WebKit

open source web browser **engine**

What is it?

- It's **not** a web browser
- It's the engine for web browsers
- Fork of KHTML by Apple
- Composed of 2 sub libraries
  - WebCore (HTML rendering engine)
  - JavaScriptCore (JavaScript engine)
- Open source software (BSD and GNU LGPL)
- Follows standards (HTML, CSS, Acid tests)

Web Technologies

- HTML 4 and 5
- CSS 1,2 (almost complete) and 3 (incomplete)
- DOM (via JS, C++ and other languages)
- SVG (animations and partial webfonts)
- XML, XSLT (through libxslt)
- JavaScript

Used by

Browsers



OS

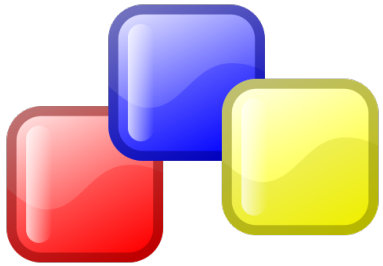
iOS 4 Kindle

Applications


STEAM™

Toolkits

Gtk



WxWidgets

Cocoa OS X

Adobe AIR

Implementation

- Written in C++
- Bindings available in
  - C/C++ (Gtk) / Objective-C (Cocoa)
  - Perl, Python, Ruby, Vala
  - Glib introspection
- HTTP layer
  - cURL
  - libsoup (Gtk)
  - Qt
  - Chromium

Gtk2::WebKit

- Simple API integrated with Gtk2
- Bindings XS (rafl - Florian Ragwitz)
- Advantages
  - API very “Perlish”
- Disadvantages
  - Some functions are not available
  - Some objects are not available

GObject 101

- Objects in C
- Simple inheritance
- API is language bindings friendly
- Introspection
- Objects have
  - Property (signal 'notify::PROPERTY-NAME')
  - Signals
  - Methods
  - Constructors

Example - Gtk2::WebKit

```
# Simple WebKit window - my first browser

use Gtk2 -init; # Initialize Gtk2
use Gtk2::WebKit; # Load WebKit

my $window  = Gtk2::Window->new();
my $scrolls = Gtk2::ScrolledWindow->new();
my $view    = Gtk2::WebKit::WebView->new(); # WebKit instance

$scrolls->add($view);
$window->add($scrolls);
$window->show_all();
$window->signal_connect(destroy => sub { Gtk2->main_quit() });

$view->load_uri('http://ba.pm.org/en'); # Load a page

Gtk2->main(); # Main loop
```

Glib Introspection

- Perl bindings auto generated through GIR
- The future of GObject and its by products
- Advantages
  - API coverage automatic
  - Generated at run-time (no compilation required!)
- Disadvantages
  - API not always very “Perlish” (getting closer)
  - Start up can be slow

Example - Introspection

```
use Glib::Object::Introspection;
Glib::Object::Introspection->setup(
    basename => 'Gtk',
   version => '2.0',
   package => 'Gtk2');
Glib::Object::Introspection->setup(
  basename => 'WebKit',
  version => '1.0',
  package => 'Gtk2::WebKit');
Gtk2::init(0, []);

my $window = Gtk2::Window->new('toplevel');
my $scrolls = Gtk2::ScrolledWindow->new();
my $view = Gtk2::WebKit::WebView->new(); # WebKit instance

$scrolls->add($view);
$window->add($scrolls);
$window->show_all();
$window->signal_connect(destroy => sub { Gtk2->main_quit });

$view->load_uri('http://ba.pm.org/en'); # Load a page

Gtk2->main(); # Main loop
```

My patches

- WebKit through GIR (WWW::WebKit)
- Bindings for libsoup through GIR (HTTP::Soup)
- Code and examples in github
  - https://github.com/potyl/Webkit
  - https://github.com/potyl/perl-WWW-WebKit
  - https://github.com/potyl/perl-HTTP-Soup

WWW::WebKit

```
package WWW::WebKit;

use warnings;
use strict;

our $VERSION = '0.01';

use Glib::Object::Introspection;

Glib::Object::Introspection->setup(
	basename => 'WebKit',
	version  => '1.0', # 1.0 -> Gtk2, 3.0 -> Gtk3
	package  => __PACKAGE__,
);

1;
```

HTTP::Soup

```
package HTTP::Soup;

use warnings;
use strict;

our $VERSION = '0.01';

use Glib::Object::Introspection;

Glib::Object::Introspection->setup(
	basename => 'Soup',
	version  => '2.4',
	package  => __PACKAGE__,
);

# A bit of XS
use base 'DynaLoader';
sub dl_load_flags { $^O eq 'darwin' ? 0x00 : 0x01 }
__PACKAGE__->bootstrap($VERSION);

1;
```

What can we do with WebKit?

- Another web browser ...
- Graphical interfaces with HTML
- Web pages profiling
- Automation through WebKit
  - JavaScript interaction
  - Network manipulation (via libsoup)
  - DOM (should be available in 1.5 through GIR)
  - Screenshots

We can automate this

Elements | Resources | Network | Scripts | Timeline | Profiles | Audits | Console

Search Network

Name Path	Method	Status Text	Type	Size Transfer	Time Latency	Timeline 972ms 1.46s 1.94s
data:image/png;base(	GET	Pending	image/png	3.34KB 3.34KB	15150.3 days 0	
:3001/	GET	200 OK	text/html	8.18KB 8.23KB	283ms 239ms	
maps maps.google.com	GET	Pending	Pending	0B 0B	147ms 0.0 days	
style.css /css	GET	200 OK	text/css	3.53KB 3.55KB	178ms 158ms	
custom.css /css	GET	200 OK	text/css	2.07KB 2.09KB	192ms 189ms	
jquery.min.1.5.0.js /js	GET	200 OK	application/j…	82.38KB 82.42KB	520ms 266ms	
jquery.tools.min.1.2.5 /js	GET	200 OK	application/j…	115.92KB 115.95KB	339ms 293ms	
maps.js /js	GET	200 OK	application/j…	3.20KB 3.24KB	346ms 345ms	
like.php www.facebook.com/plu	GET	Pending	Pending	0B 0B	197ms 0.0 days	

Code

Examples

Screenshots (png through gtk2)

```
# Grab a screenshot as soon as the page is loaded
$view->signal_connect('notify::load-status' => sub {

    # Patiently wait for the page to be loaded
    my $uri = $view->get_uri or return;
    return unless $view->get_load_status eq 'finished';

    # Grab the screenshot
    my $pixmap = $view->get_snapshot() or return;
    my $allocation = $view->allocation;
    my ($width, $height) = ($allocation->width, $allocation->height);

    # Save the image
    my $pixbuf = Gtk2::Gdk::Pixbuf->get_from_drawable(
        $pixmap, undef, 0, 0, 0, 0, $width, $height) or return;
    $pixbuf->save('capture.png', 'png');
});

# Run with  xvfb-run --server-args="-screen 0 1024x768x24" screenshot.pl "$@"
```

Screenshots (pdf through gtk3)

```
# Grab a screenshot through Cairo as soon as the page is loaded
$view->signal_connect('notify::load-status' => sub {

    # Wait for the page to be loaded
    return unless $view->get_uri and ($view->get_load_status eq 'finished');

    # Use Cairo to grab a PDF (we can also use SVG, PostScript or PNG)
    my ($width, $height) = ($view->get_allocated_width, $view->get_allocated_height);
    my $surface = Cairo::PdfSurface->create($filename, $width, $height);
    my $cr = Cairo::Context->create($surface);
    $view->draw($cr);
});


# With Gtk3 we can use offscreen rendering!
 my $window = Gtk3::OffscreenWindow->new();
 $window->add($view);
 $window->show_all();
```

Execute JavaScript

```
# Execute a JavaScript commad as soon as the page is loaded.
$view->signal_connect('notify::load-status' => sub {

        # Wait for the page to be loaded
        my $uri = $view->get_uri or return;
        return unless $view->get_load_status eq 'finished';

        # Let's call jQuery (the page must have jQuery loaded already)
        $view->execute_script(q{
                $('img').hide();
        });
});
```

Resource tracking

```
# Load the bindings for libsoup
use HTTP::Soup;

# Get the session that's responsible for all HTTP access made by WebKit
my $session = WWW::WebKit->get_default_session();

# Track all new download requests
$session->signal_connect('request-started' => sub {
        my ($session, $message, $socket, $resources) = @_;

        # A new download request
        my ($uri, $start) = ($message->get_uri->to_string, time);

        # Track the when the download time
        $message->signal_connect('finished' => sub {
                my $end = time;
                my $elapsed = $end - $start;
                my $status_code = $message->get('status-code') // 'N/A';
                printf “%s in %.2f seconds; code: %s\n", $uri, $elapsed, $status_code;
    });
});
```

Nanny

```
my $allowed_host_port = URI->new($uri)->host_port;

# Intercept all web pages requests and reject all requests that will bring us to an external web
# site. This works only for the iframes and links that have been clicked by the user.
# JavaScript and CSS are not blocked!
$view->signal_connect('navigation-policy-decision-requested' => sub {
        my ($view, $frame, $request, $action, $decision) = @_;

        # Accept the request only if we stay in the same site
        my $host_port = URI->new( $request->get_uri )->host_port;
        return FALSE if $host_port eq $allowed_host_port; # Default behavior: download!

        # Going to a different site
        print "Access denied $host_port\n";
        $decision->ignore(); # We reject the download request
        return TRUE;
});
```

~~Super~~ Über Nanny

```
my $allowed_host_port = URI->new($uri)->host_port;

# Block ANY resource that goest to an external site. This works for all kinds of resources.
# Even for resources that are built at runtime through JavaScript.
$view->signal_connect('resource-request-starting' => sub {
        my ($view, $frame, $resource, $request, $response) = @_;

        my $host_port = URI->new( $request->get_uri )->host_port;
        return if $host_port eq $allowed_host_port;

        # Download any request to an external URI. WebKit doesn't download resources that
        # point to  'about:blank'
        print "Access denied $host_port\n";
        $request->set_uri('about:blank');
});
```

</end>

Questions?