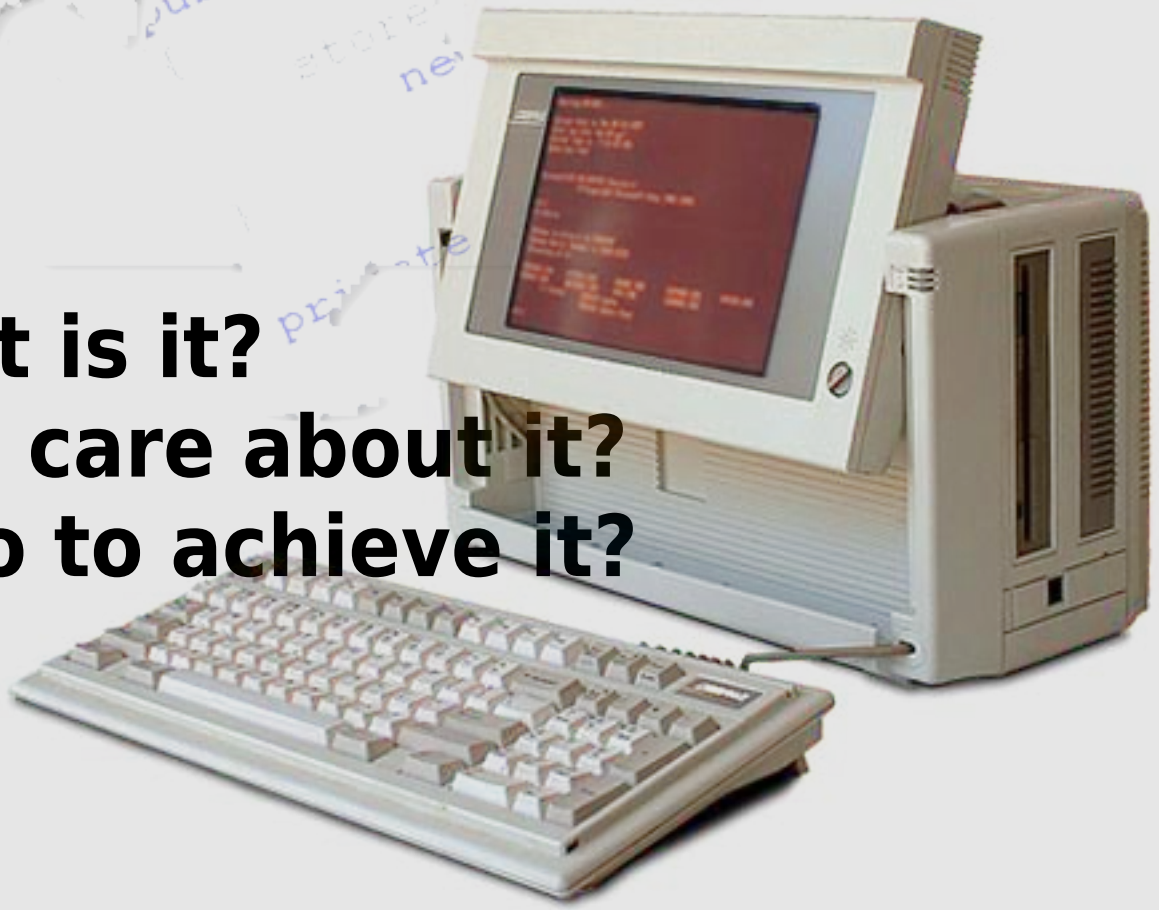
Why Portable Code

Steve Love

Steve Love
steve@arventech.com

Portable Code



What is it?
Why should I care about it?
What can I do to achieve it?

Steve Love
steve@arventech.com

What Is Portable Code?

Running on a mobile device?

Running on 2 mobile devices?

Running on Linux and MS Windows?

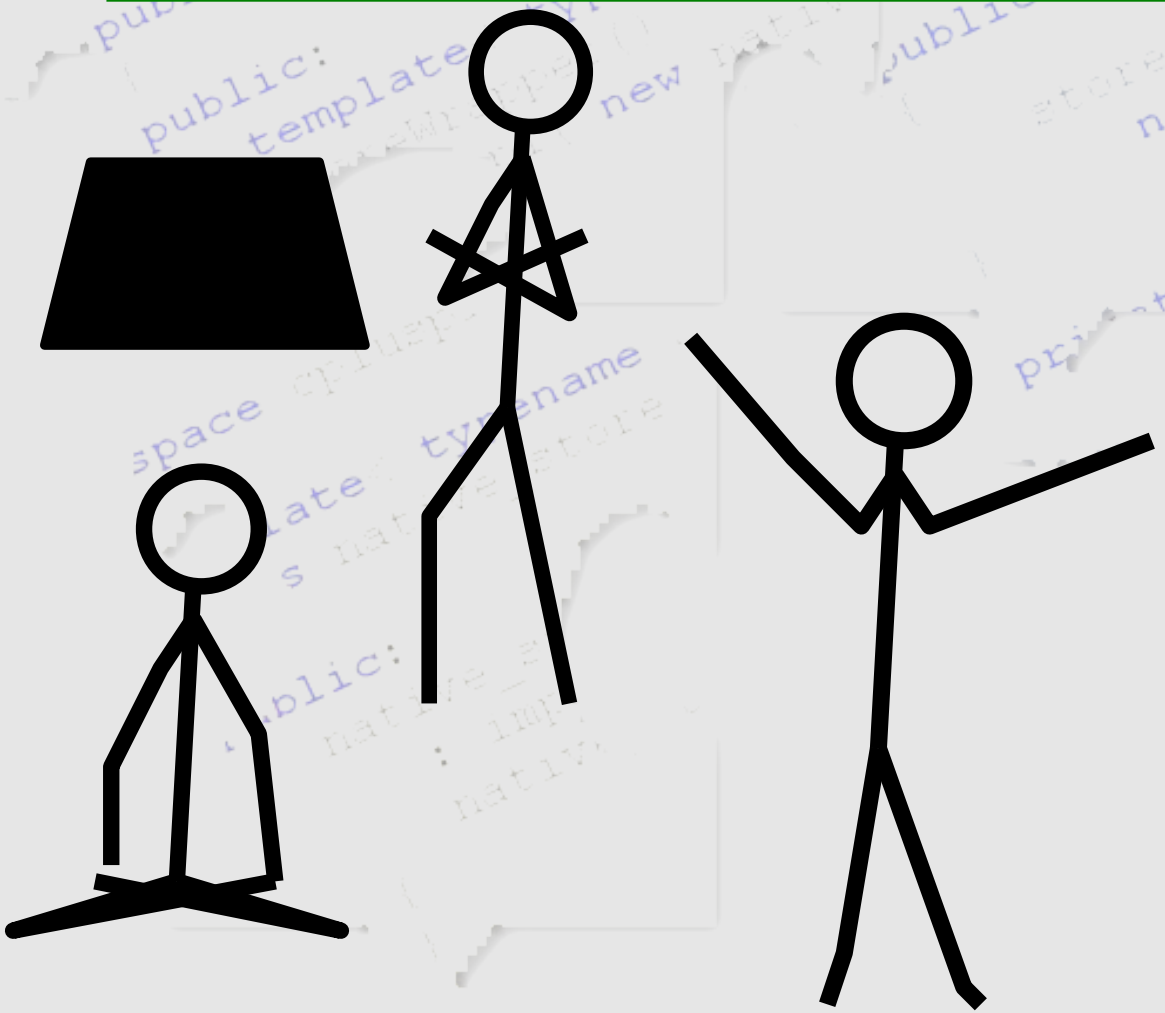
Cross-compiled for embedding?

Written in Java? Python? C?

Uses a “portable” toolkit
(what makes that portable?)

Steve Love
steve@arventech.com

Who Cares?



The Ignorant

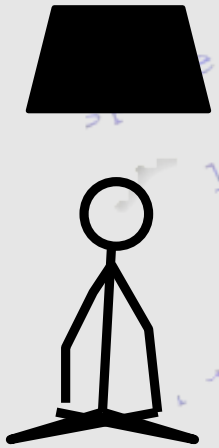
The Skeptic

The Zealot

Steve Love
steve@arventech.com

The Ignorant

Integers are intended to be 32bits wide.



I know my platform intimately.
I can always depend on it.

Well, it works on **my** machine.

Steve Love
steve@arventech.com

Skeptic

My Platform is the one true platform.

The features of my platform make writing software easier and more useful.

The benefits of platform specific features far out-weigh the effort of portability.

Everyone else should use the same platform as me (see (1) above)

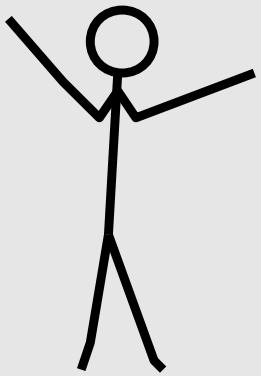
Steve Love
steve@arventech.com

The Zealot

Portability Is All.

All software I write *MUST* run on as many different platforms as possible.

Without alteration of any kind.



Steve Love
steve@arventech.com

What is a Platform anyway?

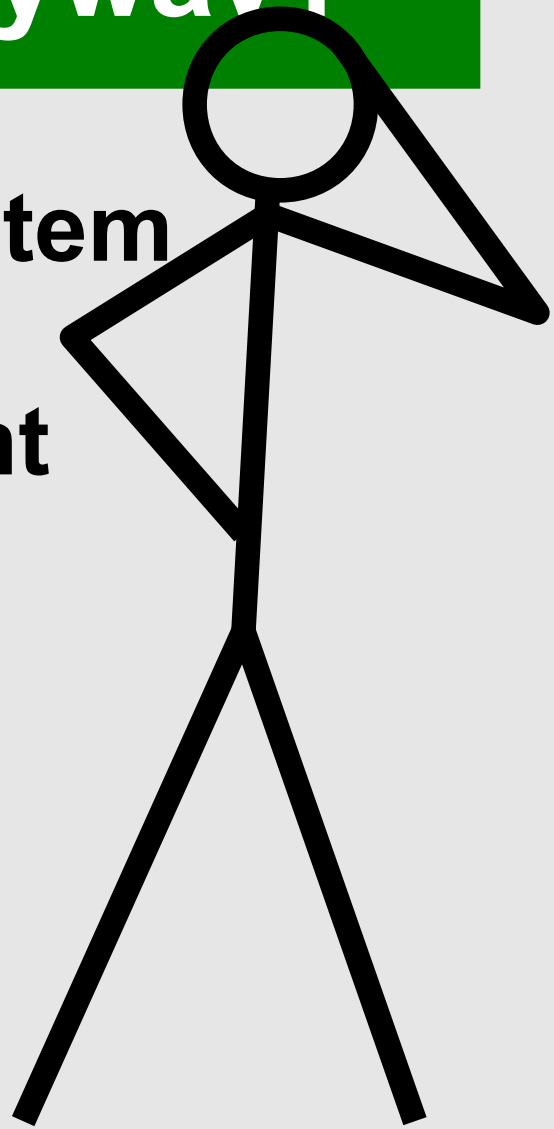
Hardware and Operating System

+

Programming Environment

+

Standard Libraries



Steve Love
steve@arventech.com

What About....?

Compiler and Library Versions
MSVC6 anyone?

Operating System Releases
XP to Vista?

Basic Platform Changes
32 – 64 bit

Steve Love
steve@arventech.com

The Common Myths

Portability is not possible for most applications

Portability is about running on any Platform

Portability means sacrificing efficiency

Steve Love
steve@arventech.com

Another Myth

My code is portable because it's written using...

Python
Java
SOA
wxWidgets
Boost

Steve Love
steve@arventech.com

So What?

Portability is more than platform independence

Standards are about more than platform independence
...but they do assist portability.

Steve Love
steve@arventech.com

Bottom Line

Portable Code is

SIMPLER
not
HARDER

To Create, Understand, Maintain

Steve Love
steve@arventech.com

Trade Off

Sometimes platform dependence is **desired.**

Sometimes non-portability is **unavoidable.**

How much and how far us up to you.

Steve Love
steve@arventech.com

Common Pitfalls

The Usual Suspects

Byte-ordering and data type sizes
Size of pointers
Endian-ness
Order of evaluation

Steve Love
steve@arventech.com

Example 1 – Compiler Upgrades

- Using Common Extensions
  - Calling Conventions
  - Dynamic Library Boundaries
  - Platform Libraries like Python COM
- Not Standard Enough
  - Stricter Checking
  - Deprecated or no-longer supported features

Steve Love
steve@arventech.com

Example 2 – Cross Compiled

- Portable Devices and Mobile Phones
  - Library support is often limited
  - What about testing? E.g. NUnit Bridge
  - Do you really want to debug on the device?
- Embedded Software
  - Hardware can be quite varied
  - Compiler and library support is often limited

Steve Love
steve@arventech.com

Example 3 – Testing

- Tests are a new platform
  - Make test code independent of its surroundings
  - Drop-in replacement features – UI, DB, etc. are all the platform specifics
- Design for Test == Design for Portability
  - Separate out that which you do not need.

Steve Love
steve@arventech.com

Example 4 - 3rd Parties

- Services May Change
  - Reuters vs. Bloomberg vs. ???
  - Windows Forms vs. [many]
  - RWTools.h
- SOA Abstracts Comms
  - Middleware is a platform of sorts
  - What if you change it?

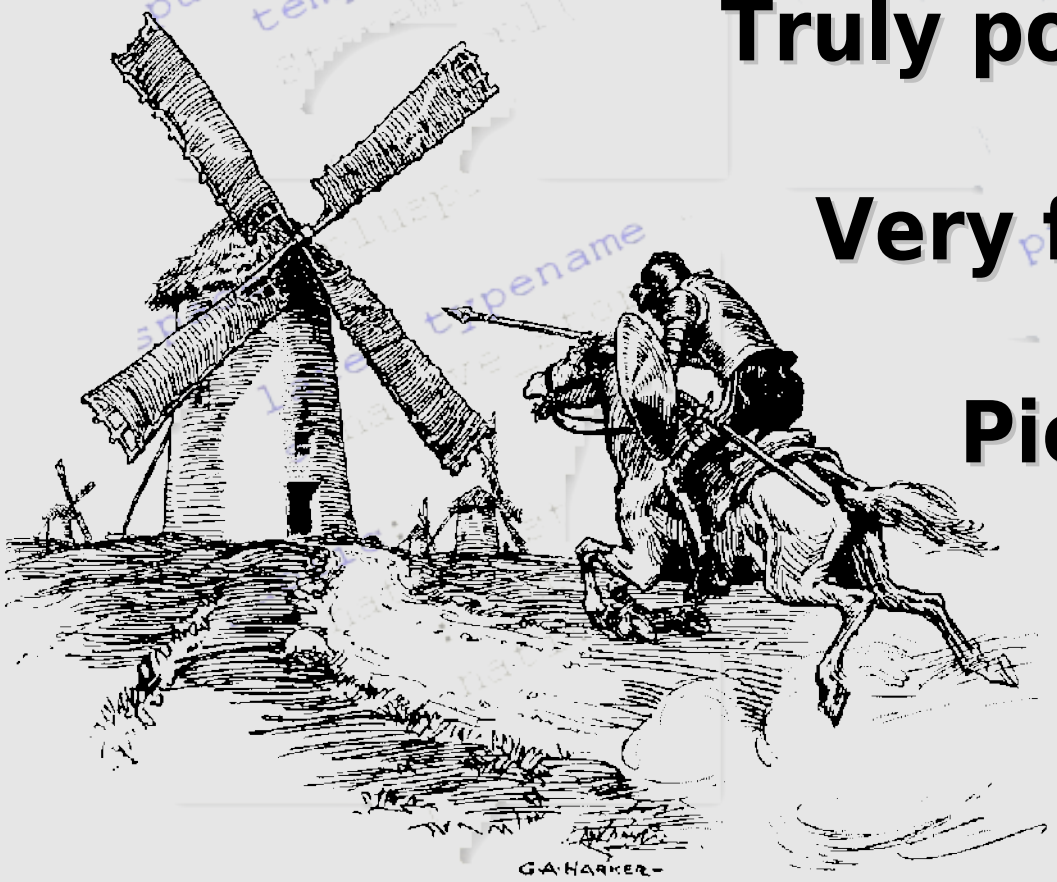
Steve Love
steve@arventech.com

Example 5 – Parallel Development

- Who Does What?
  - A different team?
  - A different organisation?
- When Does it Happen?
  - Their timescales may be different to yours
  - How do you code against something that ***doesn't exist yet?***

Steve Love
steve@arventech.com

The Good News



Truly portable code is hard.

Very few really need it.

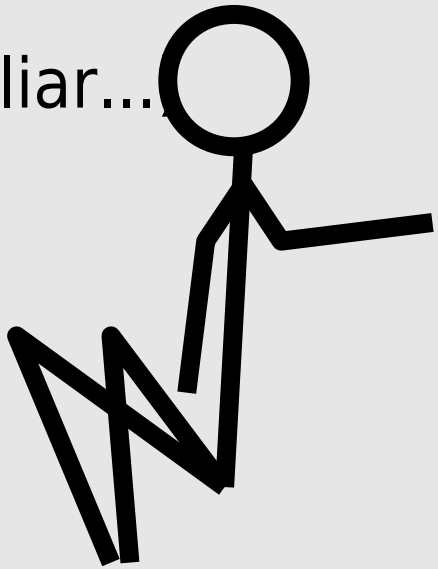
Pick your battles.

Steve Love
steve@arventech.com

So OK, I Care. Now What?

Some techniques are common to making ***ANY*** code more portable.

(By the way, these techniques should be familiar...)



Steve Love
steve@arventech.com

Modularity

Partition apps into modules
separated by interfaces

Make ALL interfaces 100% portable

Make as many portable modules as you can
Quarantine platform-specific code

Steve Love
steve@arventech.com

Standards

Code written according to published/international standards will be easier to port.

C++ ISO Standard
Java Language Reference
Common Language Interface

Steve Love
steve@arventech.com

Standards

Where you need to depart from The Standard,
(whatever that is for you)
make your interface boundaries there.

Steve Love
steve@arventech.com

People

Clever tricks in code make it harder to find new programmers to understand it!

Template Meta-Programming
Using lots of different and arcane languages.

People are an important part of your development environment.

Steve Love
steve@arventech.com

People

Not all “Clever Tricks” are just tricks.

Treat such code as if it's a different platform and provide an easier-to-use interface to it.

Steve Love
steve@arventech.com

Structure

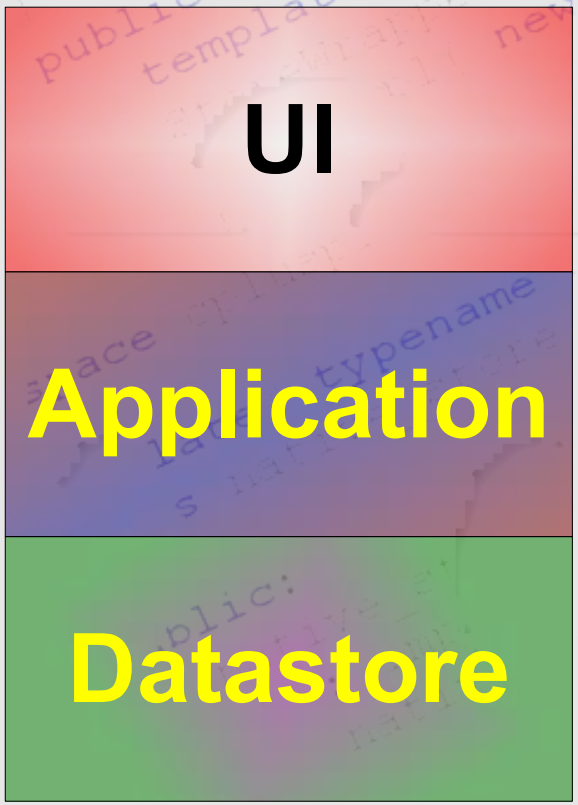
Loose Coupling and Strong Cohesion
make code more portable.

Hide platform specifics behind interfaces.

Allow for variation points – using interfaces.

Steve Love
steve@arventech.com

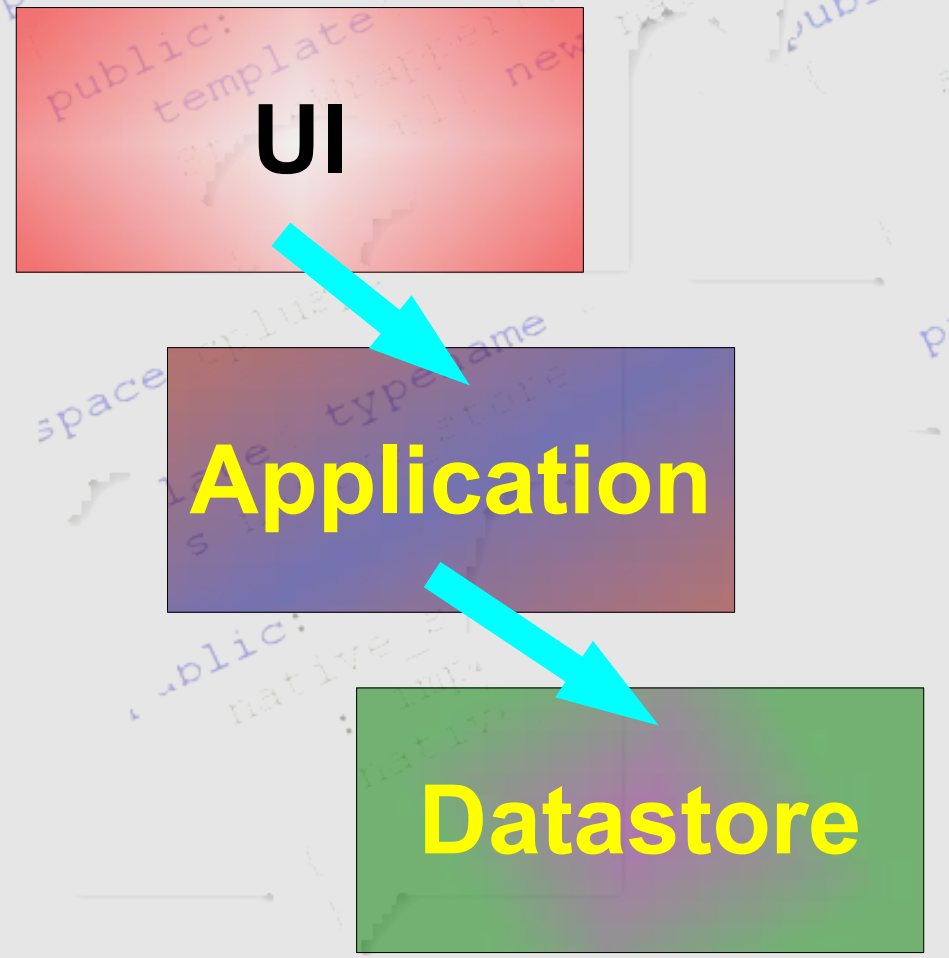
Coupled



- Monolithic
- Everything depends on its neighbour
- Modules are inextricably linked together
- Ultimately, the UI depends on the Datastore

Steve Love
steve@arventech.com

Less Coupled

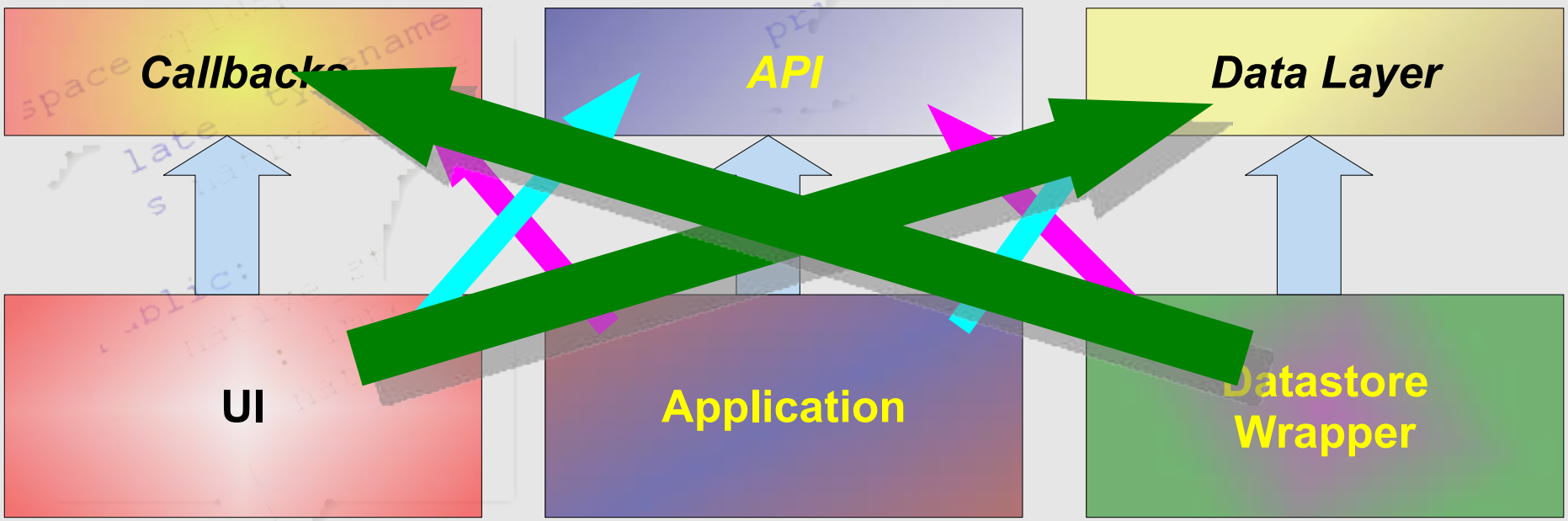


- Separate modules
- When it's one way (down) all is fine
- What happens if the Application needs to call-back on the UI?
- UI ***still*** depends on the DB

Steve Love
steve@arventech.com

Decoupled

Objects depend only on interfaces
Interfaces depend only on other interfaces
Notice how dependencies go *upward*!



Steve Love
steve@arventech.com

Why Portable Code?

Why NOT Portable Code?

Steve Love
steve@arventech.com

Why Portable Code?

- Portability is not just platform independence
  - Platform can mean many things.
  - Platform independence may be important
    ...or it may not be
- Your people are important
  - think about the programmers you have today
  - ...and those you'll need tomorrow

Steve Love
steve@arventech.com

Why Portable Code?

Program Structure Is All

Separate Interface From Implementation
Hide clever tricks, optimisations, complexity, platform specifics behind interfaces

Design for portability – like testing –
results in better design

Steve Love
steve@arventech.com

In The End....

It's really about making it simpler.