KJSEmbed and QtScript

What is application scripting?
What is KJSEmbed?
What is QtScript?
How do KJSEmbed and QtScript compare?
What would we need to do to make QtScript usable for KDE?
Where do we go from here?

What is Application Scripting?

- Parts of an application can be written in higher-level languages than C++
- Allow users to extend applications
- Allow users to automate applications
- Not a new idea, there are many applications written like this, both open source and closed source

Examples of Scripted Applications

- Classic example is emacs
  - Almost all of emacs is written in lisp
  - Very extensible
  - Modes for everything from C++ to guitar tablature
  - Users can heavily customise all aspects of the editor
- Dreamweaver
  - Uses Javascript to bind together components
  - User extensions written in Javascript + HTML
  - Mainly scripting the IE HTML component

Examples of Scripted Applications

- Paintshop Pro
  - Mainly written in C++
  - Allows users to automate it using Python
  - Possibility for user extensions written as scripts

What is KJSEmbed?

- A framework for embedding KJS into applications
- Support for both KDE and Qt applications on both UNIX and win32
- A high level API for creating interpreters
- Glue so applications can expose QObjects to scripts
- Bindings to create QWidgets and load UI files from Designer
- Bindings to QDom, QFile etc. to allow IO from scripts

What is KJSEmbed?

Kross contains support for using KJSEmbed
Support for creating custom widgets in JS including event handling

What is QtScript ?

- This may feel a bit familiar after the last two slides...
- A framework for embedding JS into applications
- A JavaScript interpreter
- Support for Qt applications on all platforms
- A high-level API for creating interpreters
- Glue so that applications can easily expose QObjects to scripts
- A plugin based extension mechanism allowing additional facilities to be added

What is QtScript?

- QtScript is only distantly related to the Qt 3 QSA
- Uses standard ECMAScript 262 rather than the ECMAScript 2 draft
- New bytecode based interpreter
- Unlike QSA the facilities are not crippled
- Extending it is *much* easier than QSA

Similarities

Both use templates for binding (rather than void *)
Both query the QMetaObject to provide instant bindings to QObjects
Both let you call slots as if they were Javascript functions
Both let you connect signals to slots
Both let you connect C++ signals to Javascript functions
Both expose QObject properties as Javascript properties

Similarities

The C++ QObjects that application authors must provide are virtually identical for all the facilities mentioned so far

In fact it should be possible to make it transparent to scripts which interpreter is in use!

Differences

- KJSEmbed lets scripts create any widget supported by Designer
- KJSEmbed lets you load .ui files to easily create dialogs from scripts
- QtScript has no built in facility for this, though it's easy to add as an extension
- Kross already supports KJSEmbed but has no QtScript support

Differences

- QtScript has a nice plugin based extension mechanism
- We aimed to add a facility like this to KJSEmbed but it hasn't been implemented
- QtScript is maintained by Troll Tech, KJSEmbed is currently poorly maintained

Differences

- QtScript has some nice extras that KJSEmbed doesn't as Troll Tech were able to extend the QMetaObject
- Q_INVOKABLE marker allows methods to be called from scripts without being slots
- A QScriptable marker interface that classes can inherit
- A qscriptvalue_cast<> template method that makes it very easy to safely extract C++ objects from script objects
- All in all, I think QtScript has a nicer API

What about performance?

- KJSEmbed uses KJS which while very quick to startup, has relatively slow performace
- QtScript is much faster, performance is roughly equivalent to Mozilla's SpiderMonkey and to python
- My own benchmarks have shown it significantly out performs KJS
- That said, performance has not really been a problem with KJSEmbed it's usually just wiring together C++

Making QtScript usable for KDE

- QtScript has most of what we need
- Discussions with Troll Tech during the pre-releases have ensured that the rest can be built on top of it
- I've already got UI file loading and widget support working, and will port it to be a plugin this week
- Because QtScript and KJSEmbed both use the meta object, most KDE widgets will 'just work' as they've already got properties etc. defined
- Kross support is a major lack

Where do we go from here?

- We need to consider if we want to ship 4.0 with both KJSEmbed and QtScript
- Long term having two solutions to the same problem seems wasteful
- Offloading the maintenance to Troll Tech makes a lot of sense
- A big question is which should we use for plasma?
- Hopefully we can decide what to do this week...