

A decorative graphic at the top of the slide features four overlapping spheres. From left to right, they are light green, light blue, light red, and light yellow. The spheres are partially cut off by the top edge of the slide.

Gmail: Past, Present, and Future

Adam de Boor, Google



The Plan

Where we came from
Where we're at
What we've learned
Where we're going



Where We Came From

The screenshot shows the Gmail web interface from 2004. At the top left is the Gmail logo with "by Google" and "BETA" text. To the right, there's a search bar with "Search Mail" and "Search the Web" buttons. Below the search bar are links for "Show search options" and "Create a filter". The main navigation area on the left includes "Compose Mail", "Inbox (1)", "Starred", "Sent Mail", "All Mail", "Spam", and "Trash". A "Labels" section is visible with a "friends" label and an "Edit labels" link. The main content area displays a list of emails with checkboxes, star icons, and timestamps. The list includes emails from "NewsScan (HTML)" and "Gmail Team". At the bottom, there's a storage usage notification: "You are currently using 0 MB (0%) of your 1000 MB. Visit settings to save time with keyboard shortcuts!". Below this are links for "Terms of Use", "Privacy Policy", "Program Policies", and "Google Home", followed by the copyright notice "©2004 Google".

Gmail
by Google BETA

@gmail.com | [Feedback](#) | [Contacts](#) | [Settings](#) | [Help](#) | [Sign out](#)

Search Mail Search the Web

Show search options Create a filter

Compose Mail Archive More actions... Refresh 1 - 7 of 7

Inbox (1) Select: All, Read, Unread, Starred, Unstarred, None Apply label...

<input type="checkbox"/>	<input type="checkbox"/>	NewsScan (HTML)	Welcome to mailing list newsscan-html - Welcome ...	Apr 2
<input type="checkbox"/>	<input type="checkbox"/>	Gmail Team	Gmail is different. Here's what you need to know. - ...	Apr 2
<input type="checkbox"/>	<input type="checkbox"/>	Starred		1:56pm
<input type="checkbox"/>	<input type="checkbox"/>	Sent Mail		1:53pm
<input type="checkbox"/>	<input type="checkbox"/>	All Mail		1:43pm
<input type="checkbox"/>	<input type="checkbox"/>	Spam		8:56am
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Trash		Apr 2

Select: All, Read, Unread, Starred, Unstarred, None 1 - 7 of 7

Archive More actions...

You are currently using 0 MB (0%) of your 1000 MB.
Visit [settings](#) to save time with **keyboard shortcuts!**

[Terms of Use](#) - [Privacy Policy](#) - [Program Policies](#) - [Google Home](#)

©2004 Google

Where We Came From - 2004

April 1, 2004

- Slick webmail app using AJAX
- Plain-text compose
- 9400 lines of JS, downloaded as a block, in a frame
- JS compiler:
 - Condense code
 - Catch references to unknown vars
 - Verify function arity
 - Interpolate constants
 - Remove dead code
- No use of object classes
- HTML constructed as 'str' + var + 'str'
- Uses iframes for different views (switched)
- CSS in created STYLE elements



Where We Came From - 2005

April 1, 2005

- Added "web 1.0" HTML interface
- Rich formatting in compose
- Now 22,000 lines of JS (+ 10,000 lines of comments)
 - Still one download
- 12 non-US languages
 - JS compiler used to find and replace messages
- JS compiler now also inlines functions



Where We Came From - 2006

- Chat launched in February
- 52,000 lines of JS (75,000 with comments)
 - 4 modules - main broken into blocks
 - Classes
 - Start of Closure library
- 30 languages
- CSS still generated in JS
- Code base getting unwieldy
 - Combinations exploding
 - JS compiler looks for "frequently wrong" patterns



Where We Came From - 2007

- Rewrite! (shipped in October)
 - Make code base manageable (object classes!)
 - Speed
- 90,000 lines of JS (187,000 with comments) in 31 modules + libraries
- New module system
 - Dependency graph
 - Mods
- Closure Templates
 - Way easier than string concatenation
 - Automatic escaping
- Macro processing of CSS, served from server
- Simple type checking in JS compiler + optimizations



Where We Came From - 2008

- Innovation speed increased
 - Gmail Labs
 - Themes
 - New feature launched / week
- JS Compiler: better type checking, type-based code stripping, more optimizations
- 190k lines of JS (403k with comments)

Where We Came From - 2009

- Out of beta
- Offline using Google Gears
- JS Compiler: data-flow analysis, more checks, move functions/methods between modules
- 279k lines of JS (610k with comments)
- Released Closure library, compiler, and templates
- Released Open GSE

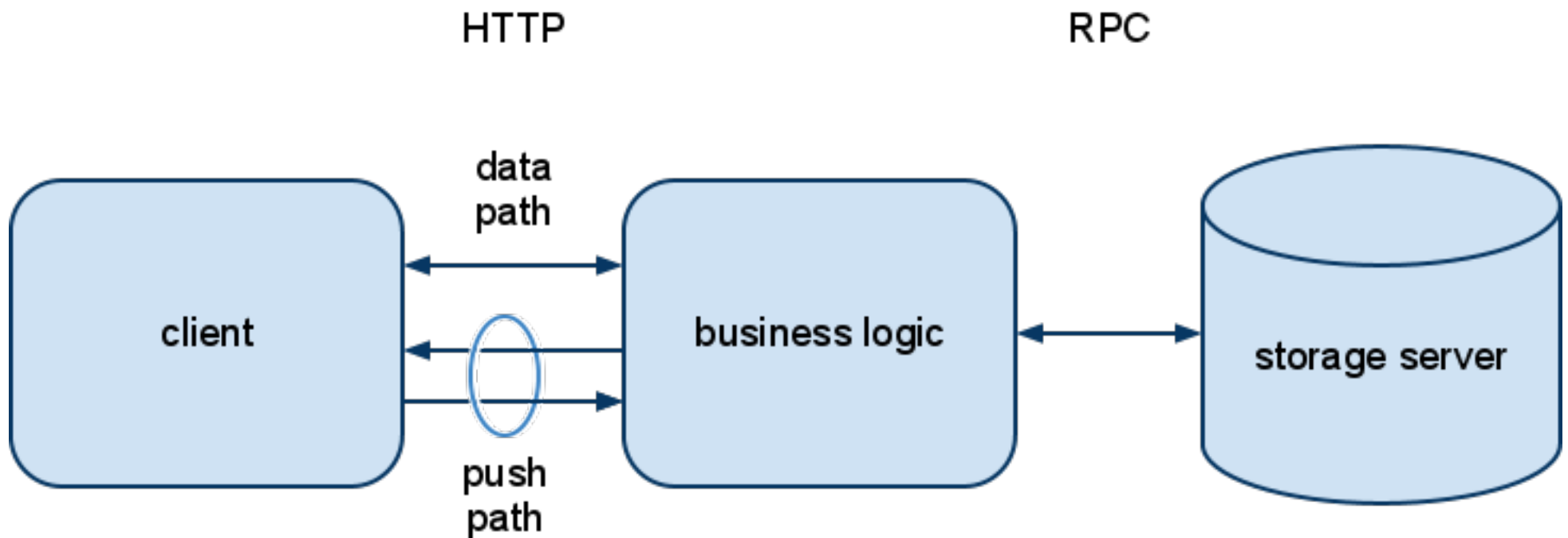


Where We Are

- More than just mail:
 - Google Buzz now in Gmail
 - Video and voice chat
 - SMS
 - Extensible through Google Apps Marketplace
- 60+ active labs
- 443k lines of JS (978k with comments)
- *Really complex application*



Macro-architecture



data path: code, styles, preferences, user data, ...

push path: chat, new mail, presence, contact updates, buzz updates, ...

Client

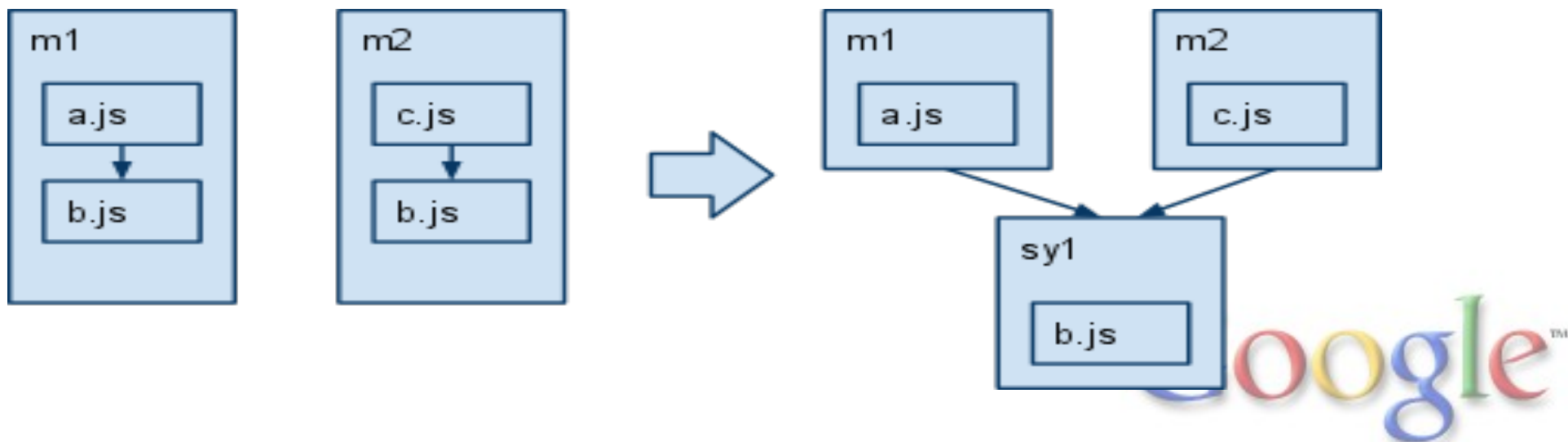
- Builds all UI
- Loads code when needed
- Fetches and caches data
- Records actions for performance analysis
- Reports presence / idle
- Gadget container
- Drives multiple windows

Server

- Routes / translates between client and 10+ backend servers
 - Talk, contacts, search, spell-check, translate, antivirus, SMTP in, SMTP out, authentication, ...
- Compiles and serves JS
- Compiles and serves themes as encoded stylesheet
- Incoming mail processing
- Synchronization for offline support

Client Details - Modules

- Modules based on entry points
 - Particular services (e.g. mole manager, chat)
 - Particular views (thread list, conversation, etc.)
- Non-entry-point code assigned to modules automatically
 - Files say what classes they require and provide
 - Classes needed multiple places => synthesized modules



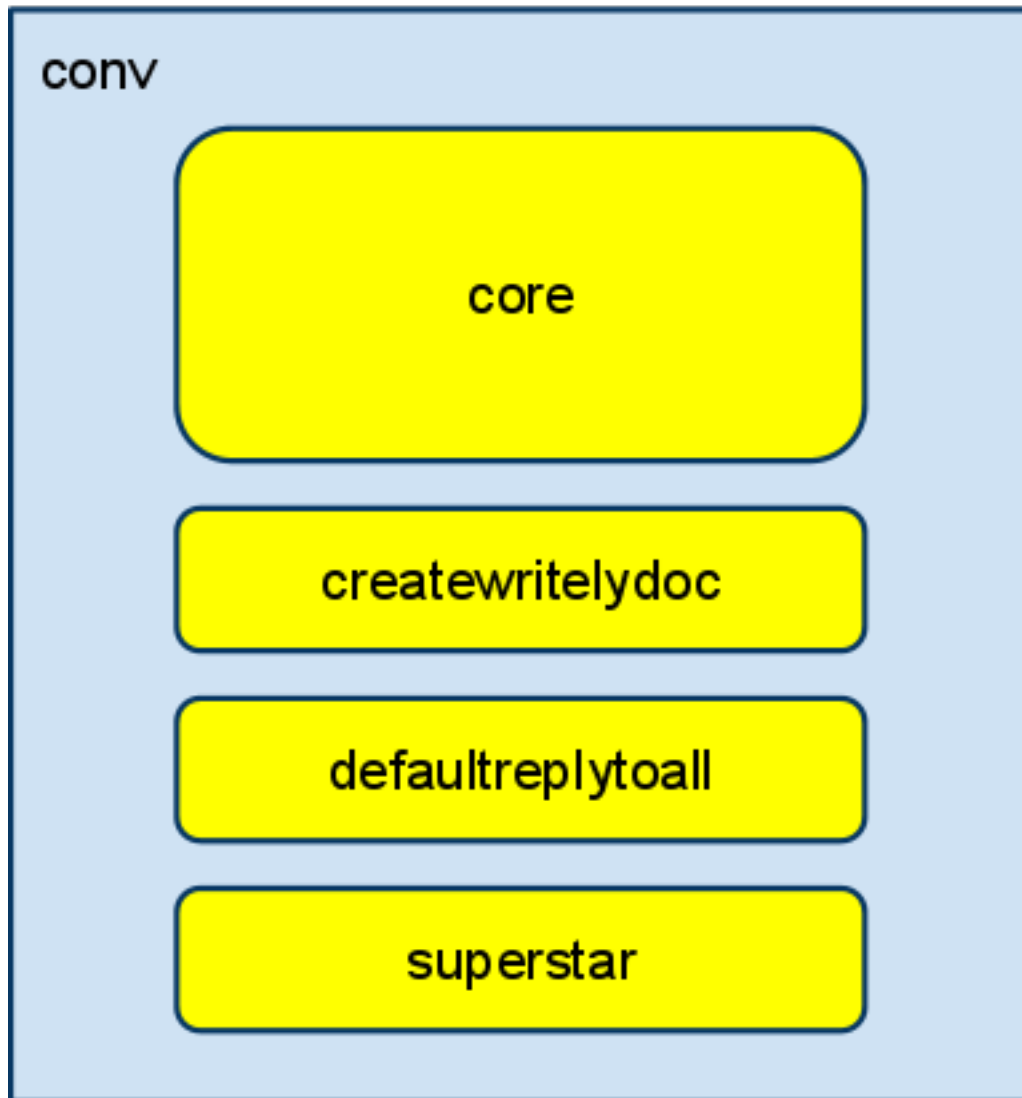
Client Details - Mods

- Mods enable tailored code w/o storage explosion
 - Mod = named code segments enabled per-user
 - Appended to module if enabled
 - Tweaks base code
- Whole app compiled / optimized, then fragmented
- Modules assembled from fragments based on enabled mods

Module/Mod Example

What's Possible

What's Served



Themes

- Mostly colors / images (can be radical)
- Macro processing of stylesheets
 - Everything skinnable is a macro
 - Last definition wins
- Start with basic color palette definitions
- Define attributes of all components in those terms
- Theme can tweak base palette or components
 - Theme definition is last file in the compilation

Services / Components

- Named services with defined interfaces
- Service objects registered in registry tree
 - Root registry for entire app
 - Child registry for each window
 - Failed lookup in child is repeated in parent
- Service object can be late-loaded
 - Callback when service defined, or error
- Components re-usable in alternate environments
- Replaceable for tests, alternate environments, alternate look&feel

Latency Tracking and Alerts

- All user actions timed, including server time
- Timing data uploaded to server and gathered
- Graphable along many axes
 - Country, Browser, Operation, Release, ...
 - Local, Server, Queue-delay, ...
 - Median, Mean, 25th pctl, 90th ptcl, ...
- Automated system predicts timing and count along many axes and alerts if the world is different

Test Automation

- Unit tests with system akin to JUnit
 - Compiled and uncompiled
- Some tests in Selenium
- In-application UI tests
 - Simpler for developers to write
 - Isolated in module and mod
 - Still can't generate real mouse events
- Use automation suite to gauge latency impact of a change or feature
- Multiple continuous builds test all aspects of client and server

What We've Learned

- Type-checking is important and possible
- Instrument everything
- Codify learnings in sanity tests & compiler warnings
 - `.manager-page .searchbar span{color:#000}`
 - `.CSS_IMG_DIV:hover .CSS_PLAY_DIV {opacity: 1;}`
- Testing is vital

Where We're Going

- HTML5
 - Change to leverage CSS3 reduced DOM by 30% and initial load time by 12%
 - Attachment / image drag-in
 - AppCache
 - Database
- Moving the platform forward
 - Dragging files out
 - Magic IFRAME
 - Installable apps with persistent background page

Drag Out

- Leverage drag-and-drop from HTML5
- Add new data transfer format: DownloadURL
 - String of form *mime-type:name:url*
- On drop, browser downloads file and streams it to drop target, marked as insecure

Magic IFRAME

- Targeted at apps with multiple windows
- All code and data go into an IFRAME
- If window hosting the IFRAME unloads, it gets adopted by another of the windows
- In Gmail for example:
 - Tearoff / pop-out compose creates bare window that is filled by code in IFRAME in main window
 - If you close the main window, the code looks for a tearoff that can accept the IFRAME and moves it
 - You finish your compose and can still send the email
- Old way: create new instance of Gmail tailored to the task.

Apps with Background Window

- User installs web app => greater trust
- App opens background page that is always loaded
- App defines domain extent that puts pages in same process
- Page loaded from web can find background window
- In Gmail:
 - Background page holds all code and data
 - Background code fills in DOM of foreground page
 - Background keeps data up-to-date
 - Really fast startup



A decorative header featuring four overlapping spheres: a green one on the left, and blue, red, and yellow ones on the right. A thin black horizontal line is positioned below the spheres.

Questions?

The Google logo, consisting of the word "Google" in its signature multi-colored font (blue, red, yellow, green, blue, red) with a trademark symbol (TM) to the right.

Google™