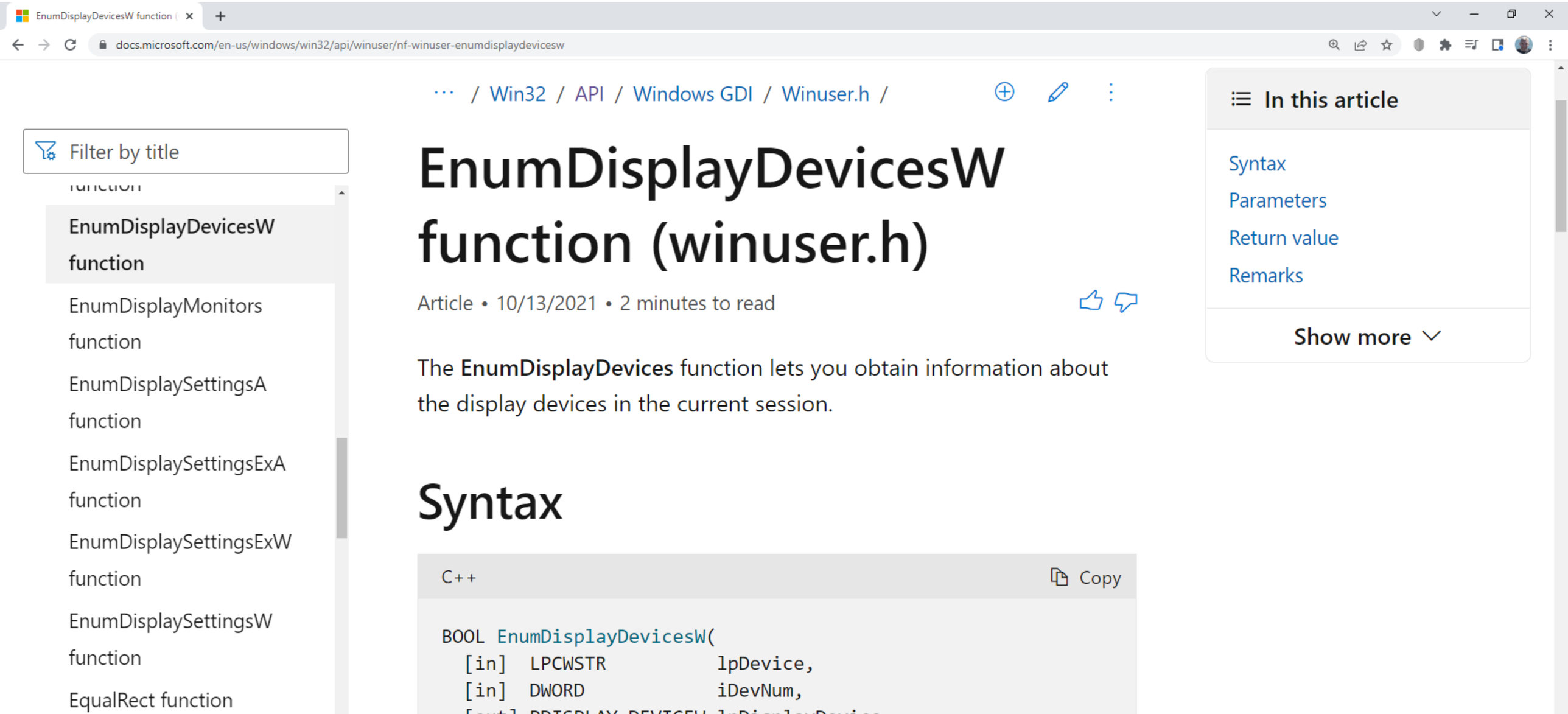
You must still select appropriate managed types

You'll need to recognise common unmanaged types



Returning Data from an API Function

Displaying Monitors and Graphics Cards

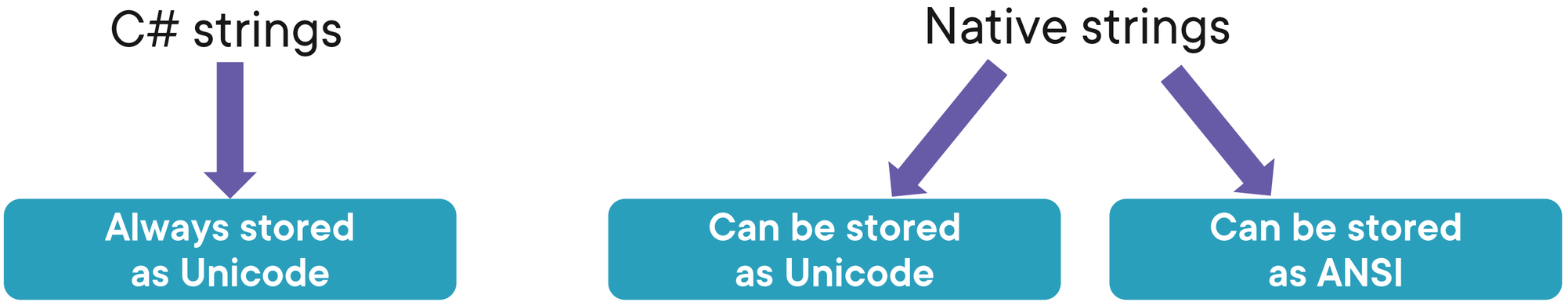


Demo

Display monitors and graphics cards

- Using `EnumDisplayDevicesW()`
- Unicode vs. ANSI API methods
- Define struct to hold return values
- Custom marshalling

Strings



ANSI	Unicode
Uses less memory	Much bigger character range
Better performance	Required for globalisation

EnumDisplayDevicesW function (winuser.h)

Article • 10/13/2021 • 2 minutes to read

The **EnumDisplayDevices** function lets you obtain information about the display devices in the current session.

Syntax

C++ Copy

```
BOOL EnumDisplayDevicesW(
  [in]  LPCWSTR          lpDevice,
  [in]  DWORD            iDevNum,
  [out] PDISPLAY_DEVICEW lpDisplayDevice,
  [in]  DWORD            dwFlags
```

Calling a Visual Basic Method

Demo

Outsource `DisplayAdapters()` method to VB

- This works just like calling a C# method

Calling an F# Method

Demo

Outsource `DisplayAdapters()` method to F#

- Works just like calling C# in theory
- But you need to be aware of type differences

Summary

Calling the Windows API

- Opens full Windows functionality to your C# apps
  - Including things not covered by the BCL
- Declare API methods as static extern
  - `[DllImport]`
- Work out appropriate types to marshall to unmanaged types
  - Use `[MarshallAs]` to guide marshalling
- Different coding techniques
  - Specifying data sizes
  - Calling a function multiple times

Summary

Calling other managed languages

- May be required if other teams use other languages
- VB and F#
  - Managed interop generally works out of the box
  - But beware that F# often uses different types