A11y TODO

OLD TODO (most of these are done)

Events

To Node and Scene
For GetAttribute and ExecuteAction

* Internally use events to connect Node#accGetAttribute and Node#accExecuteAction
* Allows run before and after default handler
* Add default handlers in the constructor to ensure event order

alternateText

Add it to Canvas, ImageView, and Labeled (Label, Button, Cells, etc)
As a rule: any place that answers TITLE should offer alternateText.

- Pro: easy to provide a more descriptive text than what is in textProperty.
- Con: not sure that is common case
- Con: that can be done using the event
- Con: can be added later

Priority: alternateText() has the highest priority of all (beats, text and promtText).

Note regarding LABELED_BY: The string obtained from TITLE and the string obtained LABELED_BY should co-exist, both are read out. Thus there is no priority order between the two.

Note: adding a alternativeText() for Labeled is not bad, by maybe unnecessary.

labeledBy vs labelFor

The accessibility attributed is called LABELED_BY, the property in Label is called labelFor.
currently LABELED_BY is implemented using Label#labelFor in Node using accessor pattern.

Adding a labeledBy in Node:

- Easy to set any Node as a label to any node
- Confuses/Conflicts the with Label#labelFor

Not adding a labeledBy in Node:

- Only Label can easily be used as label
- Use the event to any node as label

Other Option: Move Label#labelFor up to Node
*Current resolution is to take no action. *

Virtualization

Pages in the pagination. -> Probably Yes
Tabs in the tab pane. -> Probably yes
Children in the parent. -> Probably not, to the best of my investigation, Mac for example, insists on getting the whole NSArray in AXChildren.

No conclusion reached.

getAccessible() & Accessible

Probably remove the method from Node and Accessible class from the API.

yes to remove getAccessible() and Accessible()

Method names

node#accGetAttribute
node#accExecuteAction
node#accSendNotification

The first two are the default event handlers for GetAttribute and ExecuteAction respectively.

Subclasses should always override the method (instead of installing their event handler), this helps to keep the order handlers are called predicable.

Better names ?

node#accGetAttribute ->
* node#accessibilityGetAttribute() - Too long?
* node#accessibleGetAttribute()
* node#getAttribute() - break the rule (such as user data)
* node#queryAttribute()
* node#valueForAttribute()
* node#namedAttribute()

* Current plan is to use getAttribute(), executeAction(), sendNotification() *
Yes: breaking the rule for getAttribute()

Use Jira Search

OLD TODO (in process to port this to Jira)

- NO - View.getAccessible() should follow the notify*() pattern to reflect that it's a native callback. Anthony Petrov
- YES - Classes in win/, mac/, etc. should start with a Win*, Mac*, etc. prefixes (as much as I don't like the naming scheme, we should stick to the pattern to avoid confusion).
- NO -   Use javaIDs instead of statics for MIDs in GlassAccessible.cpp
- YES - Perhaps instead of using the PlatformFactory, we should add a new factory method to the Application class in Glass.
- Support Windows 7. Currently GlassAccessibility extends ISelectionItemProvider, which is only supported since Windows 8. We may want to eliminate the direct code dependency on Win8 interfaces and use some dynamic techniques to support both Win8 & Win7. Anthony Petrov
  The problem is that after trying to expand the FX window in the UISpy tree, the get_HostRawElementProvider() in GlassAccessible gets a NULL hwnd from Java code, then it passes it to UiaHostProviderFromHwnd(). The latter function reports an error E_INVALIDARG. Windows 8 just eats it and proceeds, but Windows 7 can't do so for some reason. I suppose we should make sure the HWND is never NULL (it shouldn't be, really). Investigating further...
  Fixed by returning S_OK from get_HostRawElementProvider().
- YES - Fix lines indentations
- RadioGroup. AccessibleProvider.
- (Factory, Variant prefix, IAccessibleProvider - DONE), Combo, Popup, Menu - Anthony
- Avoid Class Cast exceptions - right now PlatformAccessible expects the type from getAttribute() to match, need to add checks and fail gracefully.
- Check for Leaks. Make sure the native peers are being disposed (i.e, when the respective node is removed from the scene)
- On Off switch. Add command line option to disable a11y, initially default to off.
- A11y support enums and classes should probably be moved to javafx.scene.accessibility package rather than be in the scene package itself.
- Parfait - have at least one parfait run and make sure the new code does not add parfait problems to JFX
- Findbugs - have at least one findbugs run and make sure the new code does not add findbug problems to JFX
- Public API, better names for all the accGet, accSend, accExecute, etc. Overall API polishness.
- Finish more controls: See Controls Accessibility
- Fix lists/tables on Mac OS X 10.8 (they work on 10.9 but not on 10.8)
- Get rid of debug prints to stdout and error (replace will log message or use a debug flag)
- NULL checks in native code to avoid memory corruption
- Rename STRING_VALUE to ???
- Node.labeledBy - use an accessor
- Hide getAccessible() and the Accessible class from public API (IGNORE, see next item)
- Why do we even need the Accessible class? It only delegates to PlatformAccessible. In Node/Scene we always override its methods and actually delegate to their acc*() counter-parts. So why not use the PlatformAccessible directly.
- Async disposal of Accessible, Replace Scene#pulse mechanism with ReferenceQueue (could be done at the same time as the previous item) SEE RT-36304
- Ensure calls to accSendNotification() are in fact be called in all cases. For exemple, ListView#focus(int) sends listView.accSendNotification (Attribute.SELECTED_ROWS), is that reliable ? (For example, if the focus model was replaced in the ListView, would it break the notification).
- PlatformAccessible#getNativeAccessible() ideally should be private for package visible (not protected).

To ensure we're compatible with Win7, we should use the Windows Desktop documentation in MSDN and check if interfaces that we use are supported on Win7.