





You will need:



You will need:

The Java language



You will need:

- The Java language
- The Java Virtual Machine (VM)



You will need:

- The Java language
- The Java Virtual Machine (VM)
- The Java libraries



Choices, Choices



Sun's Abstract Windowing Toolkit (AWT)



- Sun's Abstract Windowing Toolkit (AWT)
- Netscape's Internet Foundation Classes (IFC)



- Sun's Abstract Windowing Toolkit (AWT)
- Netscape's Internet Foundation Classes (IFC)
- Microsoft's Application Foundation Classes (AFC)



- Sun's Abstract Windowing Toolkit (AWT)
- Netscape's Internet Foundation Classes (IFC)
- Microsoft's Application Foundation Classes (AFC)
- Sun's Java Foundation Classes (JFC)



Pure Java Libraries Are Cross-Platform



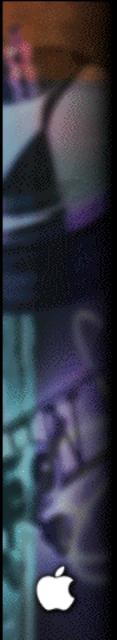
Pure Java Libraries Are Cross-Platform

- The JDK provides the platform
 - Graphics
 - Events
 - Windows



Pure Java Libraries Are Cross-Platform

- The JDK provides the platform
 - Graphics
 - Events
 - Windows
- Java libraries built on top of JDK





Complete JDK implementation on Mac OS and Rhapsody



- Complete JDK implementation on Mac OS and Rhapsody
- All pure Java frameworks run on the Mac



- Complete JDK implementation on Mac OS and Rhapsody
- All pure Java frameworks run on the Mac
- JFC, AFC, or something else?



- Complete JDK implementation on Mac OS and Rhapsody
- All pure Java frameworks run on the Mac
- JFC, AFC, or something else?
 - You make the call...





Agenda

- AFC
 - Really Java?
 - AWT, how does it fit in?
- UI Controls
 - Accessibility, internationalization
- FxClasses
 - Greater Power



What Is AFC?

Comprehensive suite of UI controls, graphics, and effects classes

- Written completely in Java
- Builds on knowledge of AWT
- Intuitive, component-based structure
- Automatic accessibility and internationalization



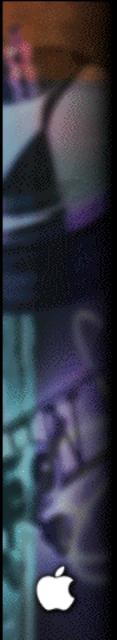
Written in Java?

- You betcha!
- No native code
- JDK 1.1 Compatible today
- JDK 1.1 and 1.0.2 compatibility for release
- Can run on any Java VM, any Java-enabled platform



Builds on AWT?

- No new model to learn
- Leverage existing AWT experience



AFC Design Goals

- Cross platform UI elements with common look and feel
- Support additional "native" look and feels within Java
- Provide the ability to extend and customize elements!



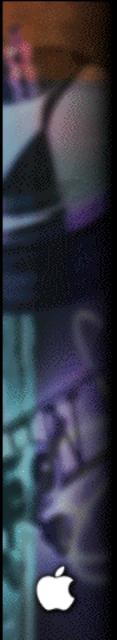
UI Controls

- UIStatic—text control
- UIStatus—a progress control
- UIButton
 - UIPushbutton
 - UIRepeatButton
 - UIMenuButton
 - UICheckButton
 - UIRadioButton



UI Controls (cont.)

- UIThumb—dragger control
- UIFormattedText—rich text control in Java
- UIScroll
 - UIScrollBar
 - UISlider
 - UISpinner
- UIViewer—container control
- UISelector—selection from list control
- UIGroup
- UIPopup



FxClasses

- As UI Controls are to AWT, so FxClasses are to AWT Graphics
 - Richer primitives
 - Fully extensible and customizable
- Plugs straight into existing AWT applications



FxGraphics

- Basis of Fx extensibility
- AWT Graphics—basic primitives only (Oval, arc, line, rectangle)
- FxGraphics—richer set of primitives
 - Basic primitives fully extensible (dotted lines, for example)
 - Scanlines
 - Drawing formatted and tabbed opaque text
 - Beziers



FxColor

- Extends AWT Color to support
 - Textures
 - FxPen
 - FxRubberPen
 - FxBrushPen
- Improved texture support
 - Tile
 - Texture lines
 - Set background or texture of any UI or AWT component





JFC and AWT

- History
- Rational
- The details...



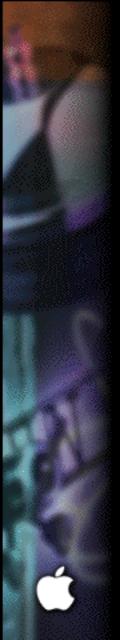
History

- Was: GUIs for dynamic web pages
- Is: Comprehensive foundation for commercial applications



Rational

- Write Once...
 - Core to the JDK
 - Common Platform
- Choice and Flexibility
 - Open component and application design
 - Easy subclassing
 - CHUI, FUI, etc.



JFC

- Built as an extension to the core AWT
- Provides services and components to the serious application developer
- Provides the security yet the flexibility needed



Topics

- Lightweight UI Framework
- "Pluggable" Look and Feel
- New Components!
- Drag and Drop
- 2D Graphics API
- Delivery Plans



Lightweight UI Framework

What is it?

- Benefits
 - Transparency
 - Light (no native code)
 - 100% Pure Java™ platform-independent
- Future: java.awt.Component
- Will get lighter



"Pluggable" Look and Feel

Wby?

- JavaSoft's "Write Once, Run Anywhere"™ promise
- There are lots of different users
- There are lots of different devices
- Enable UI designers and artists



"Pluggable" Look and Feel

What is it?

- Separates component's UI from its state
- UI factory provides component's UI at runtime
- Different from an AWT peer



"Pluggable" Look and Feel

- Customizable
 - Replace entire UI
- "Classic" and "Corporate" UIs
- Accessible UIs
- Consumer UIs
 - Modify UI building blocks
 - Control UI of individual components



New Components!

- Based on lightweight, peerless architecture
- Built from smaller common classes
 - Composition in favor of inheritance
 - Interfaces in favor of subclasses
- 100% Pure Java, all common code



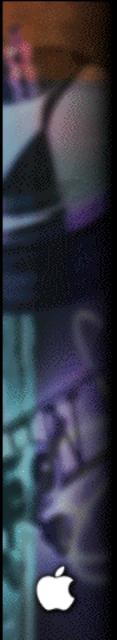
New Components!

- Higher-level components
 - TreeView, TableView, ListView
 - TabbedFolder, PaneSplitter, ToolBar
 - ColorChooser, FontChooser
 - Styled Text
- Lower-level Components
 - Icon, Tooltip, StatusBar, MessageBox
 - Slider, Gauge, Spinner



Drag and Drop

- Top priority for the next major release of JDK™
- Draft API specification on java.sun.com
- Java-to-Java implementation in early access release
- Custom cursors



Java 2D API

- Based on close collaborations with Adobe
- Supports PostScript-like rendering capabilities
- Transformations, paths, fills, and lots more...
- Plus antialiasing and alpha blending
- See JavaOne presentation on java.sun.com



Java 2D and Components

- Non-rectangular components
- Resolution independence
- Supports transformations
- Finer text rendering
- More fonts!



Delivery Plans

- Early licensees in development (Apple, Netscape, etc.)
- Developer release early June
- Product release as part of the next major JDK (late summer...)



