Theming GTK3 widgets with CSS

Cosimo Cecchi
<cosimoc@gnome.org>
Desktop Summit 2011
August 7, 2011
Agenda

1. History
2. CSS
3. GTK3 Theming
4. Questions
GTK theming model 10000ft overview

- GTK applications → set of widgets
- Widgets → drawing functions
- Drawing functions → overridden by themes
- GTK theming system → GTK engines
GTK2 Theming - Engines

GTK2 Engines

- implement the gtk_paint_* methods
- custom engine-specific rendering properties
- access the GtkWidget itself → powerful
GTK2 Theming - GtkRC

Custom text format to describe and configure GTK2 options

- style classes
- set widget style properties and engine custom rendering properties
- class/widget_class/widget name matchers
- association matchers → style classes
GTK2 Theming - Problems

- GtkRC base syntax very limited and not expressive
- dozens of very specialized different engines
- modifying an engine is difficult (C code...)
- no standard way to render a desired effect
- no standard way of testing regressions
- accessing GTK internals from the engine
- weak separation between content and presentation
CSS is a markup language designed to enable separation between a document content and its presentation semantics

- W3C standard
- Well-known syntax, documentation widely available on the web
- Natively supports a large superset of the GtkRC features
- No need to worry about whether a feature is supported in a specific engine
- Well-maintained dynamic standard, in continuous evolution driven by the web
CSS - Clear semantics

Using CSS enables a clear disambiguation on the meaning of style properties

- style properties semantics are predictable, and can be tested (*reftests*)
- CSS Box model (padding, margin, border)
- inheritance
- font properties
- *shorthand* properties
CSS - Cool Effects

CSS3 specification draft → family of appealing/rich visual features and effects
- border-image
- box-shadow
- text-shadow
- nth-child and siblings styling support
- gradient support (not yet formalized by W3C, supported as -gtk-gradient in GTK)
Not all the CSS style properties make sense in a toolkit like GTK

OTOH GTK might need some specific properties which wouldn’t make sense in the web

- icon-shadow
- transition
- add your favorite $property here...
GtkStyleContext → GtkStyle on steroids

- application-side interface to drawing and theming
- each widget holds its own different context
- independent from GtkWidget, operates on GtkWidgetPath structures
- GtkWidgetPath contains information for generic toplevel → child widget hierarchies
- easy styling of foreign toolkits (WebKit, QT, ...)

Cosimo Cecchi — Theming GTK3 widgets with CSS — Desktop Summit — Berlin
GTK3 Theming - GtkStyleContext

Style classes
.scrollbar → .slider + .trough + .button

- conceptually decompose a widget in a set of one or more base elements
- apply a style to each base widget element
- same widget → lots of possibilities without touching code
  .toolbar vs .primary-toolbar
Widget regions

GtkTreeView → row, column, column-header

- number of repeated elements of the same type in a widget not known
- named class + a set of order-based flags 
  even, odd, first, last, sorted
- use nth-child to match the desired flag from the CSS
Native \texttt{nth-child} support in containers

- construct a selector matching the position of an element in relation to its siblings
- works by default for widgets packed in \texttt{GtkBox} and \texttt{GtkToolbar}
- *animatable regions*
- of course get/set style properties as `GtkStyle` used to
GTK3 Theming - GtkThemingEngine

Theme-side counterpart of GtkStyleContext

Calls into cairo to do the actual rendering
GTK3 Theming - GtkThemingEngine

Theme-side counterpart of GtkStyleContext
- default implementation inside GTK+
- external theming engines → subclass + .so module library
- no access to GTK internals
- access to all the style information stored in GtkStyleContext
- register custom style properties
Where we want to be

You should not write a GtkThemingEngine subclass unless you have a very good reason to.

You should be able to do everything you need with CSS and SVG assets.
GTK3 Theming - Theme developers

Where we are right now

Not yet possible to write a full-featured complex theme (e.g. Adwaita) entirely in CSS and SVG

- features missing from GTK
  - multiple layer of backgrounds
  - inconsistent focus theming properties
- hard GTK limitations
  - not possible to render outside the widget allocation box
- working around widget bugs
GTK3 Theming - Theme developers

Where we are right now

Not yet possible to write a full-featured complex theme (e.g. Adwaita) entirely in CSS and SVG

But...

- Adwaita is now ~90% CSS and SVG
- about 1000 lines of C code
Upstream GTK is very receptive about improving the CSS theming engine and adding support for additional style properties

- if you’re writing a custom style property, please consider using a CSS standard and pushing it upstream
- file bugs, write patches, talk to us
Should applications completely specify a customized look like websites do?

Two schools of thought

1. applications know better
2. themes know better
The theme always knows better

Except when an application really needs to force a completely unique look altogether
The theme always knows better

- hardcoded colors in applications → basically unthemable
- the engine lacks constructs to fully describe a complex layout to match
  *the first sidebar left of a view widget*
- CSS3 has such constructs (nth-child and friends) - we should extend our support for them
- special-casing app widgets or layouts in the theme instead of hardcoding theme information in the apps
When an app needs to force an unique look altogether...

- Pro-oriented (e.g. Ardour)
- Games, educational or unconventional
- Special accessibility requirements
  - using a custom CSS theme makes a lot of things very easy
  - apps could even install and use their own theming engine module to bypass the default `gtk_render_*` implementations
GTK3 Theming - Future

- drop *Raleigh* as default GTK theme (?!)
- integrate documentation for the style classes defined by stock widgets
- tie style classes to HIG 3.0 recommendations
- multiple background compositing
- sanity-check our implementation of focus theming
- easier theming for GtkCellRenderers
- fully implement the CSS box-model logic in GTK
- make every widget capable of rendering a frame and a background
- improved use of implicit animations
Thanks for attending

Questions?