Google Web Toolkit
What, Why, and How

Bruce Johnson
Google, Inc.
bruce@google.com

Topics

<u>**A Simpler-Than-Possible Explanation of GWT**</u>

Why AJAX Matters

GWT is Software Engineering for AJAX

Common Questions

Big Applications

Summary

Q & A

What is Google Web Toolkit (GWT)?

What is GWT?

A set of tools for building AJAX apps in the Java language

What makes GWT interesting?

Write, run, test, and debug in Java

Isn't that called an applet?

Deploy as JavaScript

GWT converts your working Java source into equivalent JavaScript

GWT is a compiler?

GWT has a compiler, but the full story is more interesting

Topics

A Simpler-Than-Possible Explanation of GWT

<u>Why AJAX Matters</u>

GWT is Software Engineering for AJAX

Common Questions

Big Applications

Summary

Q & A

What I Mean by "AJAX"

Updating the browser UI without switching pages

- Traditionally called Dynamic HTML (DHTML)
- Relies on JavaScript running to direct the UI updates

Fetching data without switching pages

- Using XmlHttpRequest (XHR) to fetch data in the background

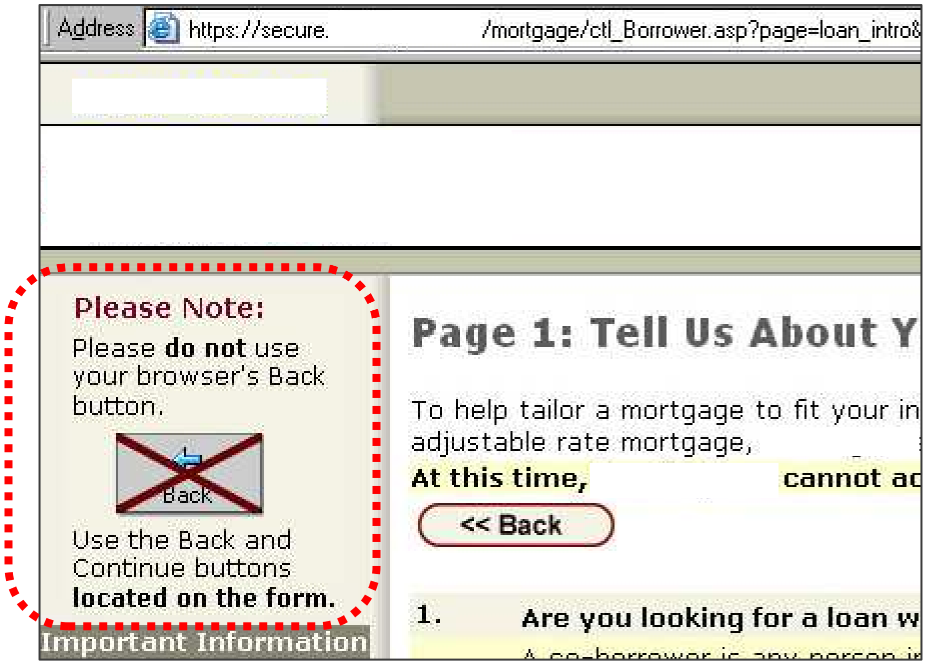
Viewing browsers as smart clients

- Instead of HTML dumb terminals
- Sharing the computational burden
- Better server utilization
- Applications that are more responsive than classic HTML

In other words, AJAX is recreating 1990s-style client/server computing without the need to install software locally

Why AJAX? Usability Benefits

"Do not use your browser's Back button"



Address https://secure. /mortgage/ctl_Borrower.asp?page=loan_intro&

Please Note:

Please **do not** use your browser's Back button.

Use the Back and Continue buttons **located on the form.**

Important Information

Page 1: Tell Us About Y

To help tailor a mortgage to fit your in
adjustable rate mortgage,

At this time, cannot ac

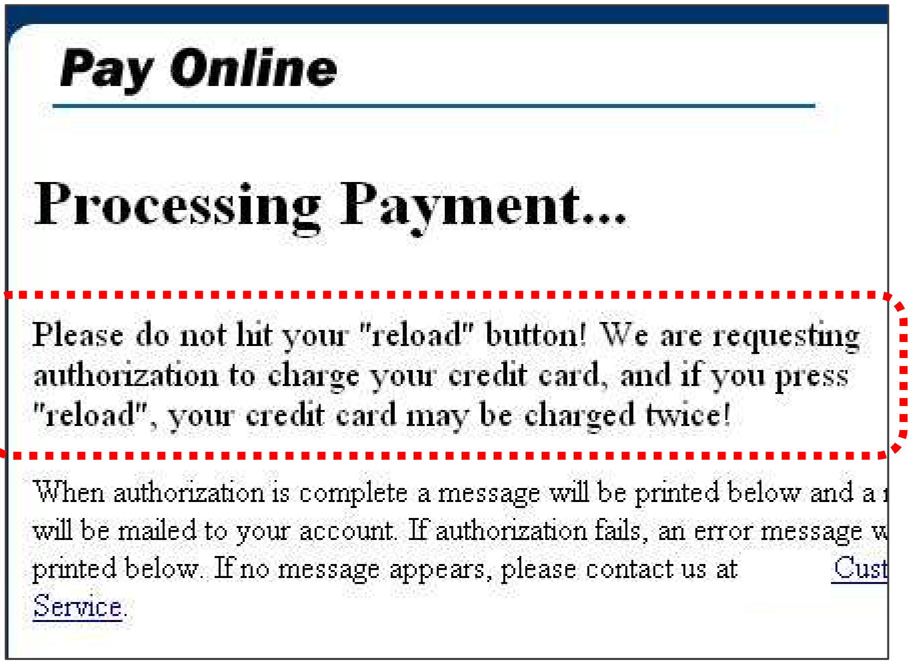
1. **Are you looking for a loan w**

A co-borrower is any person in

What if I <u>do</u> click Back?
AJAX can (in theory) solve this

Why AJAX? More Usability Benefits

"Don't hit reload or we'll charge you twice!"



Pay Online

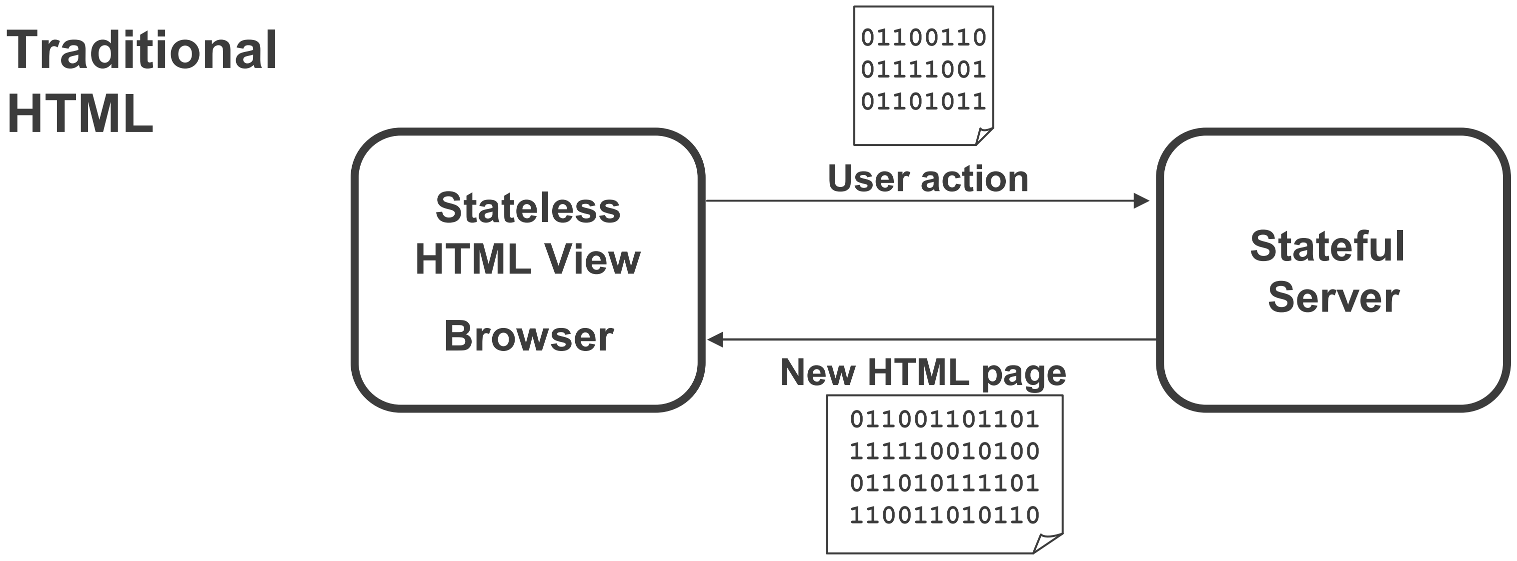
Processing Payment...

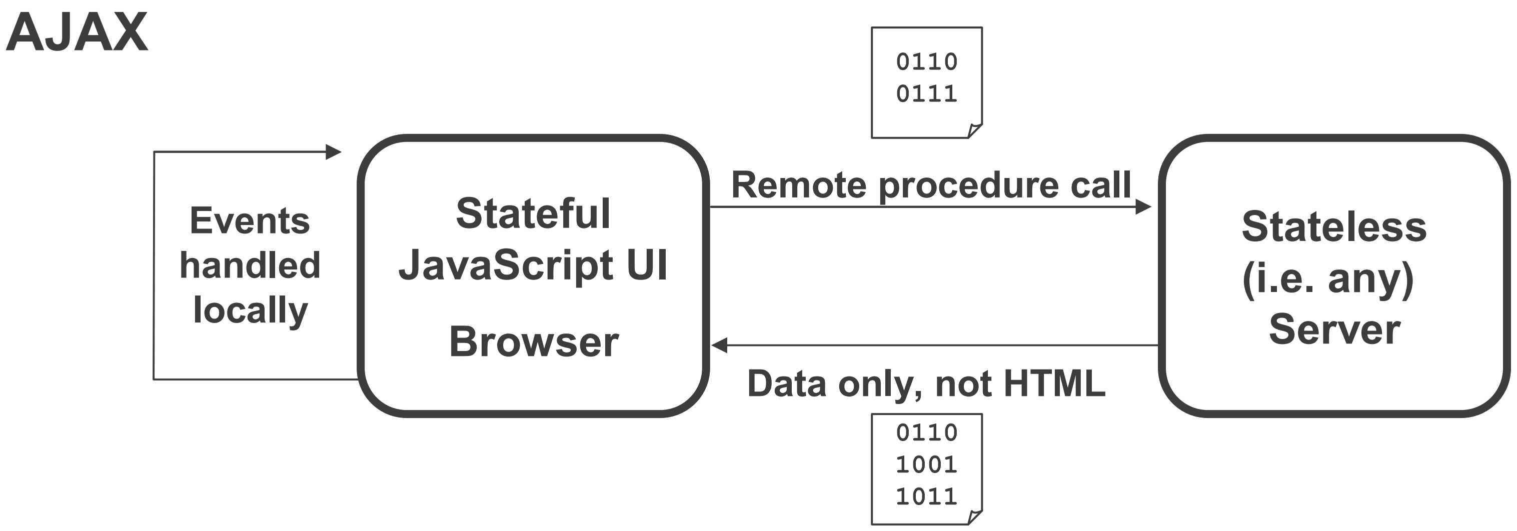
Please do not hit your "reload" button! We are requesting authorization to charge your credit card, and if you press "reload", your credit card may be charged twice!

When authorization is complete a message will be printed below and a r
will be mailed to your account. If authorization fails, an error message w
printed below. If no message appears, please contact us at Cust
Service.

What if the network hangs? What should I do?
AJAX can (in theory) solve this

Why AJAX? Scalability





Topics

Product Risks of Handwritten Script

Poor Usability

- No history
- No bookmarks
- Frozen browser chrome and pegged user CPUs
- Worst: Easy development and good usability are conflicting goals

Poor Browser Portability

- Hard to test: every line of code is a potential portability bug
- Either wrap every single browser-related call (heavy)...
- Or be paranoid about every line of code (risky)

Poor Speed

- Startup time is an extremely huge sacrifice...probably not worth it
- Large scripts run more slowly
- Worst: Maintainability and efficiency are conflicting scripting goals

Development Risks of Handwritten Script

Poor Tool Support

- Limited IDE support
- Debugging too often boils down to window.alert()
- Profilers? Code coverage? Findbugs? …

Quality Risks

- New categories of runtime-only bugs (e.g. spelling bugs)
- Poor JS reuse model encourages "from scratch" or copy/paste
- Browsers are a moving target

Long-Term Risks

- Hard to schedule (e.g. unexpected browser quirks)
- Spaghetti risk
- Poor documentation
- Hard for large teams to work on the same code base
- Hard enough to find one AJAX guru

Spelling Bugs

typos + expandos = bug-o-s

Imagine this gem on line 5912 of your script

```
x.compnent = document.getElementById("x");
// a spelling(!) bug that will bite much later
```

There’s a reason static type checking was invented

Reuse is a good way to not write bugs

Don’t forget code completion

This starts to matter a lot for big projects

To Be Crystal Clear

It is very easy to slip into making a poorly planned AJAX investment

**...but you'll live
with the consequences
for a long, long time**

GWT Requirements Laundry List

Make great AJAX apps that are still very webby

- Familiar UI, History, Bookmarks, a working Back button...

Leverage the Java language, developers, and technologies

IDEs, debugging, unit testing, profiling, and coverage

Portable across browsers with low or no overhead

Reuse at the Java language level via jars

Fast, simple RPC based on Java classes

Extreme scalability

More or less, the impossible…

Unless you translate Java into JavaScript :-)

Code Sample – Hello, AJAX

Demo time...

```
public class Hello implements EntryPoint {

  public void onModuleLoad() {
    Button b = new Button("Click me", new ClickListener() {
      public void onClick(Widget sender) {
        Window.alert("Hello, AJAX");
      }
    });

    RootPanel.get().add(b);
  }

}
```

Demo

Hello, AJAX

Wow, That's So Much Easier

Redefining the problem has been fruitful

No server-side session state required

No round trips for UI updates and event handling

Deployment? No fancy server, just compiled JS

Leverage for the biggest AJAX headaches

- Our Mantra: Solve the problem once & wrap it in a class
- History? Create a History class
- Cross-browser? Create an abstract DOM class
- RPC? Create an all-Java RPC mechanism

Rich UI: Widgets and Layout

Build (or reuse!) widgets

- Written in straight Java
- Code without worrying about browser portability

Separate UI style from logic

- Widgets are styled with CSS
- Automatically load the right CSS for your widgets

Demo

- "Mail" is a desktop-style application

Usability and Performance

Demo: User Admin Dialog Box

GWT saves you round trips

Very fast startup time

Separation of concerns in the code

Keyboard support

On-the-fly font resizing

Reduce server load <u>and</u> improve usability

Usability: History and Bookmarks

History is the first thing to go in most AJAX apps

With GWT, it's easy and works well with MVC

```
History.addHistoryListener(myController);
```

History support leads to bookmark support

http://google.com/gulp.html#beta_carroty

Demo

"KitchenSink" shows history, bookmarking, and widgets

Simple & Powerful Client/Server: RPC

Many solutions out there (JSON, XML-RPC, …)

But a pure Java RPC interface sure is nice!

```
interface SpellingService extends RemoteService {
  /**
   * Checks spelling and suggests alternatives.
   * @param the word to check
   * @return the list of alternatives, if any
   */
  String[] suggest(String word)
}
```

Client and server can speak the same language

Demo

- "DynaTable" loads records dynamically

Topics

A Simpler-Than-Possible Explanation of GWT

Why AJAX Matters

GWT is Software Engineering for AJAX

<u>Common Questions</u>

Big Applications

Summary

Q & A

Ten Things to Know About GWT

10. The GWT mission statement puts technology second

To radically improve the web experience for users by enabling developers to use existing Java tools to build no-compromise AJAX for any modern browser

See "Making GWT Better" for the full story

9. It isn't GWT vs. Everybody Else

Not sure why so many people want to couch it this way

We're not into "smackdowns" because using GWT doesn't mean foregoing another technology; mix and match is ideal

We've gone to a lot of trouble to make integration easy (JSNI)

8. It isn't Java vs. JavaScript – it's about leverage

No language wars! The goal of GWT isn't to hide JavaScript

We view GWT as a way to add leverage to JavaScript and DHTML

Leverage: Wicked Cool Optimizations

Tough decision not to support reflection and class loading

Worth it! Three words: Whole program optimization

For example, type tightening to eliminate polymorphism

```
Shape s = new Circle(2);  // radius of 2
double a = s.getArea();
```

can become

```
Circle s = new Circle(2); // radius of 2
double a = (s.radius * s.radius * Math.PI);
```

which, if Circle has no side effects, can become

```
double a = 12.5663706143591;
```

Imagine those sorts of optimizations across your entire app

In JavaScript, reducing size and increasing speed are complementary goals, which makes optimizations *really* fun

Ten Things to Know About GWT

7. We know that abstractions leak

There are only two kinds of abstractions:

- those that leak a lot
- those that leak a little

Embracing abstraction leaks makes better-educated users

UI leaks a lot, so we don't attempt to hide it

- Widget → Element → DOM → JSNI forms a useful continuum

RPC only leaks a little, mainly in that calls must be async

6. JavaScript Native Interface (JSNI)

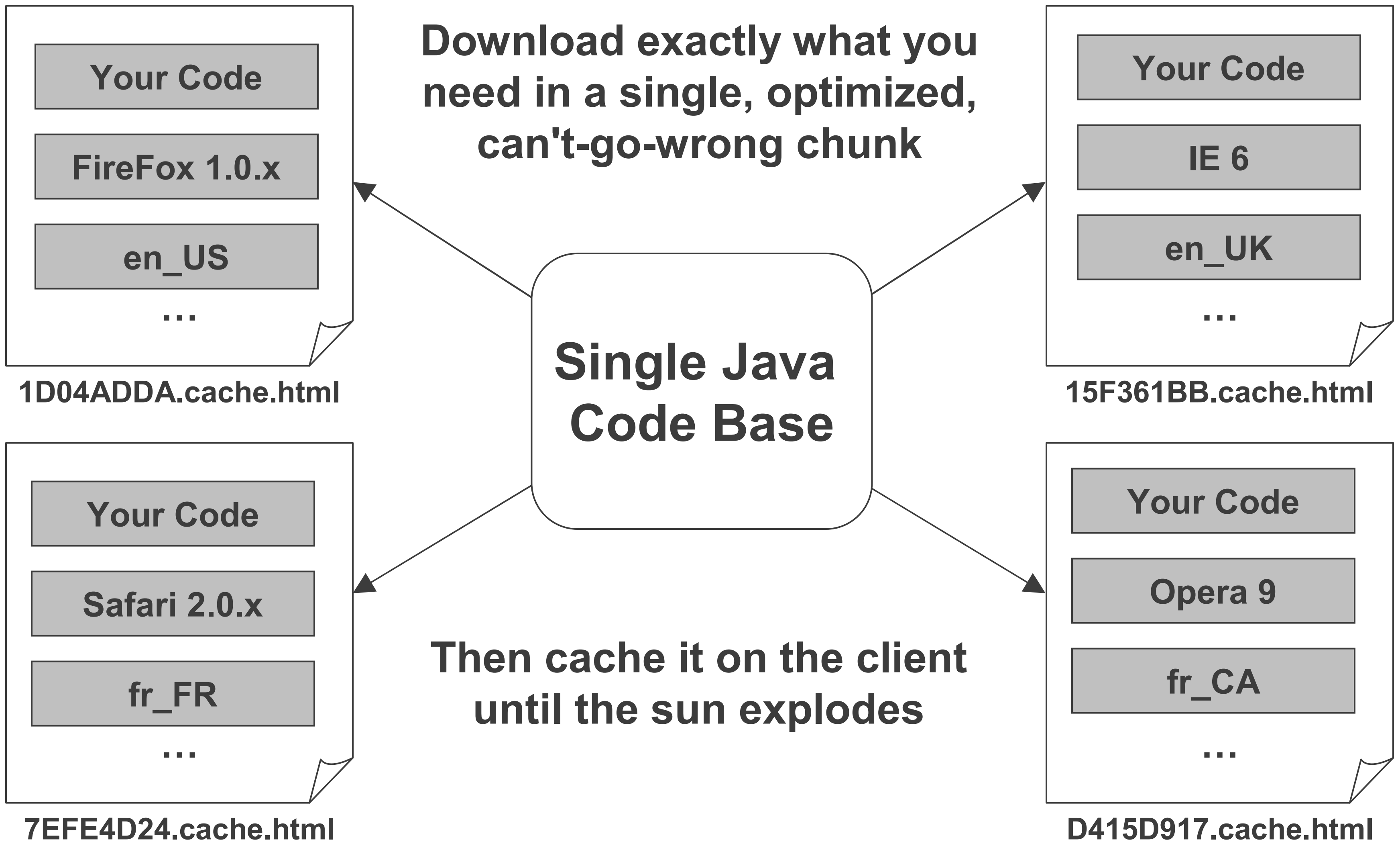
Implement **`native`** methods with JavaScript

5. Deferred binding with code generation

Manages permutations automatically

Totally extensible, including compile-time code generators

Optimized Permutations



Download exactly what you need in a single, optimized, can't-go-wrong chunk

Then cache it on the client until the sun explodes

Ten Things to Know About GWT

4. GWT doesn't try to blow you away with the first impression

- Our focus is on making a sensible, efficient set of tools that scales
- Supporting solid software engineering trumps snazzy widgets
- Team slogan: the bling is on the inside

3. Hosted mode is at least as cool as the GWT compiler

- Feels like normal browser development
- Refresh actually does recompile source to bytecode

2. GWT eats its own dogfood

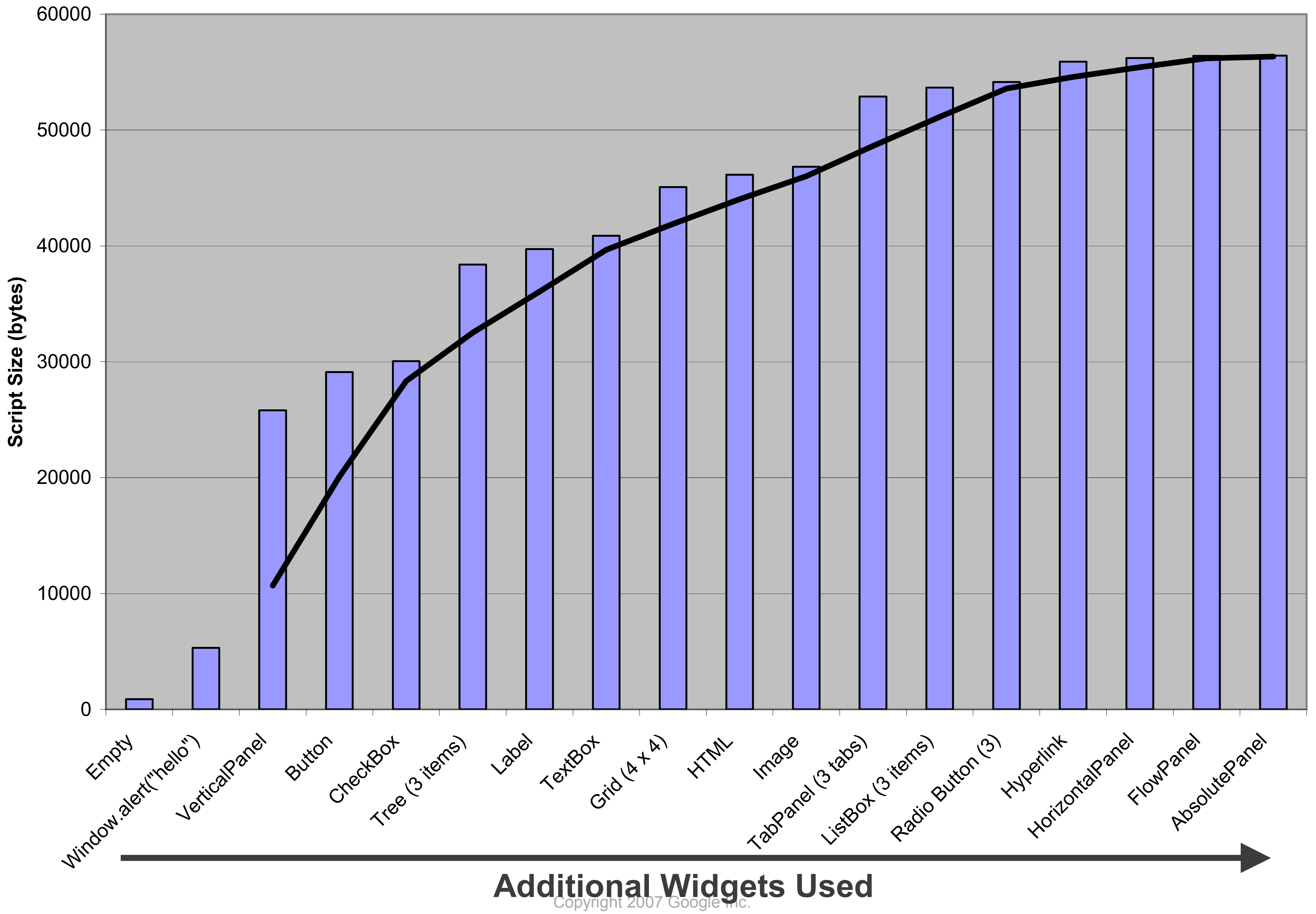
- Everything built with core facilities you can use yourself
- Browser portability, localization, RPC client proxies, …
- Upcoming ImageBundleGenerator

Ten Things to Know About GWT

1. GWT isn't all-or-nothing

- Only use what you want
- Don't pay for what you don't use
- Integrate with other technology as needed

Only Pay for What You Use



Common Questions

Which browsers are supported?

Firefox 1.0, 1.5, 2.0

Internet Explorer 6, 7

Safari 2.0

Opera 8.5, 9.0

Common Questions

What happens when a new browser comes out?
Do I have to wait for the GWT compiler to be updated?

Definitely no!

- All browser-specific code is in user-level libraries

The JavaScript language itself has very consistent support across browsers

- The DOM API is the real culprit

For backwards-compatible browsers, it's a no-brainer

For other situations, it's straightforward to change the user-level libraries

- Implement a version of DOMImpl for the desired browser

Main point: GWT was designed to never be a roadblock

Common Questions

Do I have to run Java on my server?

No, the GWT compiler produces standalone JS

Only GWT RPC needs a servlet

You can use any backend

GWT includes JSON and XML libraries to make it easier

Common Questions

Isn't it really hard to debug the JavaScript that the GWT compiler produces?

If you need to (or just want to) debug the compiled output, the GWT compiler gives you multiple output options:

-style OBFUSCATED (small, efficient, and fast)
-style DETAILED (nothing is left to the imagination)
-style PRETTY (perfect if you want to actually follow the code)

The output is normal JS, so you can always use any JavaScript debugger as you would with handwritten code.

By the way, you will likely never have to do any of this. You'll be doing your debugging in Java.

Common Questions

What functionality is included with GWT?

User Interface

Client/Server Communication

Application Infrastructure

Unit Testing

Internationalization

…

GWT Library Overview

AbsolutePanel, Button, ButtonBase, CellPanel, ChangeListenerCollection, CheckBox, ClickListenerCollection, ComplexPanel, Composite, DeckPanel, DialogBox, DockPanel, FileUpload, FlexTable, FlowPanel, FocusListenerAdapter, FocusListenerCollection, FocusPanel, FocusWidget, FormHandlerCollection, FormPanel, FormSubmitCompleteEvent, FormSubmitEvent, Frame, Grid, HorizontalPanel, HTML, HTMLPanel, HTMLTable, Hyperlink, Image, KeyboardListenerAdapter, KeyboardListenerCollection, Label, ListBox, LoadListenerCollection, MenuBar, MenuItem, MouseListenerAdapter, MouseListenerCollection, NamedFrame, Panel, PasswordTextBox, PopupListenerCollection, PopupPanel, RadioButton, RootPanel, ScrollListenerCollection, ScrollPanel, SimplePanel, StackPanel, TabBar, TableListenerCollection, TabListenerCollection, TabPanel, TextArea, TextBox, TextBoxBase, Tree, TreeItem, TreeListenerCollection, UIObject, VerticalPanel, Widget, WidgetCollection

User Interface

History, DeferredCommand, Localizable, Constants, Dictionary, ConstantsWithLookup, Messages

Usability and I18N

DOMException, XMLParser, Attr, CDATASection, CharacterData, Comment, Document, DocumentFragment, Element, EntityReference, NamedNodeMap, Node, NodeList, ProcessingInstruction, Text

XML

AsyncCallback, IsSerializable, RemoteService, RemoteServiceServlet

RPC

JSONArray, JSONBoolean, JSONException, JSONNull, JSONNumber, JSONObject, JSONParser, JSONString, JSONValue

JSON

Header, Request, RequestBuilder, RequestCallback, RequestException, Response, URL

HTTP

Common Questions

How big are GWT apps? Doesn't the compiler produce bloated script?

For tiny bits of functionality (say, < 100 lines) of handwritten JS, you might be better off writing it by hand. Beyond that, compiler size and speed optimizations will ultimately win.

Compiler optimizations that require static typing

- Dead code removal
- Type tightening
- Polymorphism removal
- Inlining (if you want it to be correct)
- Very aggressive (and safe!) compression on generated JS

We think of new optimizations all the time

- KitchenSink is 20% smaller (95K) using upcoming 1.4 optimizations
- GZipped (i.e. over the wire, once) it's 29K

Common Questions

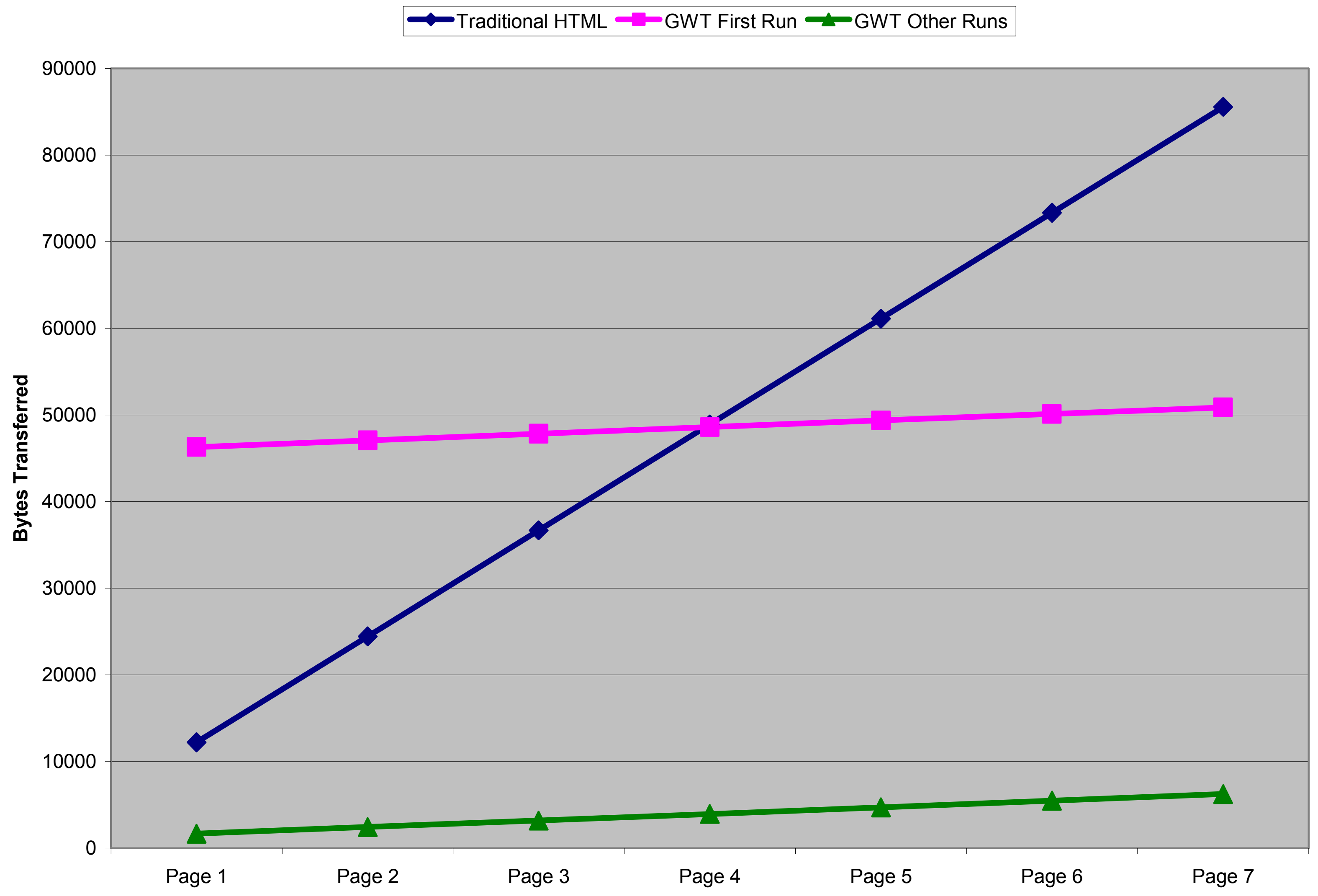
How fast are GWT apps?
Surely I could write faster apps by hand!

Likely to be true for very small apps

Unlikely to be true for bigger apps due to compiler and class library optimizations

(See next slide for experimental data)

Bandwidth and Startup Time



Common Questions

Does GWT have to control the entire page?
I can't rewrite my app from scratch!

GWT does not force you to start over!
Attach code to existing pages with a <meta> tag

```
<html>
…<meta name="gwt:module" content="…"/>
…<h1>Welcome to GWTravel Services</h1>
…<div id="reservationWizard">
…</html>
```

Your Java source is as loosely-coupled as you need it to be

```
Panel p = RootPanel.get("reservationWizard");
Wizard wiz = new ReservationWizard();
p.add(wiz);
```

Works with any HTML-generating server approach

Common Questions

How accessible are GWT applications?

GWT apps are as accessible as any AJAX app…and far from perfect

GWT does far more with keyboard support than typical AJAX

GWT is well-positioned to add comprehensive support when AJAX accessibility features are widely available in browsers

Topics

A Simpler-Than-Possible Explanation of GWT

Why AJAX Matters

GWT is Software Engineering for AJAX

Common Questions

<u>Big Applications</u>

Summary

Q & A

How Big is "Big"?

The obvious question that rarely gets asked

What exactly are we trying to optimize for?

Download speed?

- Are we supporting dial-up users?

Startup time?

- First run? Subsequent runs? How fast, exactly?

Some particular size cutoff?

- Size-on-wire? Size-in-cache?
- Is the cutoff arbitrary or based on measured effects?
- Funny: compare script size to the size of your images

Startup Time

Absolutely crucial

- Should be measured in milliseconds
- If startup time isn't acceptable, nothing else matters

Very hard to do well

- Loading code with synchronous XHR is out of the question
- <script> tags serialize HTTP requests
- GZip your script ahead of time? Good idea, but…
- Some versions of IE6 fail on gzipped .js files
- Script versioning vs. cacheability

GWT gives you leverage

- Compiled output includes only what a particular user needs
- Output is JS wrapped in HTML, which is safely gzip'able
- Loads code in an <iframe> in parallel with the page
- Scripts are named uniquely and are perfectly cacheable

Low-Hanging Fruit: Compression

Ahead-of-time script compression

- `C6BD1564339FC70220.cache.html` **(95K)**
- `C6BD1564339FC70220.cache.html.gz` **(29K)**

Our "big" app instantly became 3 times smaller

- You last build step should be to gzip GWT output

Classic HTML can't use compression so well

- Data changes frequently
- HTML changes rarely
- Mixing them forces compression into the critical path

Low-Hanging Fruit: Caching

GWT supports aggressive script caching

Combine a small "selection script"…

→ **`KitchenSink.nocache.html`**

```
Expires: <pretty soon>
```

With a larger compiled script…

→ ***`md5`*`.cache.html`**

```
Expires: <when the sun explodes>
```

Viola! Perfect caching!

Never re-fetch the big script *unless* it has changed

Never *fail* to re-fetch the big script when it *has* changed

GWT Unlocks Client-Side MVC

If you're confident that it's going to be a big app…

The default choice should be client-side MVC

Only tricky part is making your model async

Then again, not so bad…

```
myModel.requestNthItem(14);
…
class MyView implements MyModelListener {
   void onNthItemReceived(int n, Item item) {
…
```

MVC also fits perfectly with GWT history

An Algorithm to Evolve Your Architecture

1. **Start by assuming you have a single page and you're building a traditional client-side MVC app (remember client/server? :-)**
2. **Add code as if you'll never hit a brick wall**
3. **Make sure your app implements history well**
4. **Evaluate the size and speed of your app**
   A. If you're happy, goto 2
   B. If you're unhappy, do all the stuff on the previous slides
   C. If you're still unhappy, see the next slide

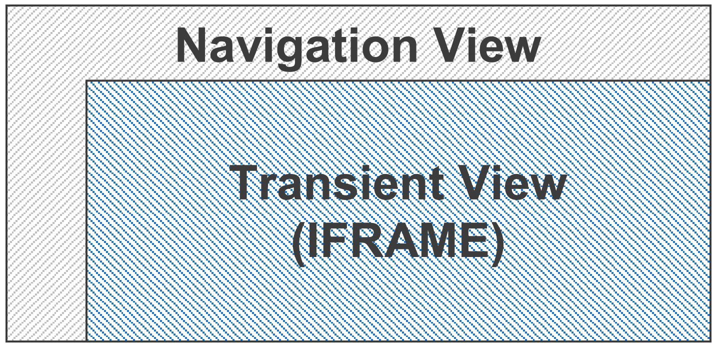
Composition Techniques

Not hard to split your GWT app across pages

- History smoothes over page transitions
- Fast GWT startup makes page switching affordable

Wrangled by GWT, IFRAMEs aren't so evil

- Divide big chunks into IFRAMEs that your controller shows/hides them as necessary



Additional Techniques

Consolidate multiple small RPCs

- Build composite structures and large-grained APIs
- Good rule of thumb: minimize HTTP round-trips
- Server replies with more data than was requested

Modularize your UI and create parts on demand

- Fits naturally with history and MVC
- Spread the cost of widget creation across user time
- See KitchenSink for an example

Topics

Not Enough Time to Show Everything

Internationalization support
- Highly optimized
- Externalized string ids are checked during compilation

Automatic, dynamic dependency inclusion
- Slurp in external CSS
- Slurp in external JS

Everything is cross-browser
- IE6+, FF 1.0.x, FF 1.5.x, Safari 2.0.x, Opera 9.x

Your choice of development platforms
- Mac OS X, Linux, Windows

Your choice of IDEs
- IntelliJ IDEA, Eclipse, NetBeans, JCreator, JBuilder

GWT is Open Source

Licensed under Apache 2.0

- Source is available via svn on Google Code project hosting

Our charter document is "Making GWT Better"

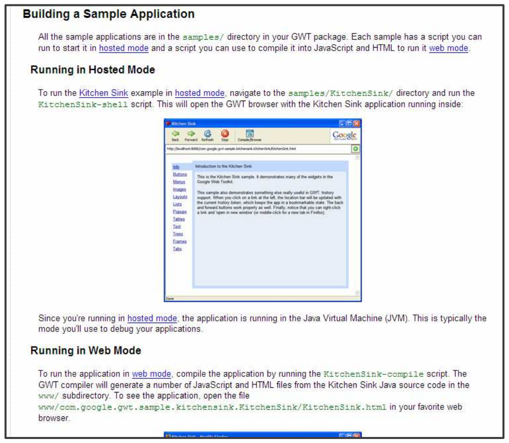
- Mission statement
- Design axioms
- Community forums
- How to build GWT from source
- Code style
- Submitting patches
- Transparent development (published minutes, roadmap)

Great participation

- 100,000+ downloads of the release candidates for GWT 1.3
- Great discussion on G-W-T and G-W-T-C lists
- 200+ developers on the contributors list
- Patches are rolling in!

Documentation Included

Getting Started Guide



Building a Sample Application

All the sample applications are in the `samples/` directory in your GWT package. Each sample has a script you can run to start it in hosted mode and a script you can use to compile it into JavaScript and HTML to run it web mode.

Running in Hosted Mode

To run the Kitchen Sink example in hosted mode, navigate to the `samples/KitchenSink/` directory and run the `KitchenSink-shell` script. This will open the GWT browser with the Kitchen Sink application running inside:

Introduction to the Kitchen Sink

This is the Kitchen Sink sample. It demonstrates many of the widgets in the Google Web Toolkit.

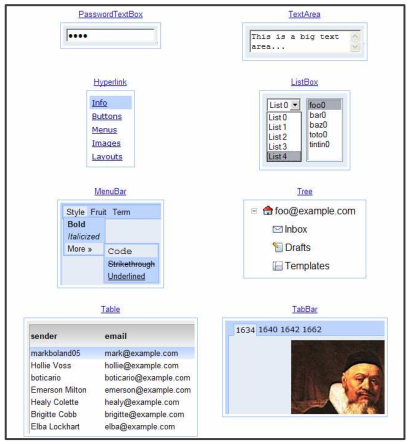
This sample also demonstrates something else really useful in GWT: history support. When you click on a link at the left, the location bar will be updated with the current *history token*, which keeps the app in a bookmarkable state. The back and forward buttons work properly as well. Finally, notice that you can right-click a link and 'open in new window' (or middle-click for a new tab in Firefox).

Since you're running in hosted mode, the application is running in the Java Virtual Machine (JVM). This is typically the mode you'll use to debug your applications.

Running in Web Mode

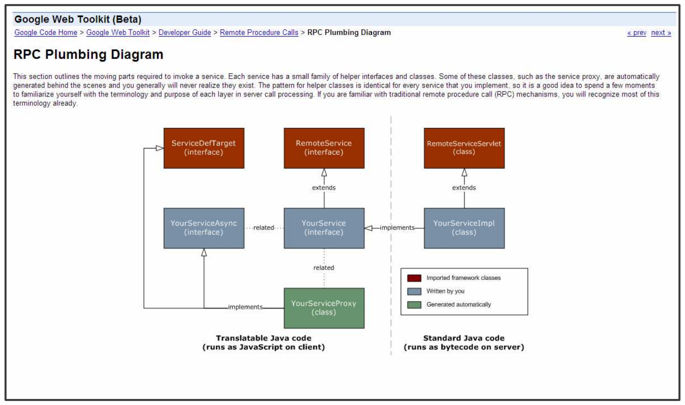
To run the application in web mode, compile the application by running the `KitchenSink-compile` script. The GWT compiler will generate a number of JavaScript and HTML files from the Kitchen Sink Java source code in the `www/` subdirectory. To see the application, open the file `www/com.google.gwt.sample.kitchensink.KitchenSink/KitchenSink.html` in your favorite web browser.

Widget Gallery



sender	email
markboland05	mark@example.com
Hollie Voss	hollie@example.com
boticario	boticario@example.com
Emerson Milton	emerson@example.com
Healy Colette	healy@example.com
Brigitte Cobb	brigitte@example.com
Elba Lockhart	elba@example.com

Developer Guide



Google Web Toolkit (Beta)

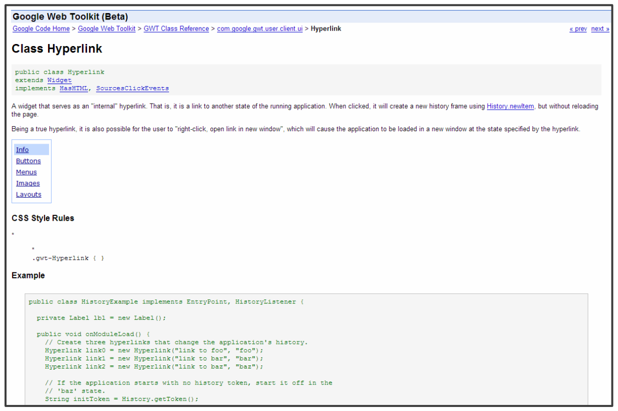
Google Code Home > Google Web Toolkit > Developer Guide > Remote Procedure Calls > **RPC Plumbing Diagram** « prev next »

RPC Plumbing Diagram

This section outlines the moving parts required to invoke a service. Each service has a small family of helper interfaces and classes. Some of these classes, such as the service proxy, are automatically generated behind the scenes and you generally will never realize they exist. The pattern for helper classes is identical for every service that you implement, so it is a good idea to spend a few moments to familiarize yourself with the terminology and purpose of each layer in server call processing. If you are familiar with traditional remote procedure call (RPC) mechanisms, you will recognize most of this terminology already.

ServiceDefTarget (interface)
RemoteService (interface)
RemoteServiceServlet (class)
extends
extends
YourServiceAsync (interface)
related
YourService (interface)
implements
YourServiceImpl (class)
related
implements
YourServiceProxy (class)
Imported framework classes
Written by you
Generated automatically
Translatable Java code (runs as JavaScript on client)
Standard Java code (runs as bytecode on server)

Class Reference



Google Web Toolkit (Beta)

Google Code Home > Google Web Toolkit > GWT Class Reference > com.google.gwt.user.client.ui > **Hyperlink** « prev next »

Class Hyperlink

```
public class Hyperlink
extends Widget
implements HasHTML, SourcesClickEvents
```

A widget that serves as an "internal" hyperlink. That is, it is a link to another state of the running application. When clicked, it will create a new history frame using History.newItem, but without reloading the page.

Being a true hyperlink, it is also possible for the user to "right-click, open link in new window", which will cause the application to be loaded in a new window at the state specified by the hyperlink.

CSS Style Rules

```
*

     *
     .gwt-Hyperlink { }
```

Example

```
public class HistoryExample implements EntryPoint, HistoryListener {

  private Label lbl = new Label();

  public void onModuleLoad() {
    // Create three hyperlinks that change the application's history.
    Hyperlink link0 = new Hyperlink("link to foo", "foo");
    Hyperlink link1 = new Hyperlink("link to bar", "bar");
    Hyperlink link2 = new Hyperlink("link to baz", "baz");

    // If the application starts with no history token, start it off in the
    // 'baz' state.
    String initToken = History.getToken();
```

Large and Growing GWT Community

Community and Support

- 7200+ members on the developer forum
- Books and articles
- Meta-sites (gwtsite.com, gwtPowered.org)

Libraries and Applications

- 86 GWT-related projects on Google Code project hosting
- Diverse products built with GWT
  - Google Base (base.google.com)
  - Google Image Labeler (images.google.com/imagelabeler)
  - Whirled (http://www.threerings.net/whirled)
  - Web-based conferencing (dimdim.com)
  - Texas Hold 'em with live chat (gpokr.com)

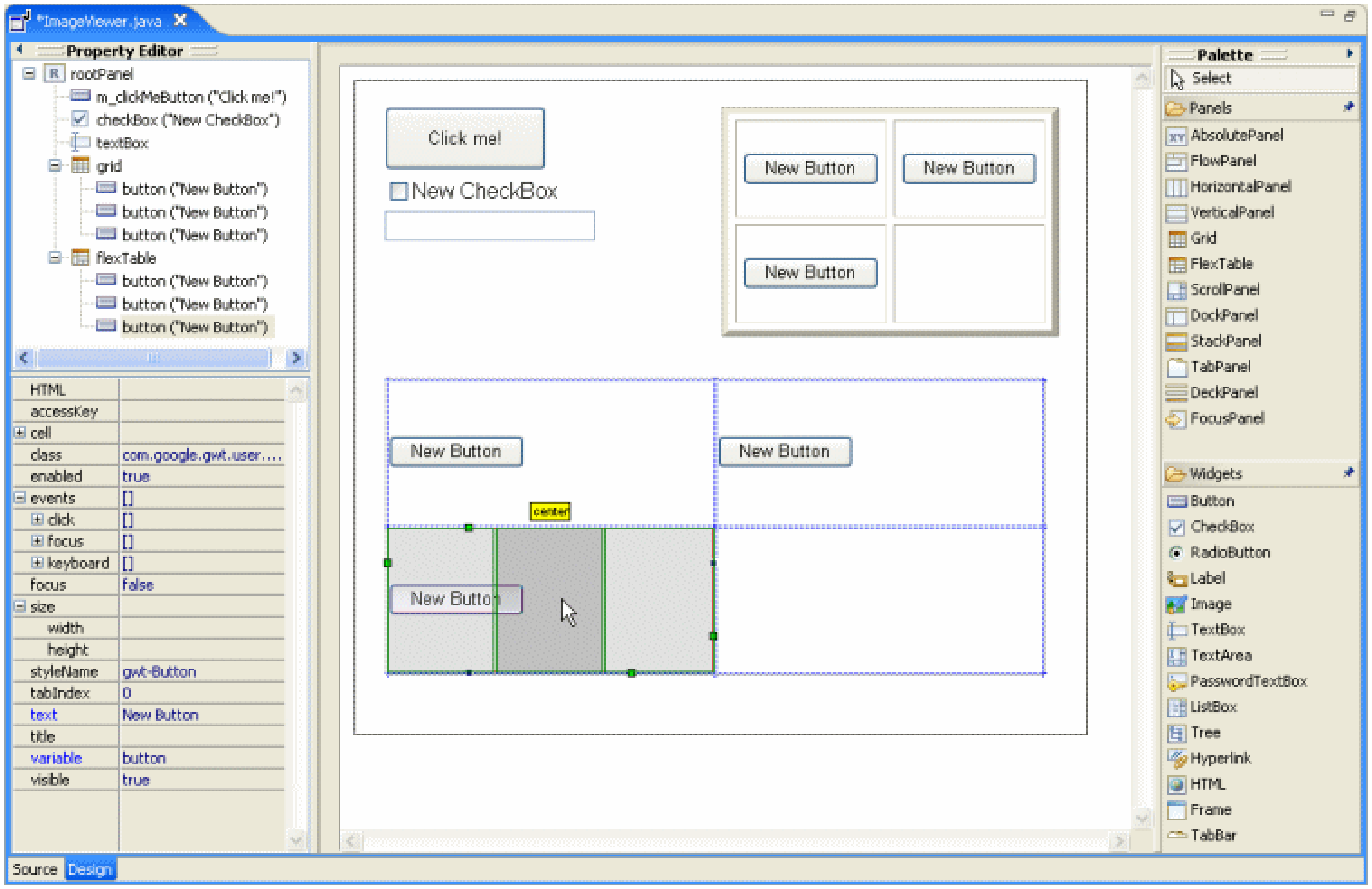
Tools, Tools, Tools

- IntelliJ IDEA, WindowBuilderPro GUI designer for GWT
- VistaFei for GWT, Googlipse, and others

Did I Mention Tools?

Instantiations GWT Designer

WYSIWYG Layout

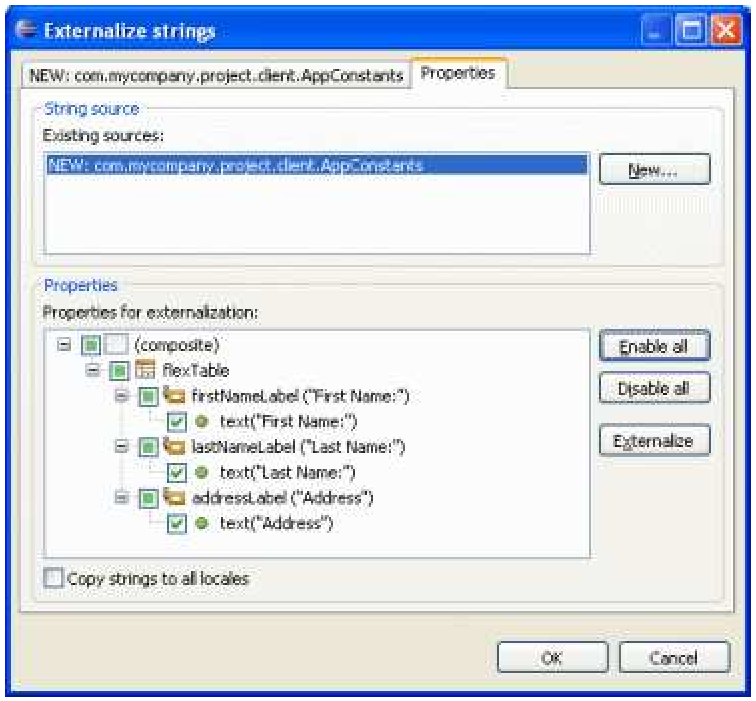


HTML	
accessKey	
⊞ cell	
class	com.google.gwt.user....
enabled	true
⊟ events	[]
⊞ click	[]
⊞ focus	[]
⊞ keyboard	[]
focus	false
⊟ size	
width	
height	
styleName	gwt-Button
tabIndex	0
text	New Button
title	
variable	button
visible	true

Did I Mention Tools?

Instantiations GWT Designer

Internationalization


Externalize strings

NEW: com.mycompany.project.client.AppConstants | Properties

String source

Existing sources:

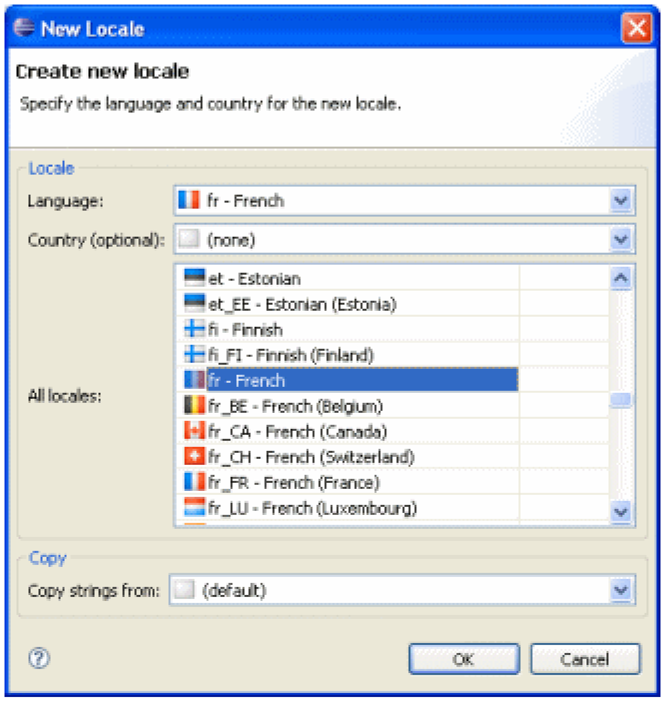
NEW: com.mycompany.project.client.AppConstants

Properties

Properties for externalization:

```
⊟ (composite)
   ⊟ flexTable
      ⊟ firstNameLabel ("First Name:")
         ✓ text("First Name:")
      ⊟ lastNameLabel ("Last Name:")
         ✓ text("Last Name:")
      ⊟ addressLabel ("Address")
         ✓ text("Address")
```

☐ Copy strings to all locales


New Locale

Create new locale

Specify the language and country for the new locale.

Locale

Language: fr - French

Country (optional): (none)

All locales:

- et - Estonian
- et_EE - Estonian (Estonia)
- fi - Finnish
- fi_FI - Finnish (Finland)
- fr - French
- fr_BE - French (Belgium)
- fr_CA - French (Canada)
- fr_CH - French (Switzerland)
- fr_FR - French (France)
- fr_LU - French (Luxembourg)

Copy

Copy strings from: (default)

Coming in GWT 1.4

New widgets

RichText, SpellCheck, SuggestBox, SplitterPanel, …

Simplifications and optimizations

Include GWT modules using only <script> tag

After first run, re-download is ~4K (80% less than 1.3)

Compiled script now can be fetched cross-domain

Compiler optimizations; typical reduction of 10-20%

New compiler output supports a better gzip ratio :-)

Utilities

ImageBundleGenerator, IncrementalCommand,

Benchmarking subsystem

Major speed improvements in collection classes

Date and number formatting and parsing

Summary: AJAX and GWT

Leverage is needed to use AJAX well with low risk

PhD in browser quirks is no longer a hiring prereq

Turn AJAX development into software engineering

GWT rewards using good engineering practices

We will share our best work and ideas with you, and we hope you will return the favor

Much more to come…see you online!

Topics

Q&A