3/11/2013, 台北I

Gordon Li

# RAD STUDIO XE 3.5 IN ACTION *LIVE!*

# Agenda

- Welcome and Overview
- Multi-Device App Development
- 10 Times Faster's Productivity - FireMonkey For iOS
- Break
- Super Powerful Engine For Your Codes - New And Next Generation Compilers for C++Builder And Delphi
- Smart Data Access For your Smart Devices - Visual LiveBindings
- Summary and Q&A

# MULTI-DEVICE APP DEVELOPMENT

# Embarcadero
## Developer Tools Strategy 2013+

- "Multi-Device" App Development
- "Real Code" Native Device Machine Code
- Enterprise Ready
- Standard Languages

# 16 Million Developers World Wide*

- 52 Billion lines of code per year
- $28 USD avg per line of GA code
- $1.5 Trillion WW per year spent developing source code

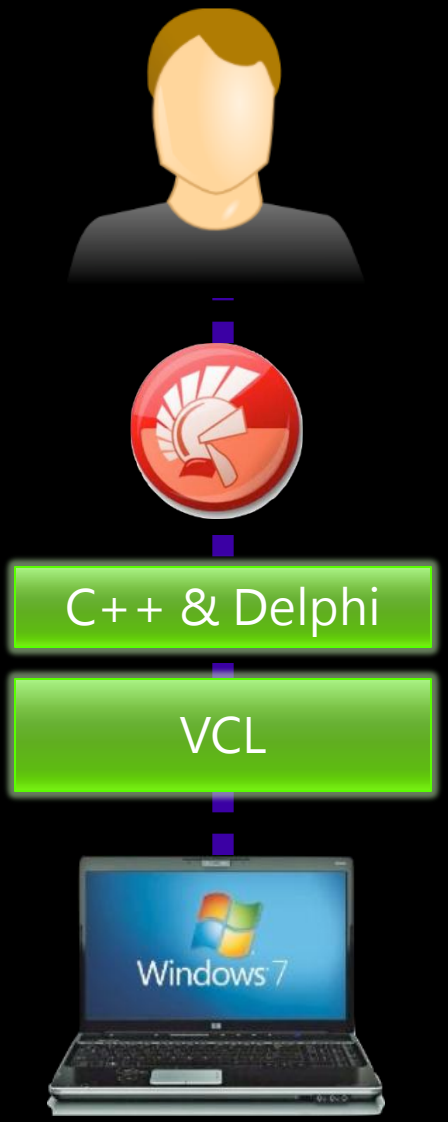
*Evans Data

# Embarcadero Developer Value

Up to 80% less code per project


- 3m Developer Community
- 9.7 Billion LOC per year - Saving 48 billion LOC/yr
- effective $5.60 per line of code (vs industry std $28/LOC)
- $220 Billion in annual developer savings

# Embarcadero's Method to Success on Windows:




80% code savings via

- Component Base Frameworks
- Visual Development
- Simplified abstraction of WinAPIs
  - Eg. collapse 100 lines of common Win API / MFC functionality into a single method or property
- RAD infused standard languages
- Make hard things easy

# Evolution of the Client Landscape

1999


Windows

2005


Windows & Web

2

W

# 2013: The Client Revolution

1 Billion


Windows

65 Million

Mac

1 Billion

Mobile

2 Billion

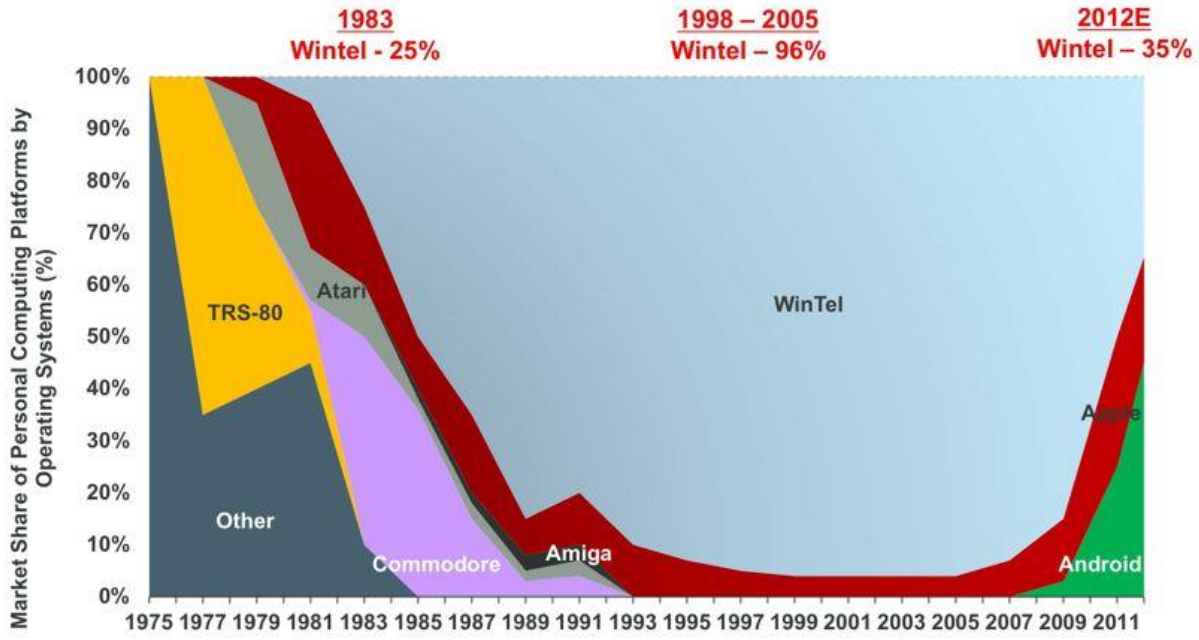
Web

Today’s Unprecedented Multi-Device Landscape

# The Client Revolution

## Re-Imagination of Computing Operating Systems - iOS + Android = 45% Share vs. 35% for Windows

**Global Market Share of Personal Computing Platforms by Operating System Shipments, 1975 – 2012E**




KPCB

Source: Asymco.com (as of 2011), Public Filings, Morgan Stanley Research, Gartner for 2012E data. 2012E data as of Q3:12.



An Unprecedented Multi-Device Landscape

# When will Tablets surpass Notebooks?

July 2012

~~Analysts Predict Tablets will surpass Laptops~~ in 2016

September 2012

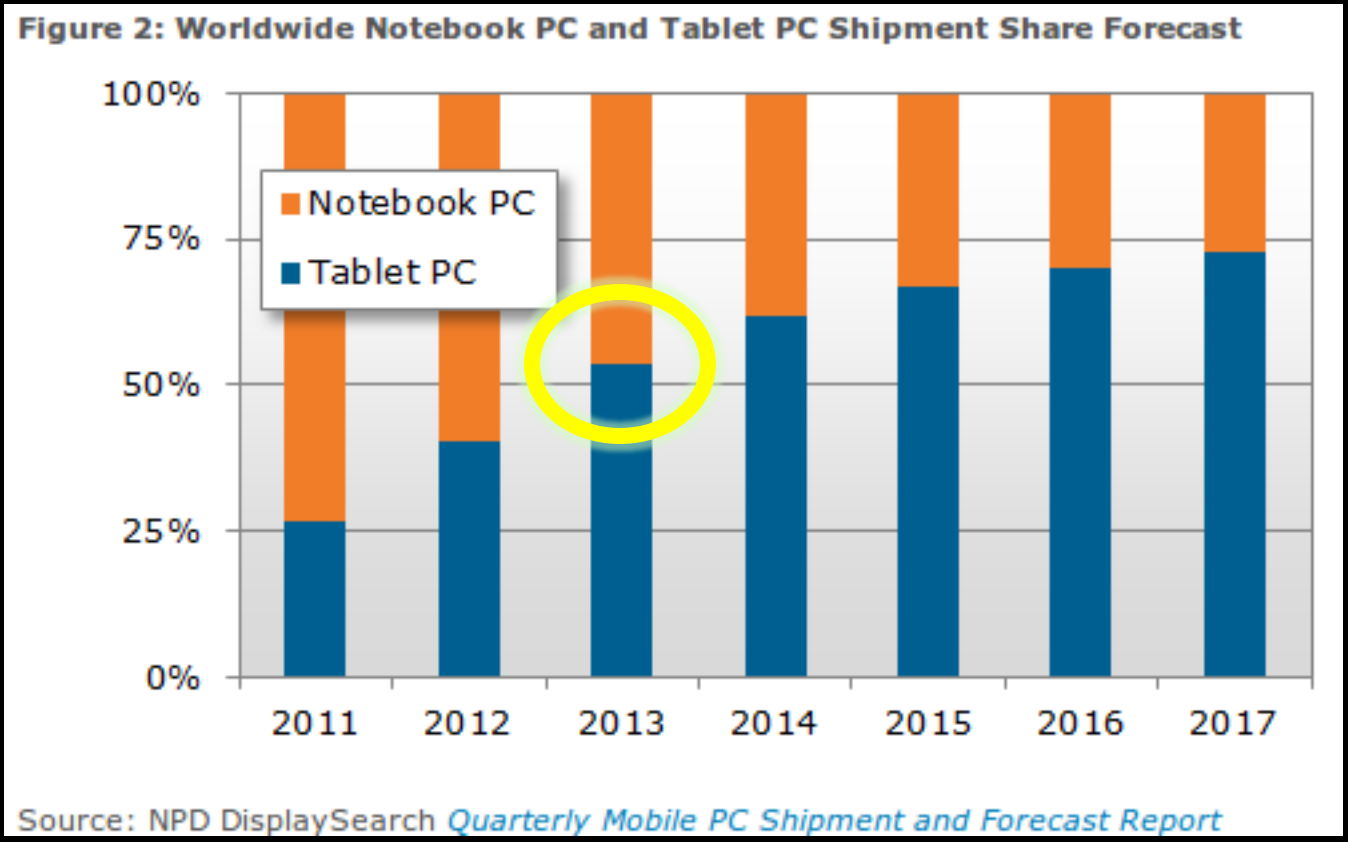
~~Analysts Predict Tablets will surpass Laptops~~ in 2015

January 2013

Analysts Predict Tablets will surpass Laptops in ……

# When will Tablets surpass Notebooks?

## This year.



# 2013+


# Client Device Diversity Will Continue to Expand

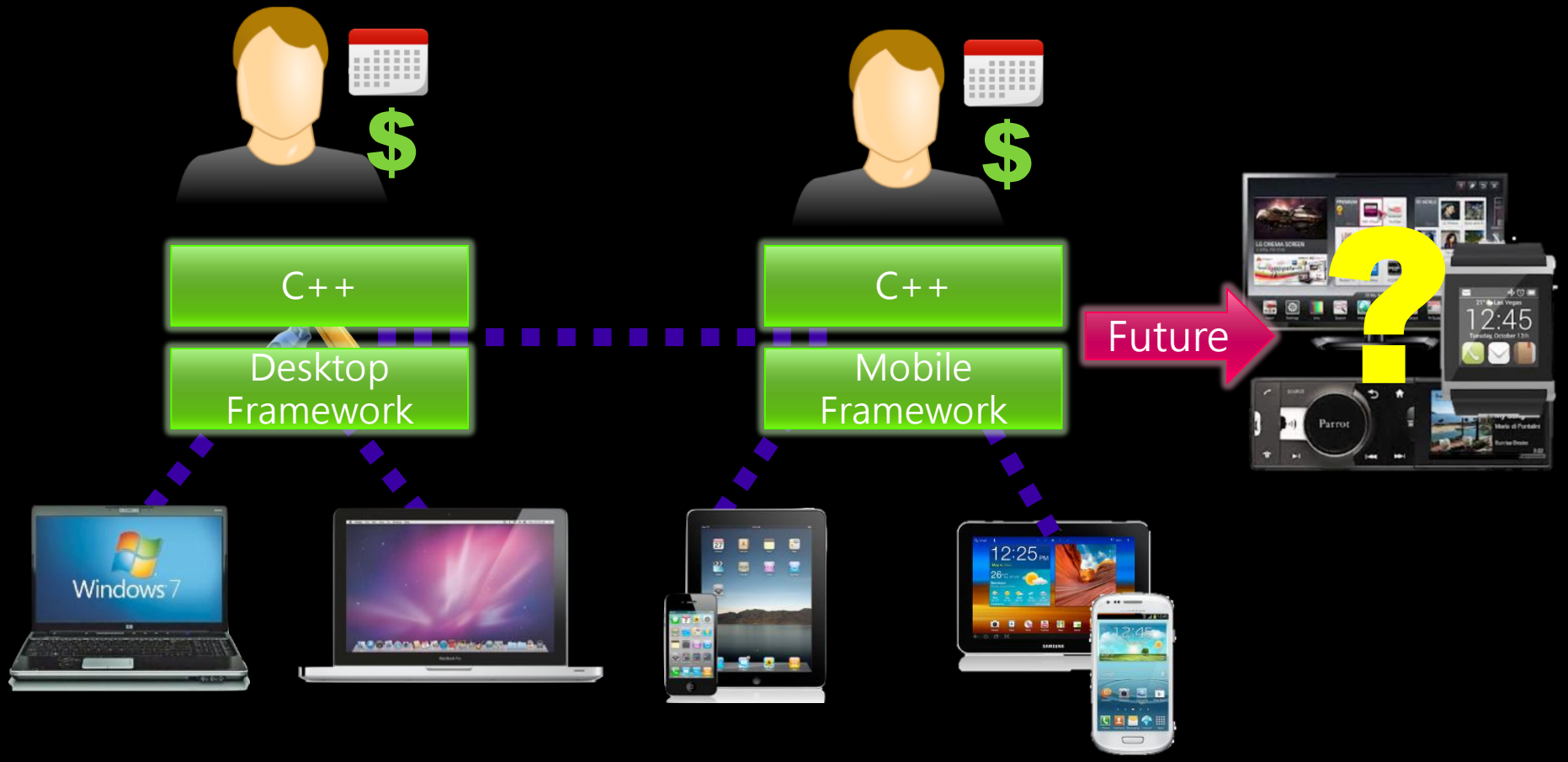
Rapid Multi-Device Development

vs

Vendor Tools
Traditional Cross-Platform
HTML5
“Platform Native” Virtual Code

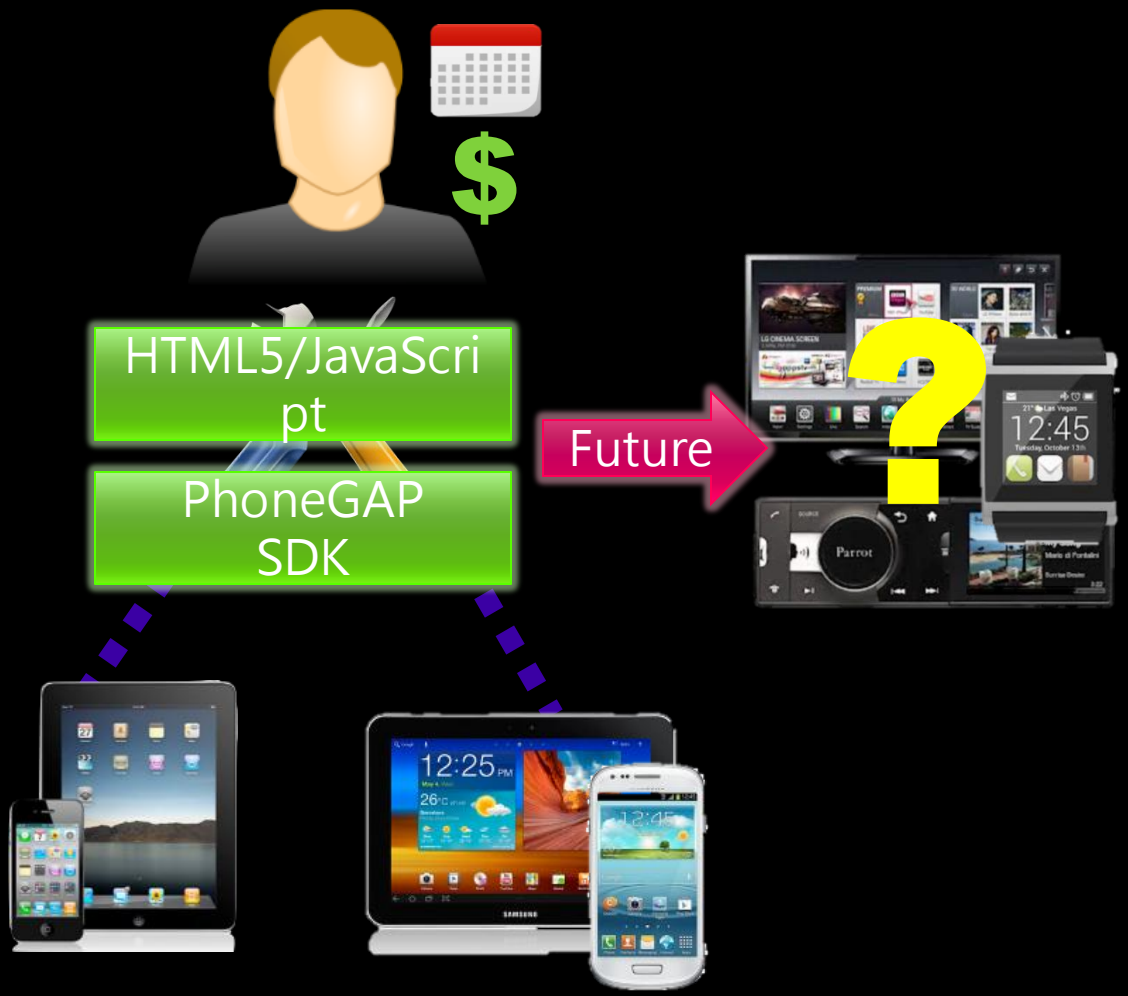
# Vendor Tool Approach for Targeting Multiple Devices

# Traditional "Like to Like" Cross-platform



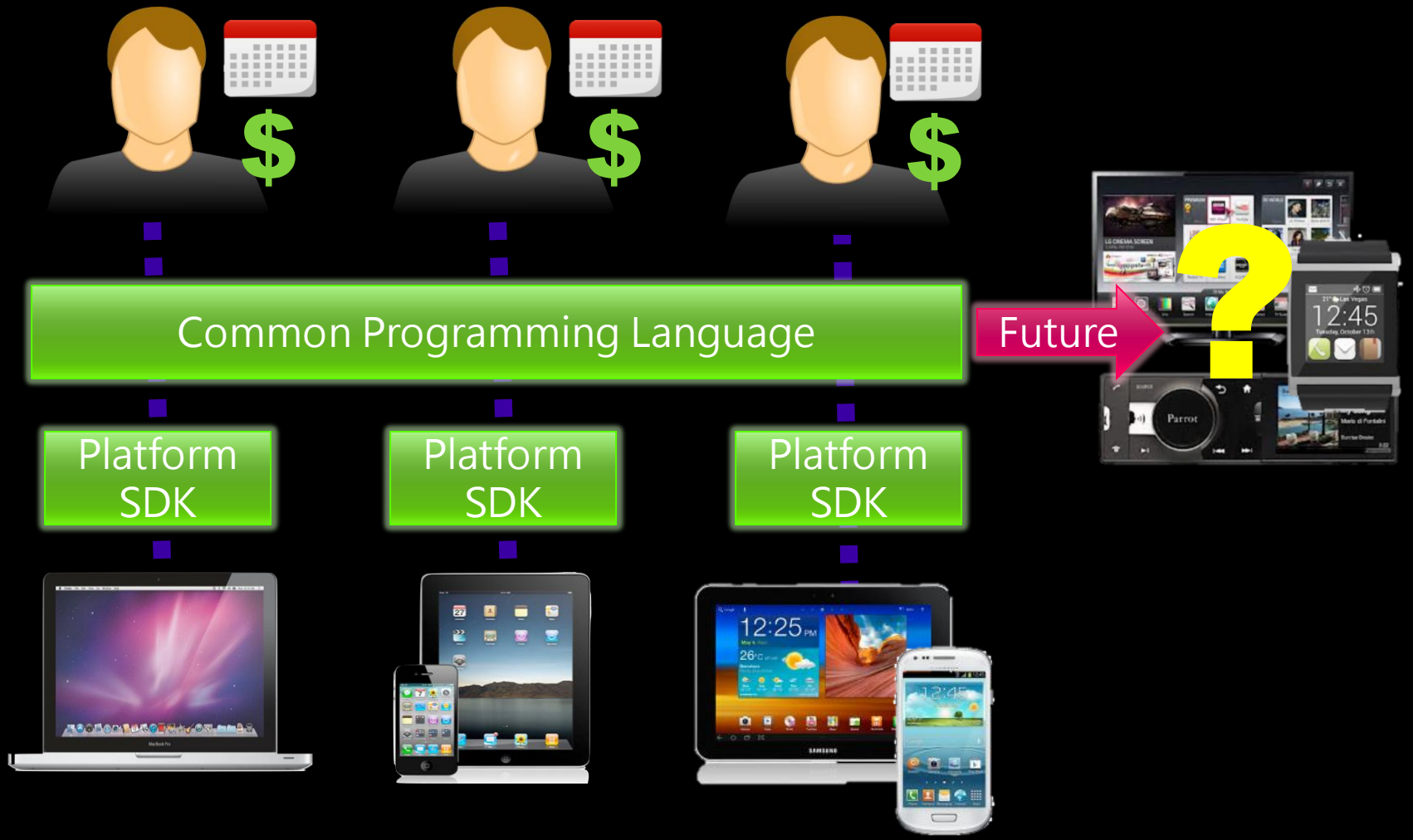

QT, WXWidgets

# HTML5 Cross-platform




Adobe, Sencha, Kendo, HTML5Builder

# So-called "Platform Native" virtual code



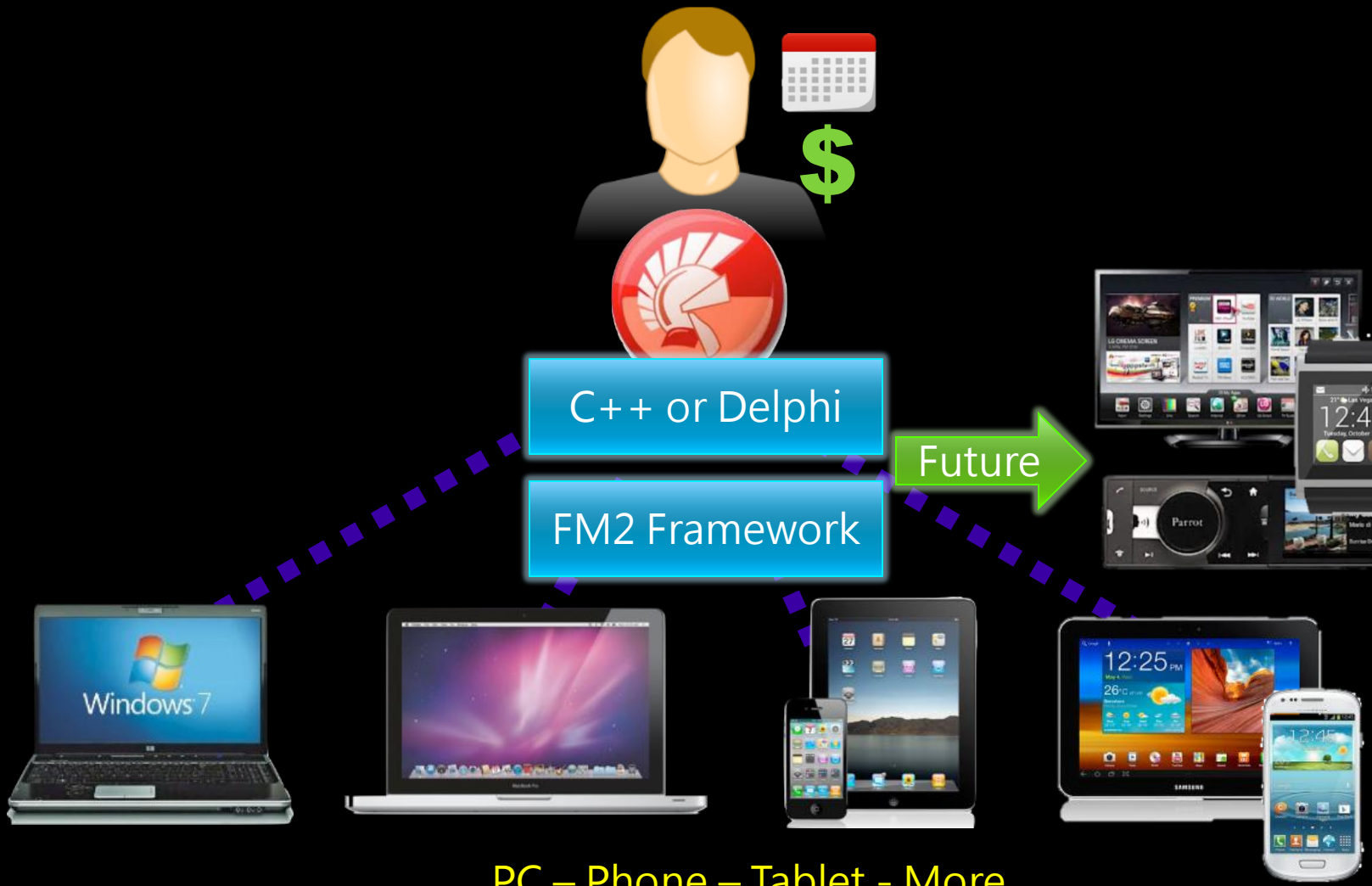

Appcellerator, Xamarin Mono, Oxygene (Prism)

Introducing…

# Multi-Device Development

Single Source Native Targeting Both PC and Mobile Devices

# Embarcadero: Multi-Device App Development



# Multi-Device

Platforms: Windows, Mac, iOS, Android

Form factors: PC, Phone, Tablet, Mini-Tablet, iPod, Phablet

Audience: ISV to Enterprise – targeting multiple devices

Connectivity: Enterprise Database Connectivity & Flexible Middleware

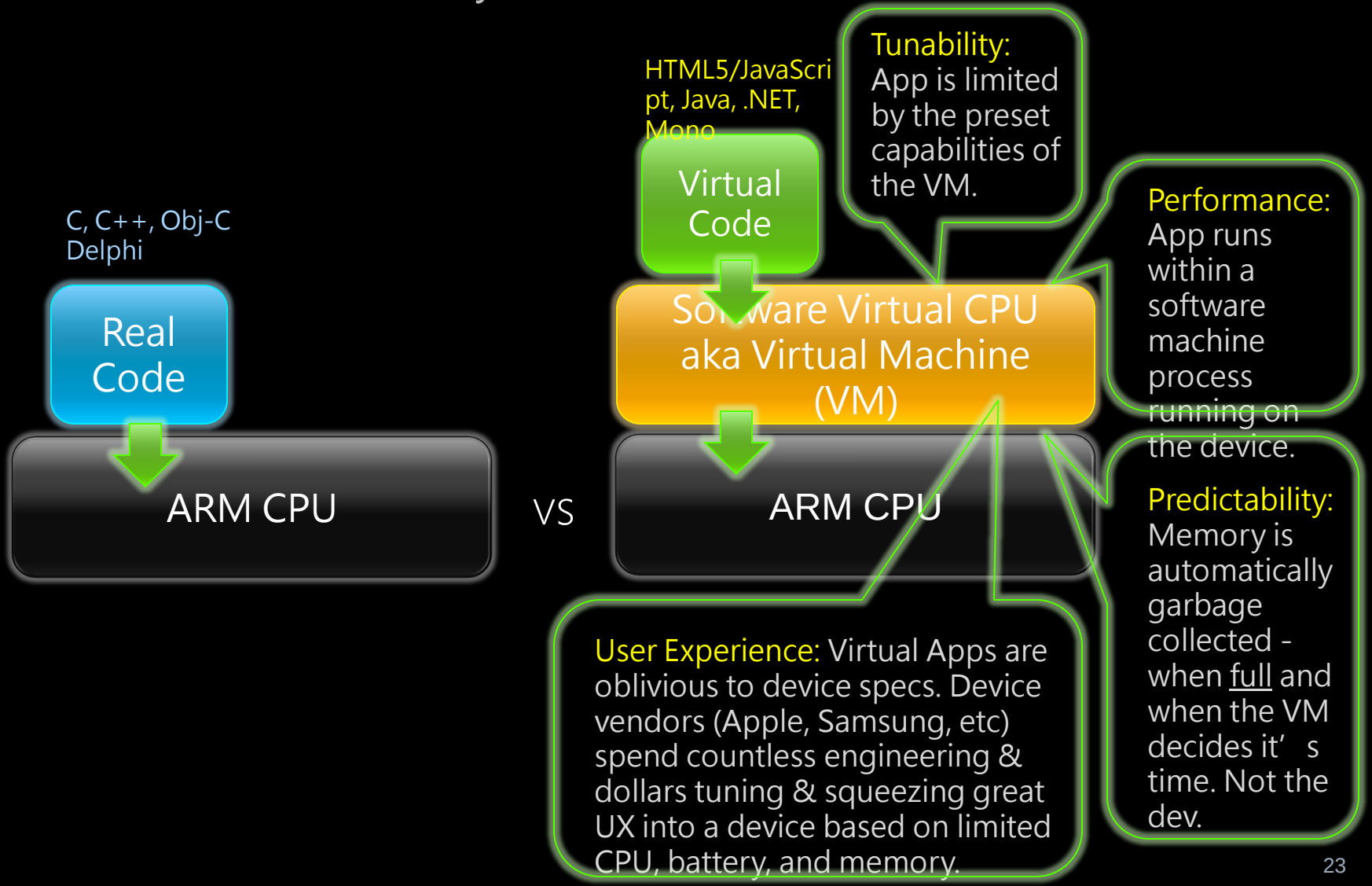
2013+

SmartTVs, Car Infotainment, Home Automation, Smart Watches, and more

# Real Code
# vs
# Virtual Code

# Real Code vs Virtual Code

"Anyone remember VB?"



## Real Code

Native Device Applications
Intel/ARM Machine Code
Maximum Performance
Highly tunable
Smallest Possible Footprint
Low Latency
Developer sched mem mgmt

## Languages:

C++, Obj-C, C, Delphi

## Best Suited for:

User/Client Apps
Embedded Applications
Real-time Applications

## Virtual Code

Virtual Device Application
Virtual Machine Code
Medium to Slow Performance
Limited tunability
Large Runtime Footprint
Med to High Latency
Runtime scheduled mem mgmt

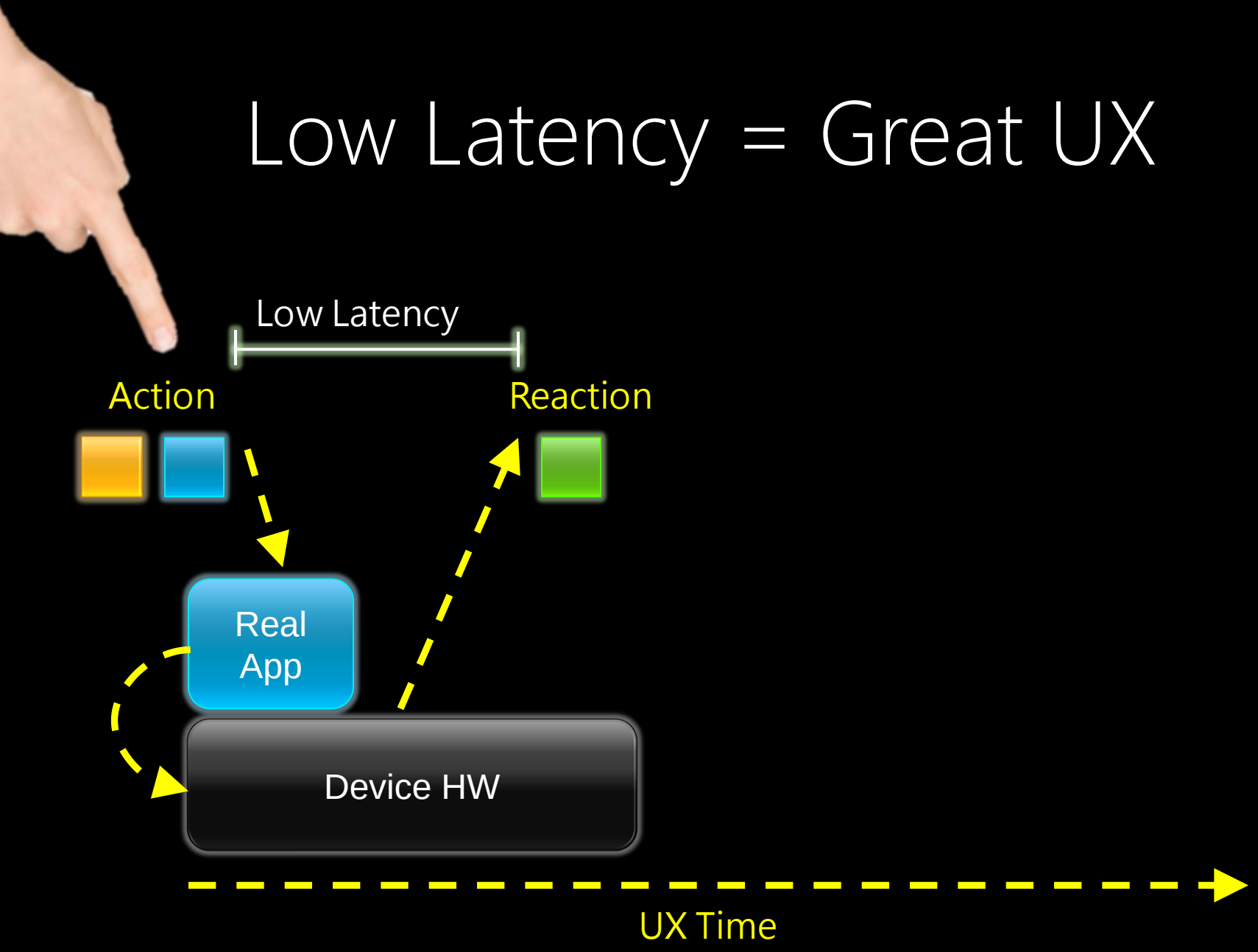
## Languages:

HTML5/JavaScript, C#, Oxygene/Prism, Java, Mono C#

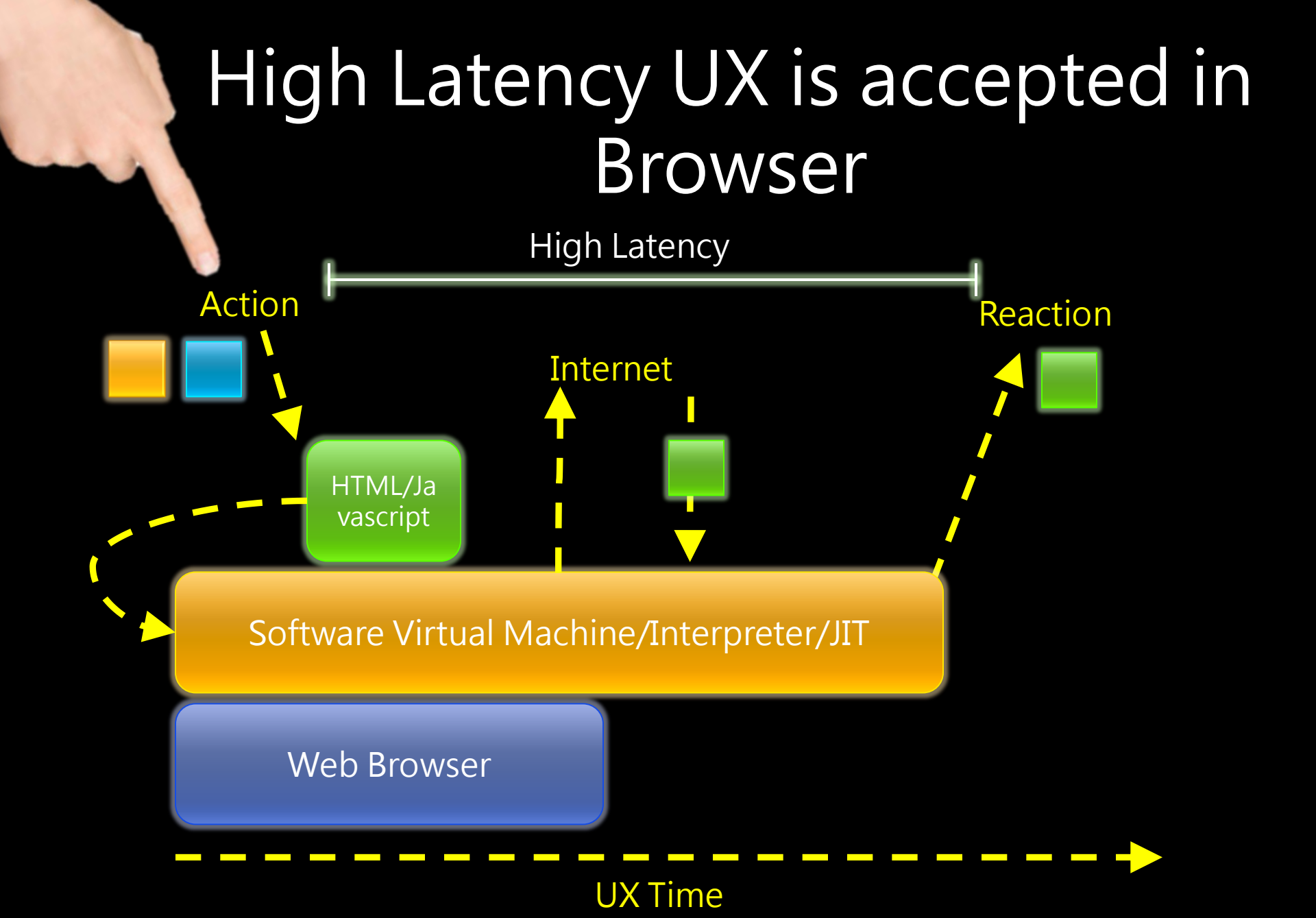
## Best Suited for:

Web Server Applications
Browser based Applications

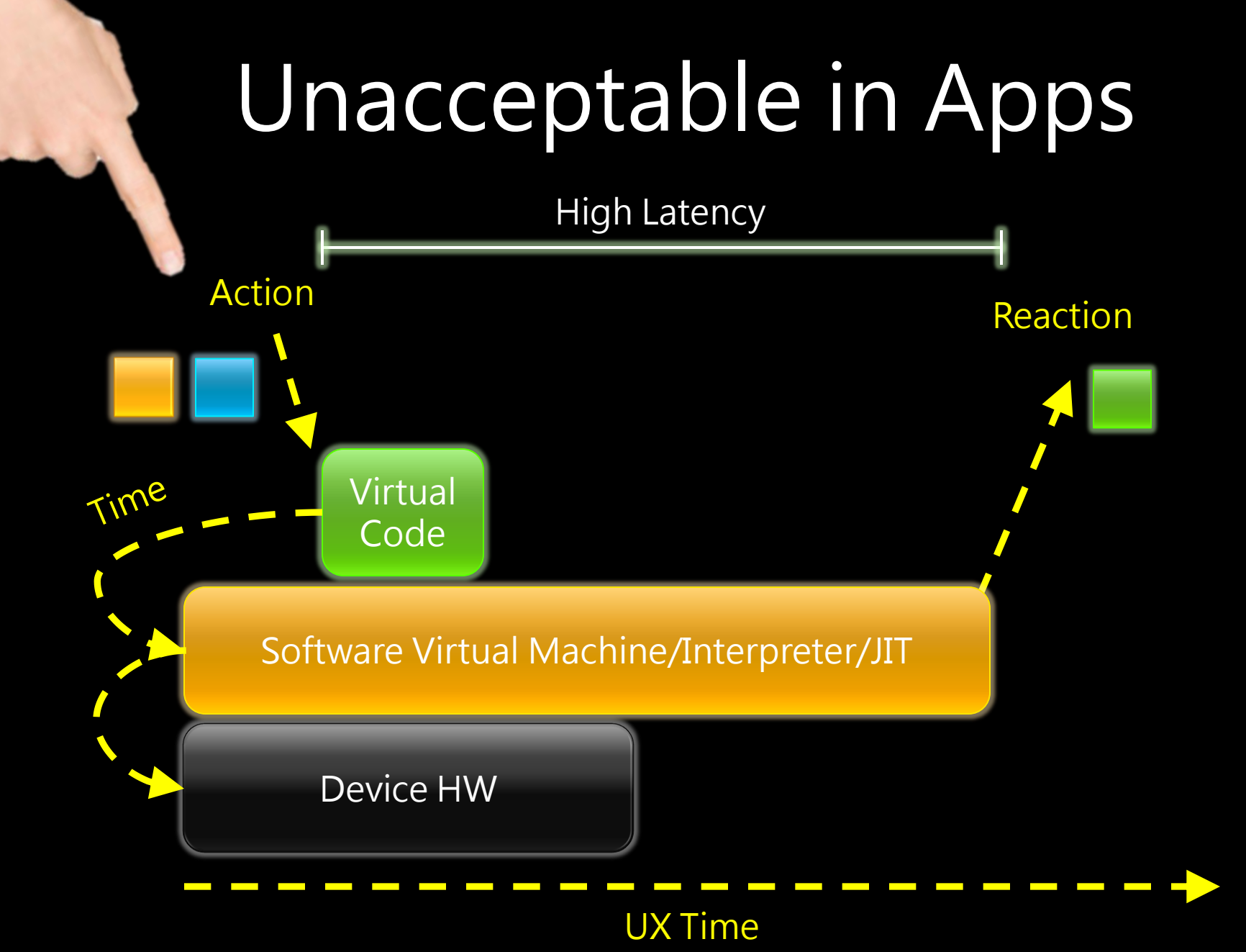
# Low Latency = Great UX



# High Latency UX is accepted in Browser



# Unacceptable in Apps



*“The biggest mistake we made as a company was betting too much on HTML5 as opposed to native”*

*Mark Zuckerberg - Facebook CEO*

*“Apple unbundling the (Java) runtime will erase a large number of security vulnerabilities”*

**AppleInsider 2011**

*"One of the biggest advantages we've gained from building on native iOS has been the ability to make the app fast.*

**Jonathan Dann – Facebook 2012**

"We are currently unaware of a practical solution to this (Java vulnerability) problem”

**U.S. Department of Homeland Security 2013**

# Virtual Code is for Servers & Browsers

## Servers & Browsers

Code Safety & Protection is Paramount
Ability to Scale Performance via HW
Web UX typically network bound

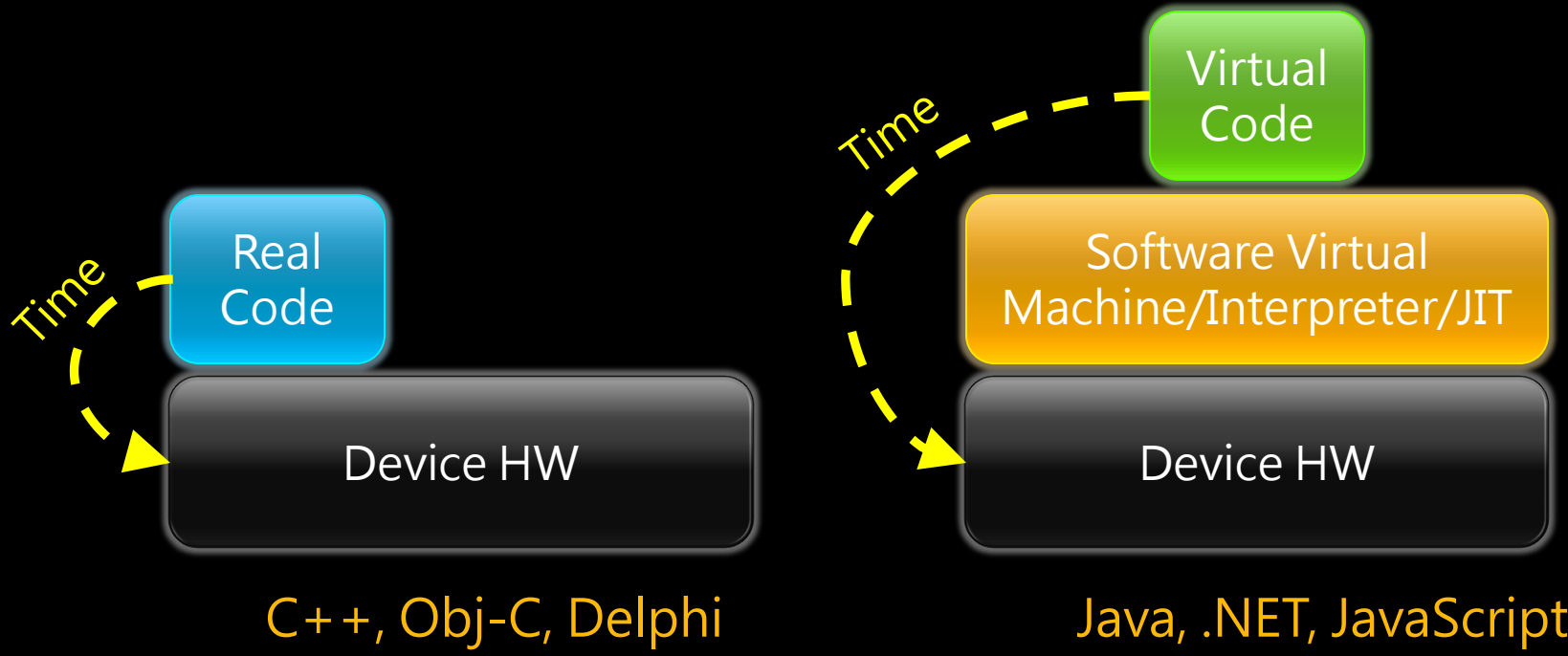
Java, .NET, HTML5, JavaScript

## Client Devices

UX Performance is Paramount
Fixed/Deployed HW Profiles
UX typically HW bound

C++, Obj-C, Delphi

# Real "Native" Code = Great App UX



# Embarcadero: Multi-Device App Development

| | HTML5 Cross Platform | "Platform Native" | Platform Vendor Tools | Rapid Multi-Device |
|---|---|---|---|---|
| Examples | Adobe, Sencha, Kendo, HTML5Builder | Appcellerator, Xamarin Mono | XCode, Visual Studio, Eclipse | Embarcadero RADStudio |
| Platforms | iOS/Android | iOS/Android | Win or Mac/iOS or Android (Sep IDE, lang, & SDK for ea platform) | Mac/Win/iOS & Android* (2013) |
| Native "Real Code" | No | No | Yes | Yes |
| Native Platform API Access | No (PhoneGap) | Yes | Yes | Yes |
| Single Source Multi-Vendor Targeting | Yes | No | No | Yes |
| Single IDE | Yes/Plugin | Yes/Plugin | No | Yes |
| Single Project | Multiple | Multiple | Multiple | Yes |
| App Performance | Low | Low | High | High |
| App Number Crunching Power | Low | Low | High | High |
| App Capacity (mem/data) | Low | Low/Med | High | High |
| App UX (User Experience) | Low/Med | Med | High | High |
| Enterprise Connectivity | Low | Low | High | High |

# Real Code

High Performance: Ideal for number crunching

Low Latency: Highly responsive native user experience (UX)

Control Hardware: Talk directly to ext peripherals and gadgets

Predictable: Developer is in control of App performance

Small Footprint: Ideal for small fixed size devices

# Standard Languages

Industry Standard: C++
Easy to Learn: Delphi
High Performance: C++ and Delphi
Millions of Developers: C++ and Delphi

# NATIVE IOS DEVELOPMENT WITH DELPHI

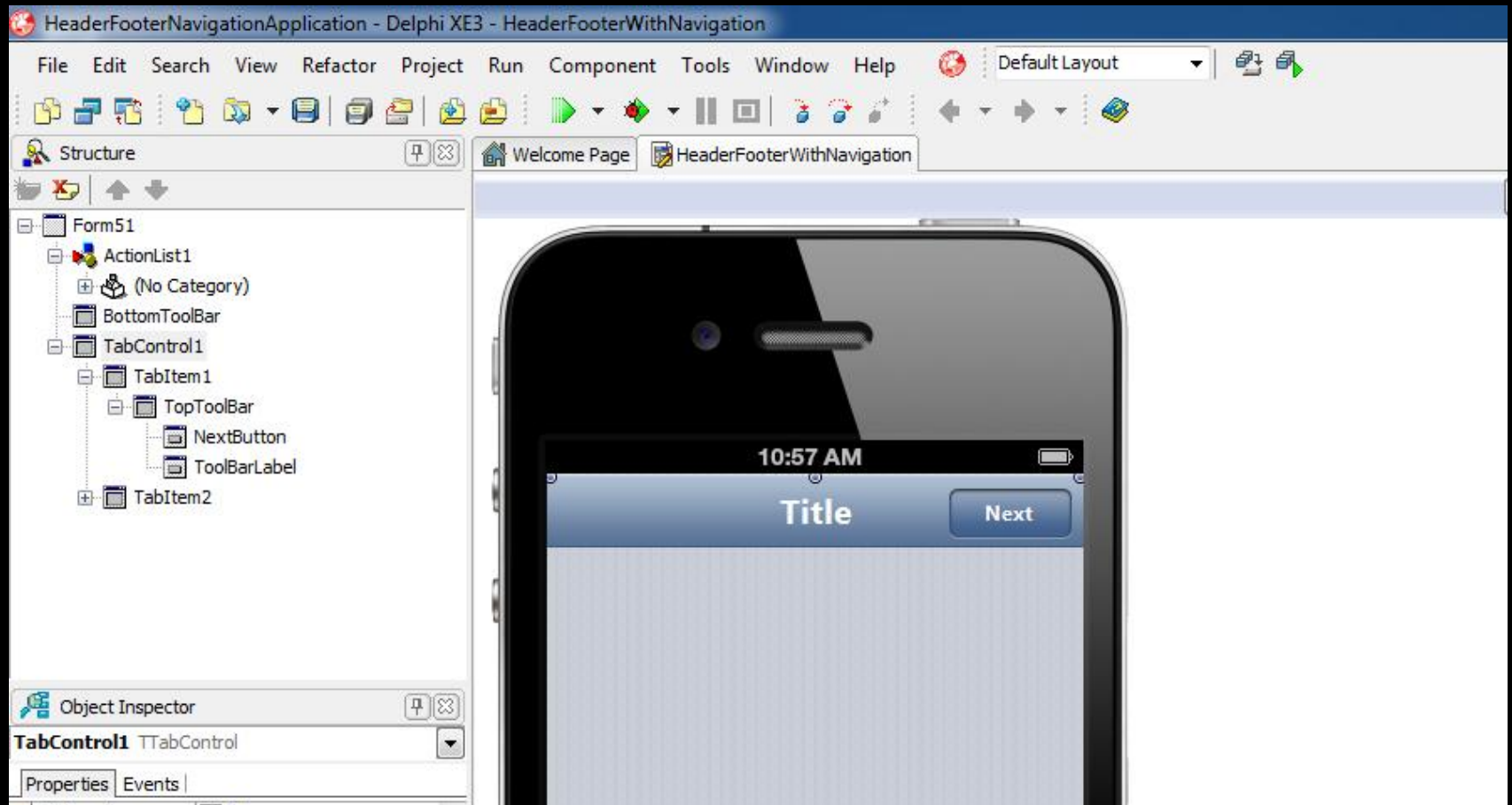
- 10 TIMES FASTER'S PRODUCTIVITY FIREMONKEY FOR IOS
- SUPER POWERFUL ENGINE FOR YOUR CODES - NEW AND NEXT GENERATION COMPILERS FOR C++BUILDER AND DELPHI

# When we launched XE2, we say…

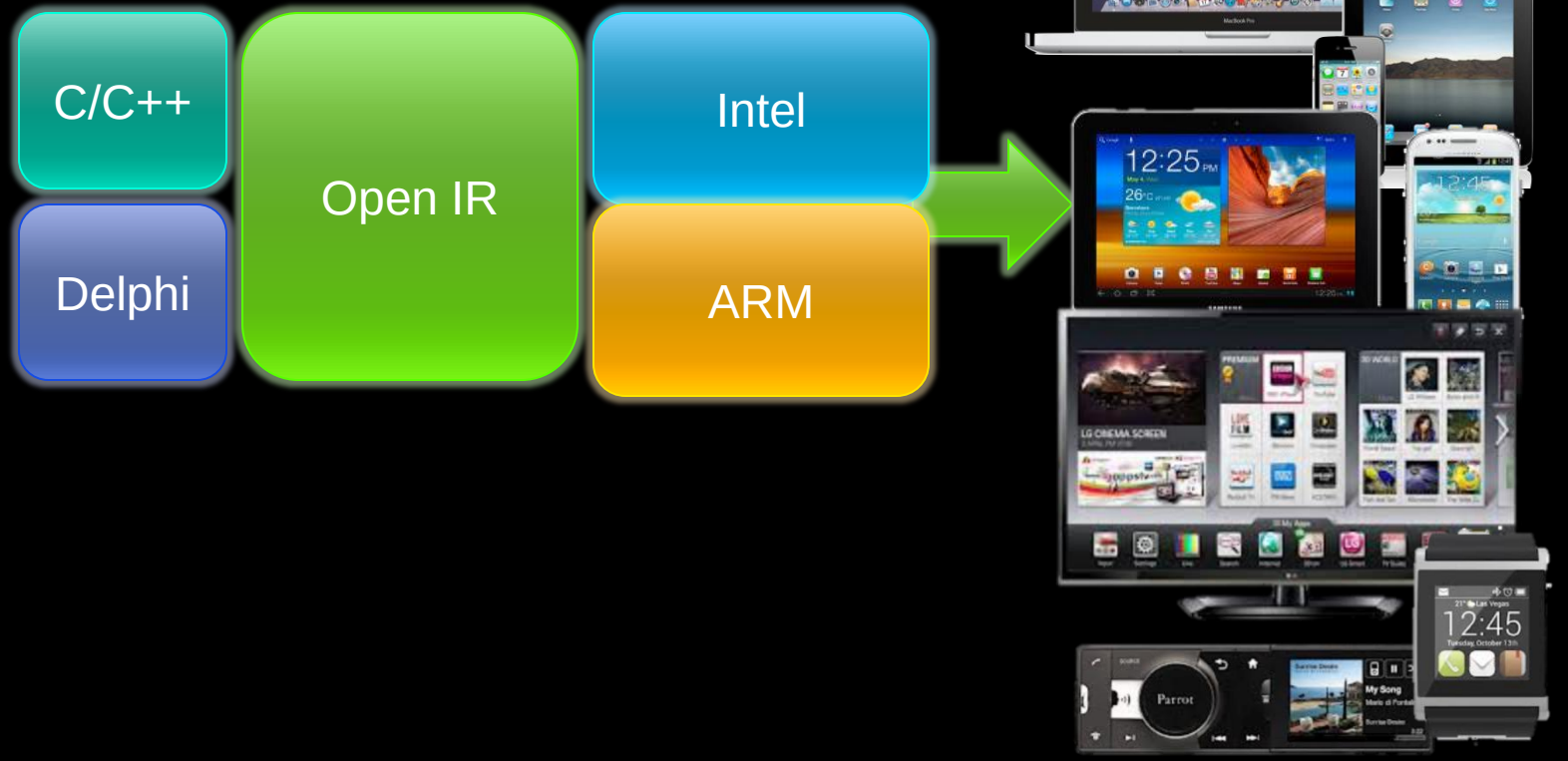
| | **Win** | **Mac** | **iOS** |
|---|---|---|---|
| VCL |  | | |
| FM |  |  |  |

# Delphi FireMonkey on iOS

- iOS Native Style
- Familiar Delphi workflow



# New Compiler Architecture



# Delphi Multi-Device Unique Features

- FireMonkey for Desktop and Mobile
  - Same UI controls, some specific ones
- Enhanced Delphi Language
  - Compiled for ARM devices
  - Development of new compiler architecture
  - New modern languages features coming, better for new comers (and for all)
  - Target new Delphi users

# OS Support for iOS Development

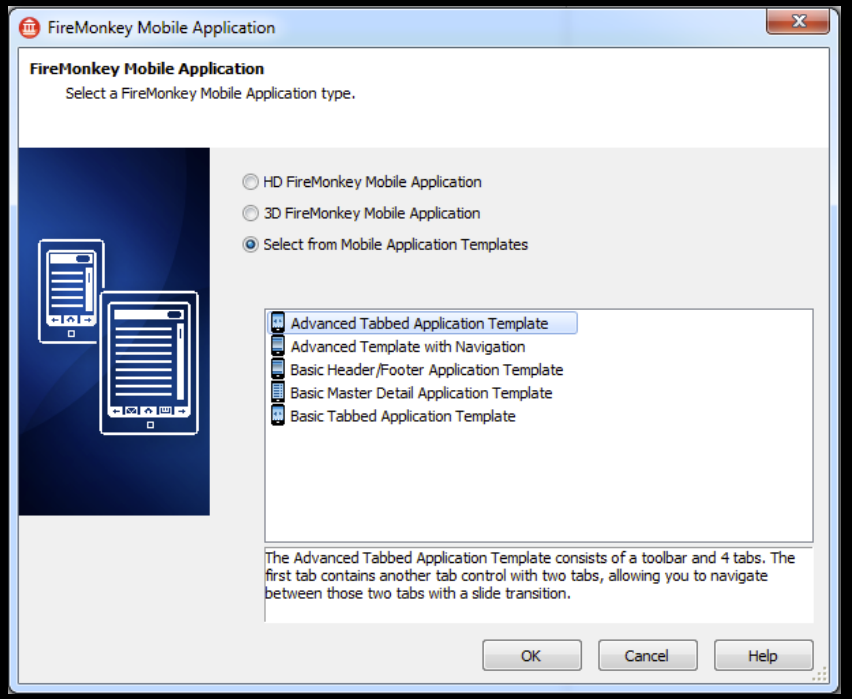
- Mac OS X – “Lion” and “Mountain Lion”
- iOS 5.1 and 6.x

# Delphi Workflow

- Use Delphi or RAD Studio IDE on Windows
- Requires a Mac OS X
  - XCode installed
  - Apple Developer account
  - Provisioned devices
  - PA Server installed
- Very different from Delphi XE2 workflow
  - XCode compilation + FP + XCode debugging

# Built-in Mobile Application Wizard

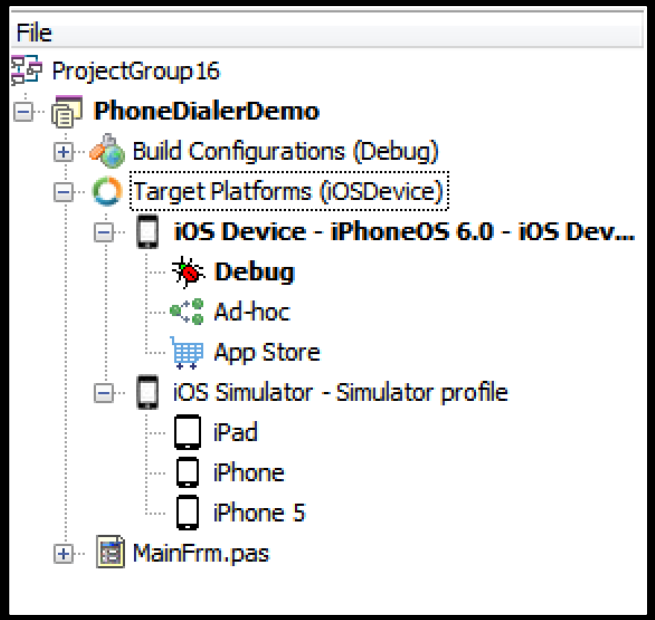
- Start with a blank HD or 3D FireMonkey application
- Choose from Tabbed Application, Header/Footer and Master/Detail Templates

# IDE deployment options for iOS

Deploy iOS apps to the Simulator

- iPhone (Retina/non-Retina)
- iPhone 5 (Retina/non-Retina)
- iPad (Retina/non-Retina)

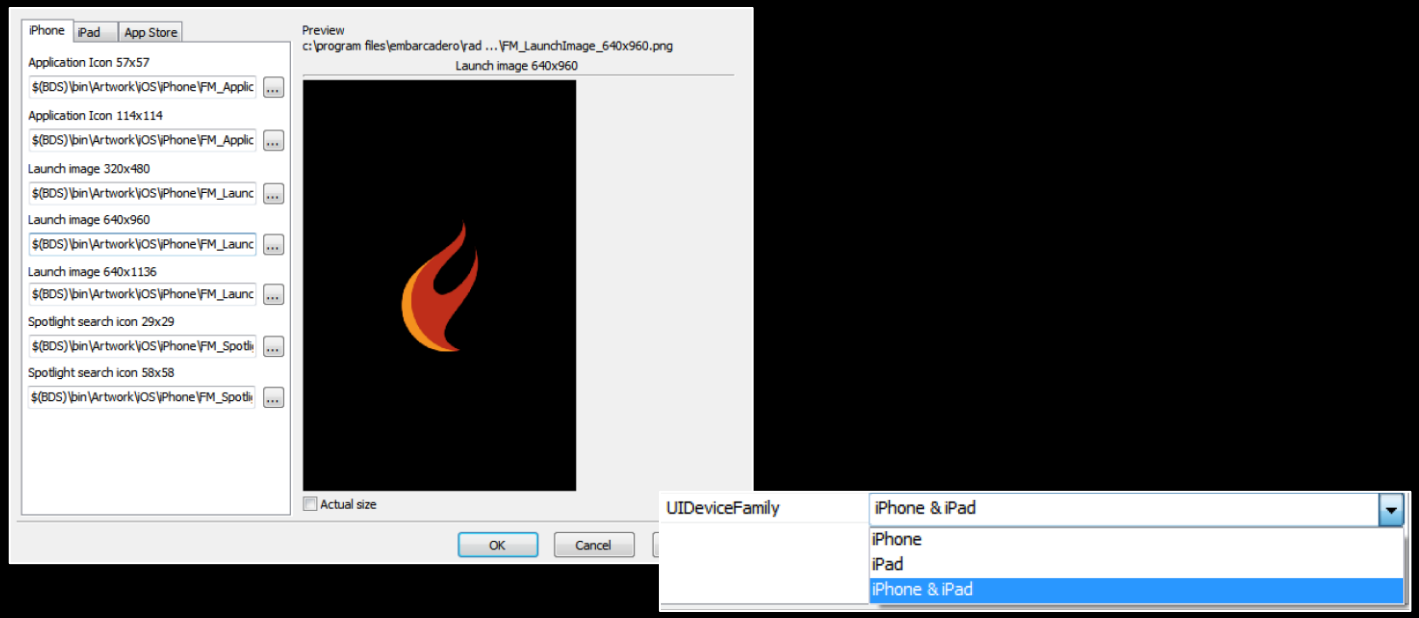



Deploy iOS apps to the Device

- Debug (debug/deploy to device)
- Ad-hoc (distribute within own enterprise)
- App Store (deploy to the App Store)

# Defining Application Settings

- Define device support in Project->Options
- Select app icons for iPhone/iPad/App Store in Project-> Options

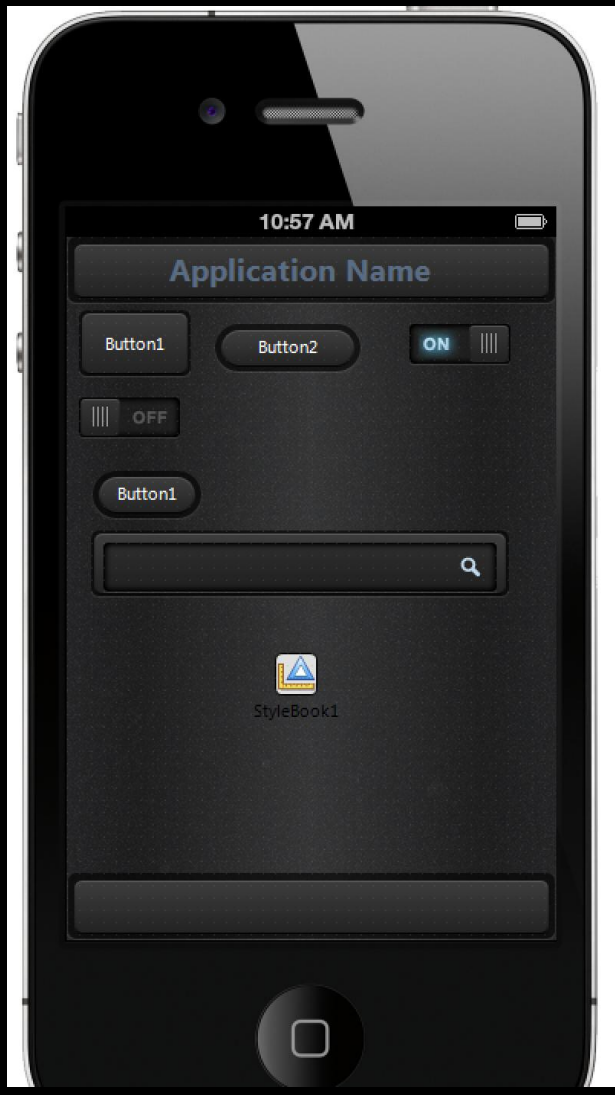


# Delphi for iOS Demo

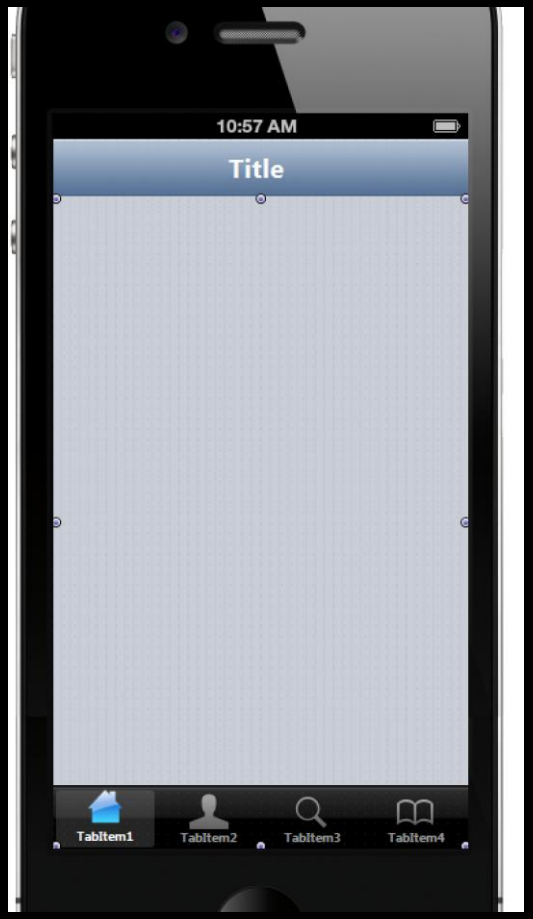
- Your first Delphi for iOS "Hello World!"
  - To see how easy it is to develop iOS App with Delphi for iOS
  - To demonstrate the whole development process

# Native and Custom Styling

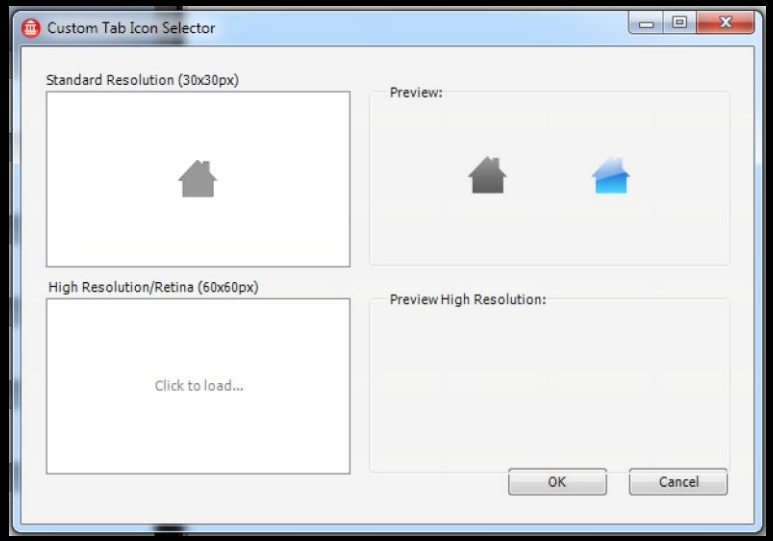


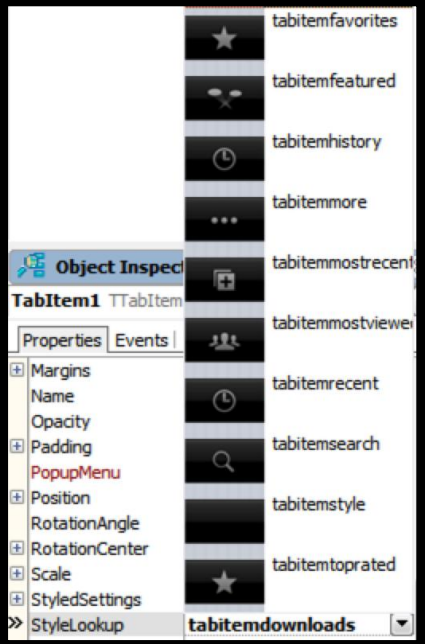


# Native and Custom Styling




- Support for custom tab icons
- Support for Retina images
- "StyleLookup" property



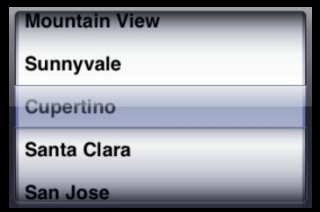





# Native iOS controls and support

- Message alerts
- Custom Picker
- Date Picker
- Phone Dialer Support
- iOS Keyboards
- Text Editing for TMemo and Tedit
  - Cut/Copy/Paste/Zoom





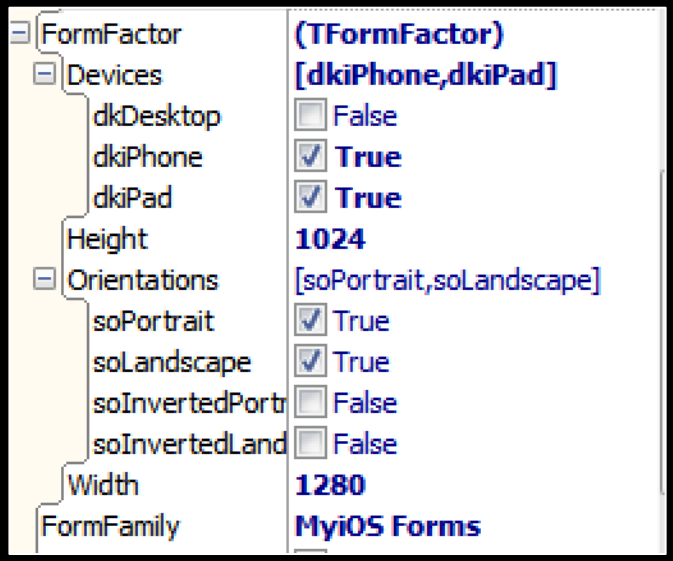


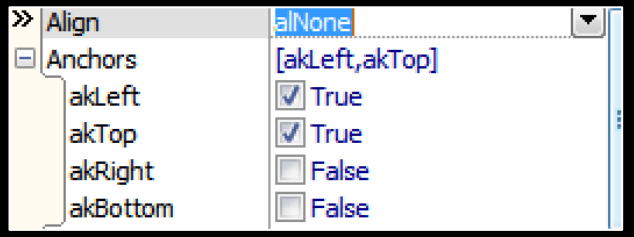
# Delphi for iOS Demo

- Delphi for iOS supports iOS native controls
  - Custom Picker and Date Picker demos
  - Phone Dialer Demo
  - Text and Keyboards Demo

# Layout Management

- Alignment
- Anchors
- Form Family for loading the correct form depending on the target device when developing different forms for iPhone vs iPad/Landscape vs Portrait



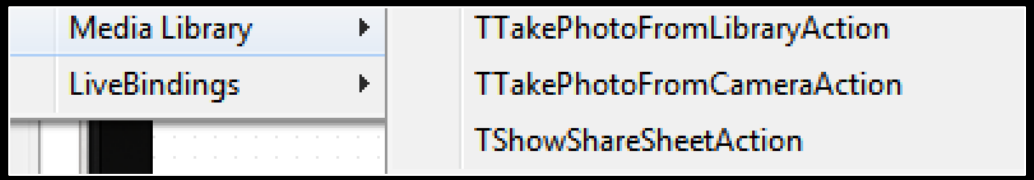



# Gestures

- Swipe
- Tap
- Pinch & Zoom
- Tap & Hold
- Double-Tap

# Extended Action Support


- Accessing the Camera App
- Accessing/retrieve from the Camera Roll
- Share Sheet functionality (share content i.e. photos via Message (SMS), Mail, Facebook, Twitter, print via AirPrint etc.)
- Slide Transitions for Tab Items in TabControl



# Sensors

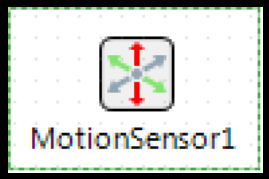
# Orientation Sensor (Gyroscope/Compass)

- Get X,Y,Z tilt values
- Get X, Y, Z distance values

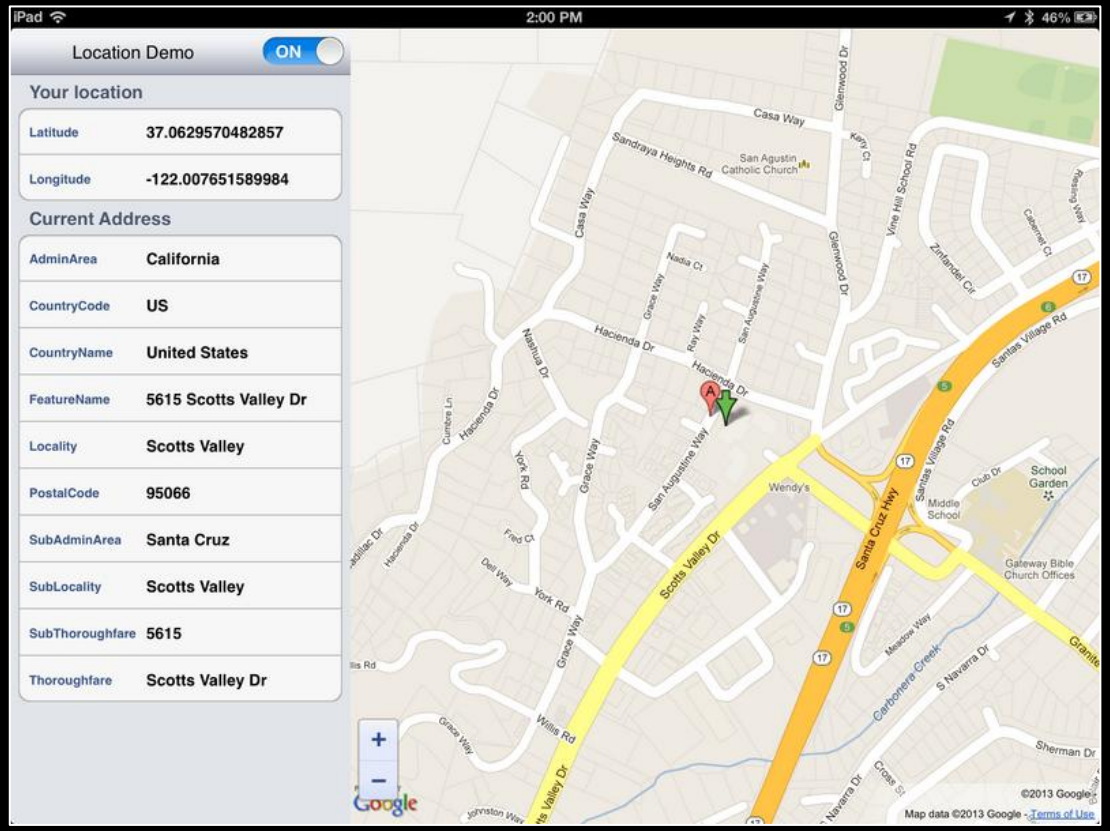

# Motion Sensor (Accelerometer)

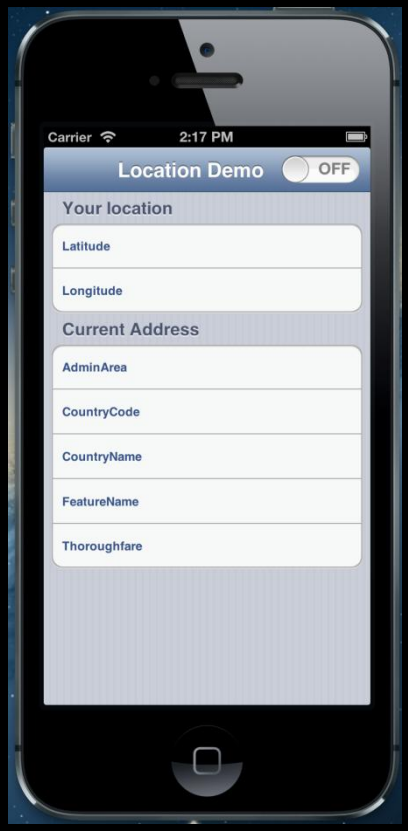
Used to detect motion in your application as you move your iOS device

- Get Acceleration Values and Angle Acceleration Values (X, Y, Z)
- Determine Speed
- Determine Motion



# Location Sensor



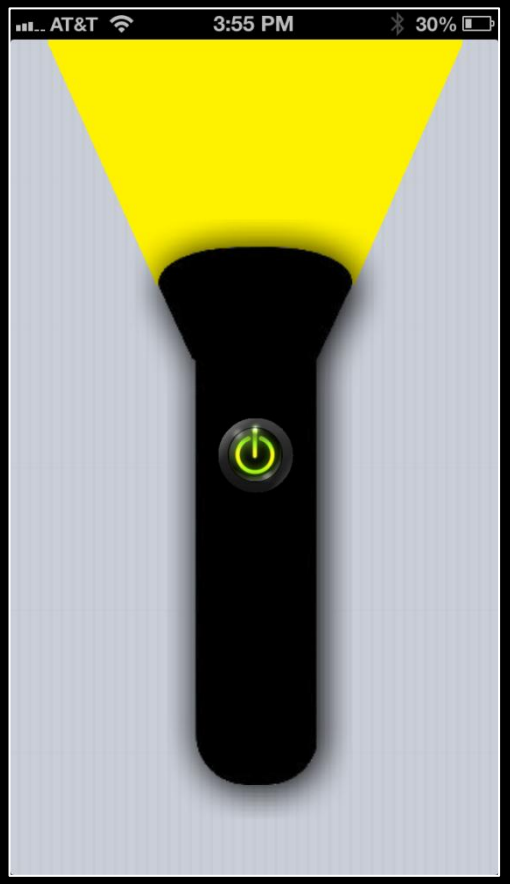

Commonly used in applications that require location awareness

- Get location of your iOS device using latitude and longitude
- Use Reverse Geocoding to convert location data to a readable address
- Works across Win/Mac/iOS
- Can be used with the WebBrowser component to display a location on the map

# Camera

Provides access to camera sensors:

- activate flash
- get sensor position etc.
- access front/back camera



# iOS Services

# Notification Center

We support local notifications in XE3.5. The following notifications are supported:

- Sending scheduled notification
- Presenting a local notification immediately
- Canceling of all notifications
- Canceling of specified scheduled local notification
- Setting of badge number for application icon
- Resetting of badge number for application icon



Key Basic Notification or Alert Styles

- Badge on Application Icon
- Notification Banner on iPad
- Notification Alert
- Notification Center on iPad

# Delphi for iOS Demo

- Camera  and Photo Library Demo
- Sensor Demo
- Location Demo

# Delphi for iOS Demo

- More demos
  - TabControl demo
    - Animation
    - Effects
    - WebBrowser
    - Style

# Delphi Language enhancements

# Delphi "NextGen" Compiler

- New Compiler Architecture
  - {$IFDEF NEXTGEN}
- LLVM
  - http://llvm.org/
  - Used also by Apple
  - Multiple back ends (ARM included)
- Two active (classic, nextgen):
  - Win32 / Win64 / Mac 32 / iOS emulator = classic
  - iOS / Android when available = nextgenDelphi compilers

# NextGen Compiler Changes

- Rationale
  - Optimize for mobile (memory and more)
  - Simplify language for new developers (benefits all)
  - Clean-up language quirks
- Practical effects
  - String types cleanup and changes
  - Reference counting for objects (ARC)
  - More (to avoid): with, pointers, static arrays

# String Type

- One string type, like UnicodeString
  - No native one-byte string type
- Implemented as reference-counted, immutable strings
- By default:
  - {$ZEROBASEDSTRINGS ON}
  - {$WARN IMMUTABLE_STRINGS OFF}
- Key role of string type helper (TStringHelper)

# String Type Compatibility

- Use TBytes for one-byte strings
  - Possibly new features to make this easier
- For immutable, use TStringBuilder in loops
- For zero-based
  - Use new XE3 string helper (0-based)
  - Use traditional RTL functions (1-based)
  - For loops from “Low” to “High”
  - …change compiler directive

# ARC

- ARC (Automatic Reference Counting)
  - http://clang.llvm.org/docs/AutomaticReferenceCounting.html
- All objects are referenced, destroyed when there are no standing references
- Similar to interface reference counting
  - Weak reference concept (weak attribute)
  - Const parameters and *unsafe* results

# ARC & Compatibility

- Classic try-finally blocks and free child in destructor still work on nextgen
  - Although they are useless / not needed
  - Free = set to nil
  - New “Dispose” pattern (still coming)
- Attributes are ignored on classic
  - Can have [weak] in all code
- Check with {$IFDEF AUTOREFCOUNT}
  - Although you’d want to minimize them

# Other Language Changes

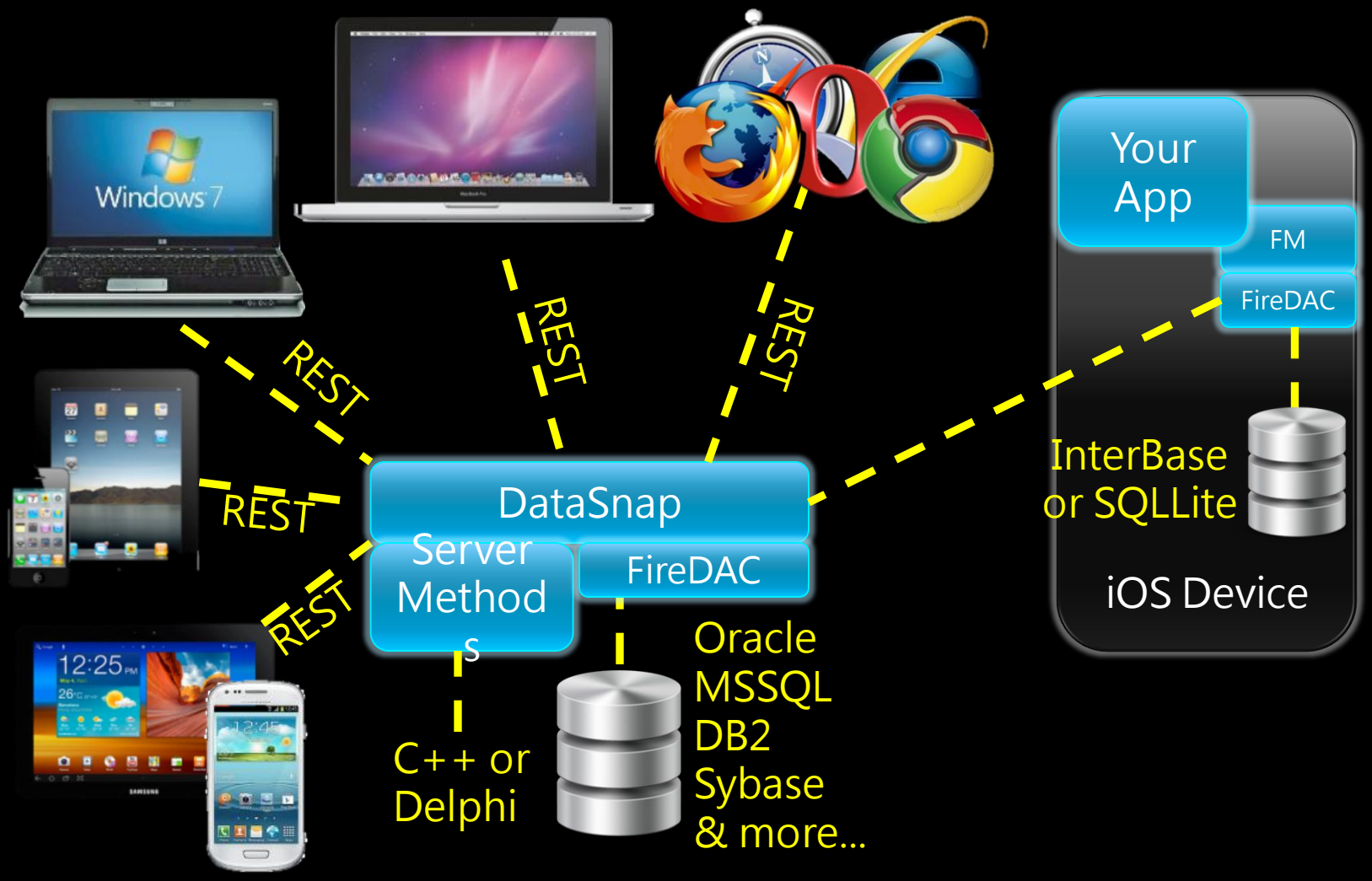
- Pointers
  - Remove all pointers, if you can
  - Avoid pointers to objects (because of ARC)
  - Use generic lists, rather than plain TList
- Arrays
  - Replace static arrays with dynamic arrays
- With
  - Going, going… still not gone!

# COFFEE BREAK

# GETTING THE MOST FROM MOBILE

SMART DATA ACCESS FOR YOUR SMART DEVICES - VISUAL LIVEBINDINGS

# Enterprise Ready



# Visual LiveBindings

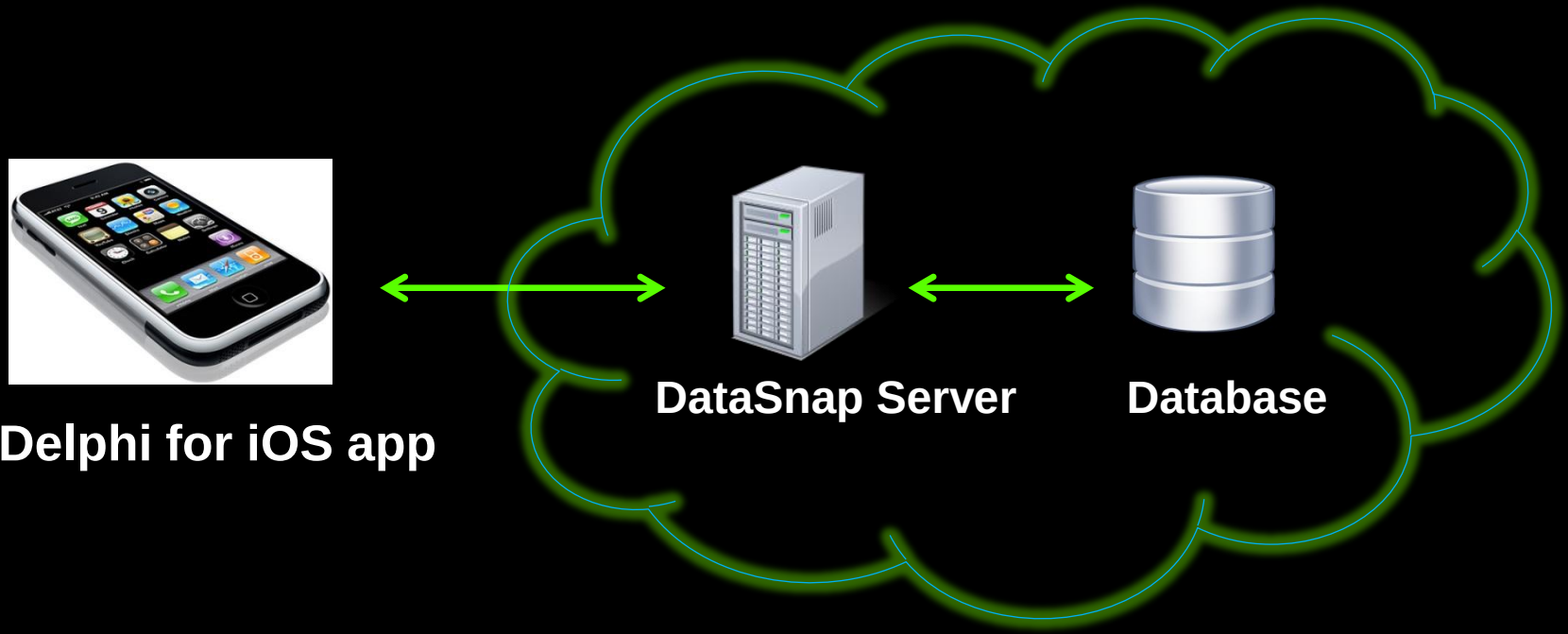

- Bind controls to data
- Rapid Prototyping

# Local Databases

| SQLite | InterBase ToGo |
|---|---|
| Free | Commercial |
| Feature light | Fully featured |
| No security | Secure Encryption |
| Simple Data Storage | Full SQL-92 RDBMS |
| Single read/write | Fast multi read/write |


# Multitier Development

- Accessing remote services
- Connecting to DataSnap servers from an iOS device



# Delphi for iOS Demo

- RESTful + JSON Demo
- XML Demo
- Delphi Feeds

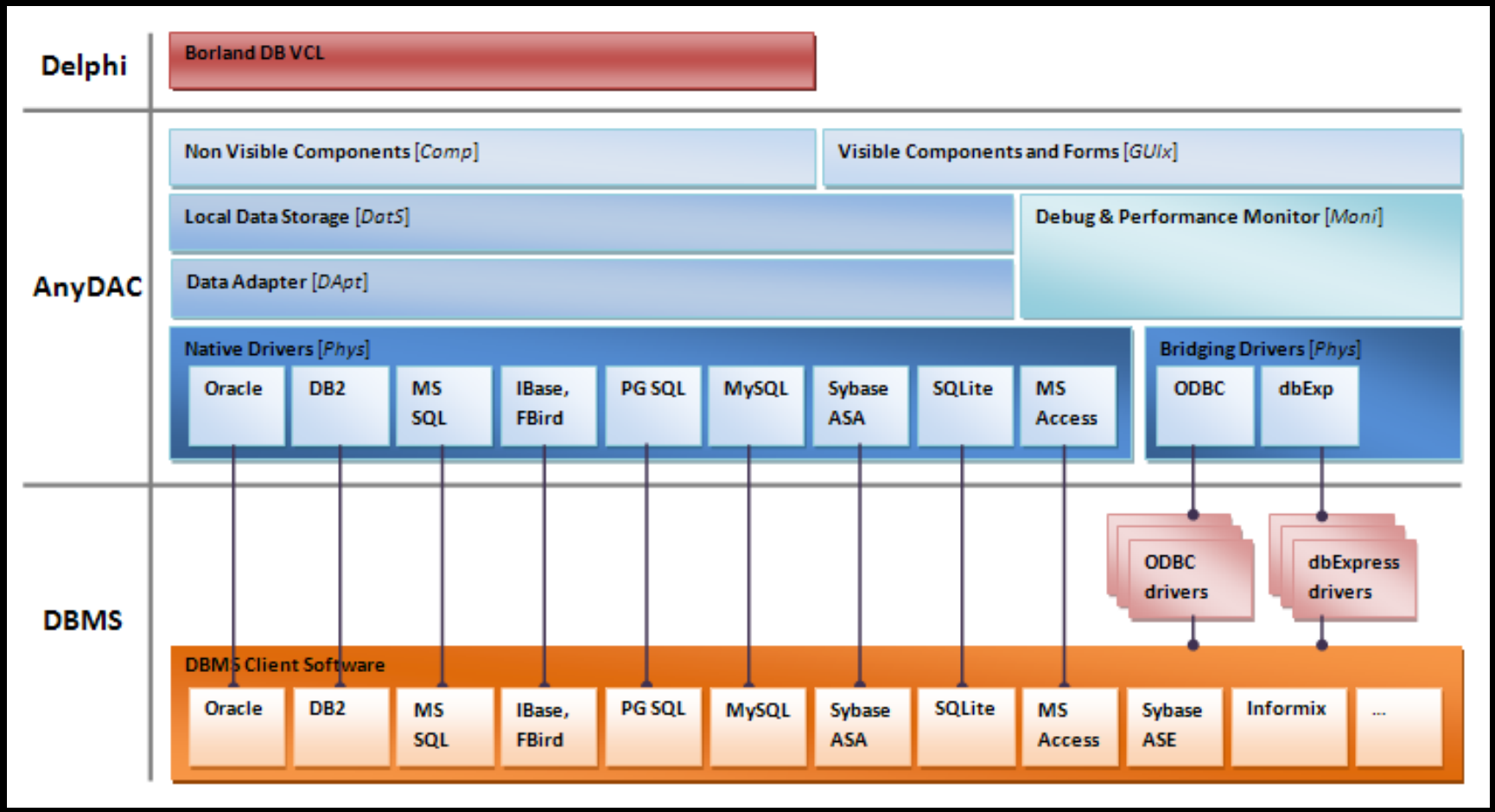
# INTRODUCING FIREDAC


# Key FireDAC Features

- Data Access Engine
  - Foundation classes + TDataSet comps
- High Performance Data Access
  - From Live Data Window to Array DML
- Unified API
  - SQL abstraction and scripting
  - Unified errors and transations

# Native FireDAC Drivers

- MySQL
- Microsoft SQL Server
- Oracle Database
- InterBase
- PostgreSQL
- DataSnap
- SQLite
- Sybase SQL Anywhere
- Microsoft Access
- IBM DB2 Server
- Firebird
- Advantage Database
- ODBC gateway
- dbExpress gateway

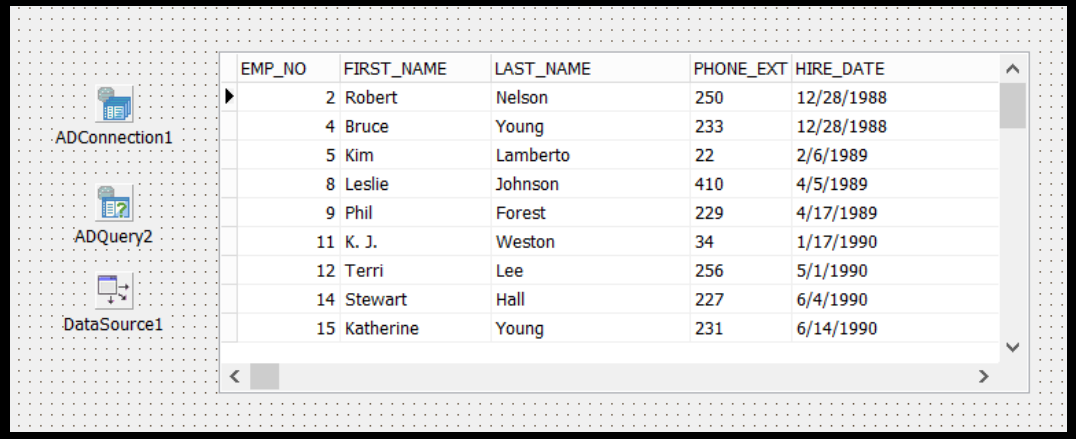

Delphi
Borland DB VCL
AnyDAC
Non Visible Components [Comp]
Visible Components and Forms [GUIx]
Local Data Storage [DotS]
Debug & Performance Monitor [Moni]
Data Adapter [DApt]
Native Drivers [Phys]
Oracle
DB2
MS SQL
IBase, FBird
PG SQL
MySQL
Sybase ASA
SQLite
MS Access
Bridging Drivers [Phys]
ODBC
dbExp
ODBC drivers
dbExpress drivers
DBMS
DBMS Client Software
Oracle
DB2
MS SQL
IBase, FBird
PG SQL
MySQL
Sybase ASA
SQLite
MS Access
Sybase ASE
Informix
...

# FireDAC Core Components

- TADConnection: connection to a database
- TADTransaction: transaction in a connection
- TADMemTable: in-memory dataset
- TADQuery: executes a SQL commands and returns result sets
- TADStoredProc: executes a stored procedure
- TADTable: open table data for navigation
- TADScript: executing SQL scripts

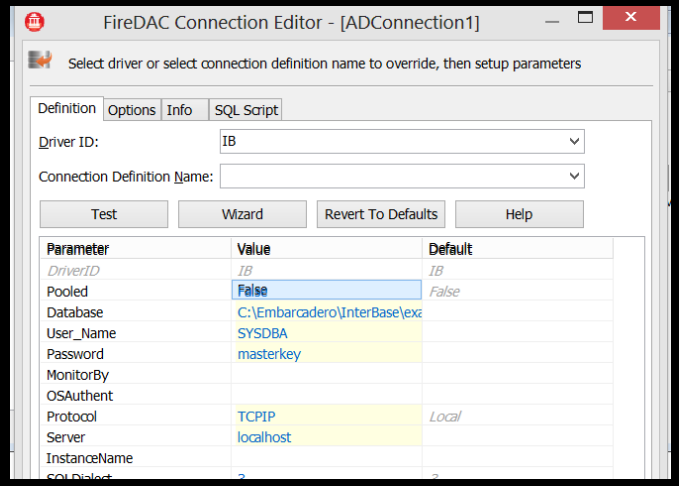
# FireDAC 101 Demo

- Simplest demo
  - Connection
  - Query
  - DS + Grid
- Fully working
  - In-memory dataset built in (local sorting OnTitleClick)
  - Cached updates optional (use Apply if On)
  - Requires ADPhysIBDriverLink



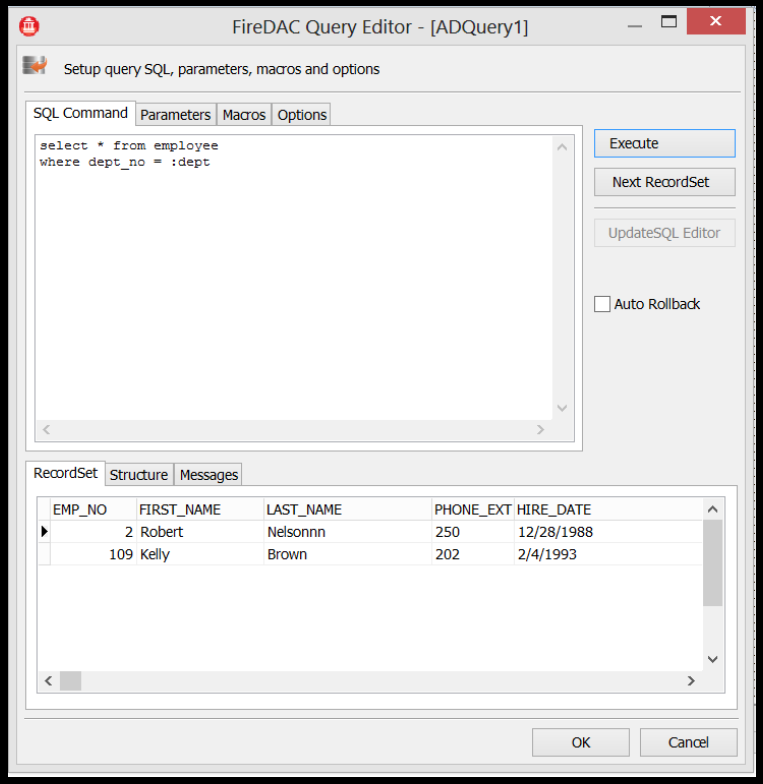
# Configurations and Connections

- ADExplorer
  - Stand alone configuration editor
  - AdConnectionDefs.ini
  - FireDAC Connection
- Editor (IDE)
  - ADConnection component configuration
  - Rich set of options, information, settings
- Connections can be in code



# Query Editor

- Manages SQL query, parameters, options
- Integrated preview



# FireDAC Tracing

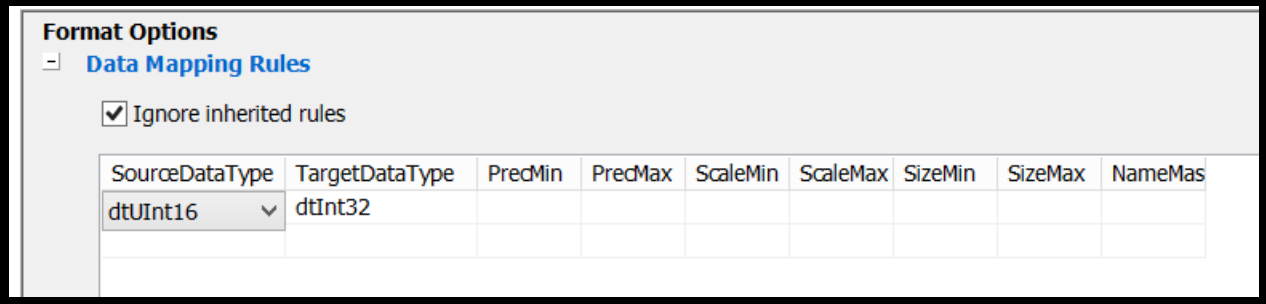
- TADMoniFlatFileClientLink: Text file
- TADMoniRemoteClientLink: ADMonitor
- Enable with
  - Tracing to True
  - MonitorBy=FlatFile or Remote in connection settings
- (See DacIntro demo)

# FireDAC Options System

- About 60 options
- FetchOptions – rows fetching
- FormatOptions – data type handling
- ResourceOptions – resource usage and more
- UpdateOptions – how FireDAC will post updates
- Option values are inheritable
  - From connection to datasets (shareable datasets)

# Data Type Mapping

- Custom mapping by connection or dataset visually




- Or in code

# Rowset Fetching

- Number of records per network round trip
- The slower the network, the more it help
- Controlled by FetchOptions.RowsetSize
- 100K records
  - RowsetSize = 1 -> 7,5 sec
  - RowsetSize = 100 -> 0.65 sec

# ADMemTable

- ADQuery has caching support
- Use ADMemTable to keep data snapshot in memory, load multiple
- Can also use ClientDataset and Provider, for compatibilty

- Demo FireDACMemTable

# FireDAC Array DML

- Execute N INSERT / UPDATE / DELETE parameterized commands as a single unit
- Universal, simple, effective and convenient
- Each parameter stores array of values
- Great on slow networks, weak servers
- INSERT of 10K of records:
  - Array DML -> 0,03 sec
  - Normal ExecSQL -> 5.5 sec

# FireDAC Async Execution

- Long running operations may run asynchronously or with time restriction
- ResourceOptions.CmdExecMode: operation execution modes
- ResourceOptions.CmdExecTimeout: operation execution timeout
- TADGUIxFormAsyncDlg, CmdExecMode = amCancelDialog – end user solution
- AbortJob – cancels an operation

# FireDAC SQL Processing

- Escape functions: single expression for any DB
- Conditional escapes: write SQL parts differently for specific DB's:
  - {IF Oracle} SELECT * FROM "Region" {fi}
- Macros: substitute variables extending parameters usage
- Full scripting support

# FireDAC Automatic Editing

- Generator is aware of DBMS SQL dialects and features
- No need to specify update SQL commands, generated automatically for the target DBMS
- TADUpdateSQL is available but optional

# FireDAC AutoInc Handling

- Recognizes IDENTITY and similar columns
- Recognizes columns with BEFORE INSERT trigger filling the column from a generator
- Automatically refreshes posted new records
- Works for immediate updates and for cached updates

# Wait, There is More!

- TADTable with the Live Data Window mode
- Powerful in-memory datasets with sorting, filtering, locating, aggregates
- Connection online and offline modes and automatic connection recovery
- Multiple transactions support
- Error reporting and database events support
- Database backup, restore, validate, repair support
- Connection pooling

# FireDAC Availability

- Included in Enterprise Editions
  - Also for existing XE3 customers
- FireDAC C/S Pack add-on for XE3 Professional Edition customers
  - Maintenance is included
  - Maintenance on base product is highly recommended

# FireDAC Summary

- A set of Universal Data Access Components
- High-performance, easy-to-use, enterprise connectivity
- Universal Data Access with many database specific features

# Delphi for iOS Demo

- Database demos
  - TClientDataSet Demo : SimpleCDSApp2
  - Livebinding Demo

Q & A

SUMMARY

# Questions?