Session 1

Windows 8 & Metro UI:
The Changing Application Paradigm
What we’ll cover

Changing landscape
– a brief history of applications

What is Metro?

What do Architects and Developers need to change to use Metro?
Changing Landscape

Client Server

Monolithic

Distributed, Web Based, Mobile
Introducing Metro
Windows Runtime – the underlying platform

Metro style apps

- XAML
- HTML / CSS
- JavaScript (Chakra)

Desktop apps

- HTML
- C
- C#

- C++
- VB
- JavaScript

WinRT APIs

- Communication & Data
- Graphics & Media
- Devices & Printing
- Application Model

System Services

- Internet Explorer
- Win32
- .NET / SL

Model Controller

- View

Core

Windows Core OS Services
# A Financial World Scenario

<table>
<thead>
<tr>
<th>Actors</th>
<th>Applications</th>
<th>Functions</th>
</tr>
</thead>
</table>
| **Portfolio Manager** | ![Portfolio Management App](#) | - Front-faces Investors  
                   |                                      | - Creates Strategies & Deals  
                   |                                      | - Watches Stock Prices               |
| **Execution Trader** | ![Order Manager Blotter](#) | - Acknowledges New Orders  
                   |                                      | - Approves Orders  
                   |                                      | - Analyzes Daily Trading Data         
                   |                                      | - Watches Stock Prices                |
| **Analyst**     | ![Analytics Tool](#)   | - Go To Person For All Research & Analysis  
                   |                                      | - Watches Stock Prices               |
Monolithic Rich Apps
Less is More

- Create smaller apps focused on specific things they are best at
  - Share charm used to share functionality
Contracts and Reuse

• Support the Metro search charm by making your application searchable

• Enables mash-up scenarios not thought of at the time of application development

• Activation contracts enable Applications to invoke one another
Think standards

• Detailed UI guidelines and standards around:

  **Navigation – [Hub, Flat]**

  ![Navigation Example](image)

  **Touch, mouse, and keyboard**

  ![Keyboard Example](image)

  **Commanding**

  ![Commanding Example](image)

• Key component of enabling support across multiple devices and form factors
Costly to support Multiple Devices
Multiple devices

- Not all parts of a large app make sense on all devices. Smaller focused apps enable easier multi-device support
- Standards enable support for multiple form factors
- Application lifecycle management changes to support devices with limited battery life
Key changes

1. Less is more!
2. Create small apps that are best at doing just one thing
3. Think of contracts that your application needs to expose
4. Think of the activation points of your application
5. Reuse existing applications to create mash ups
6. Much more discipline around standards!
7. Bring together use cases that make sense on multiple form factor devices
8. Application lifecycle
Sharing Perspectives

- Q&A session