
Google™
09
I/O

Google's HTML5 Work: What's Next?

Matthew Papakipos
May 27, 2008

Post your questions for this talk on Google Moderator:
code.google.com/events/io/questions

Browsers Started a Revolution that Continues

- In 1995 Netscape introduced JavaScript
- In 1999, Microsoft introduces XMLHTTP
- In 2002, Mozilla 1.0 includes XMLHttpRequest natively

... Then web applications started taking off ...

- In 2004, Gmail launches as a beta
- In 2005, AJAX takes off (e.g. Google Maps)

... Now web applications are demanding more capabilities

What New Capabilities do Webapps Need?

- Plugins currently address some needs, others are still not well addressed
  - Playing video
  - Webcam / microphone access
  - Better file uploads
  - Geolocation
  - Offline abilities
  - 3D
  - Positional and multi-channel audio
  - Drag and drop of content and files into and out of webapps
- Some of these capabilities are working their way through standards process

Our Goal

- Empower web applications
  - If a native app can do it, why can't a webapp?
  - Can we build upon webapps strengths?
- Understand what new capabilities are needed
  - Talking to application developers (you!)
  - Figure out what native applications people run
    - And what web applications serve similar purposes
    - And what native applications have no web equivalent
- Implement (we're going full speed ahead...)
  - We prototyped in Gears
  - Now we're implementing natively in Google Chrome
- Standardize

<canvas>

- One of the first HTML5 additions to be implemented by browsers – in Safari, then Firefox and Opera. (We got it for free in Google Chrome from WebKit).
- Provides a surface on which you can draw 2D images
- See creative uses in the client pod outside
- Talk of extending the model for 3D (more later)

```
// canvas is a reference to a <canvas> element
var context = canvas.getContext('2d');
context.fillRect(0,0,50,50);
canvas.setAttribute('width', '300'); // clears the canvas
context.fillRect(0,100,50,50);
canvas.width = canvas.width; // clears the canvas
context.fillRect(100,0,50,50); // only this square remains

(reproduced from http://www.whatwg.org/specs/web-apps/current-
work/#canvas with permission)
```

<canvas> Demo

Local Storage

- Provides a way to store data client side
- Useful for many classes of applications, especially in conjunction with offline capabilities
- 2 main APIs provided: a database API (exposing a SQLite database) and a structured storage api (key/value pairs)
- Implementation under way in Google Chrome, already working in WebKit.

```
db.transaction(function(tx) {
tx.executeSql('SELECT * FROM MyTable', [],
function(tx, rs) {
for (var i = 0; i < rs.rows.length; ++i) {
var row = rs.rows.item(i);
DoSomething(row['column']);
}
});
});
```

Workers

- Native apps have threads and processes
- Workers provide web apps with a means for concurrency
- Can offload heavy computation onto a separate thread so your app doesn't block
- Come in 3 flavors
  - Dedicated (think: bound to a single tab)
  - Shared (shared among multiple windows in an origin)
  - Persistent (run when the browser is "closed")
- Implementation is ongoing in WebKit and Google Chrome

Application Cache

- Application cache solves the problem of how to make it such that one can load an application URL while offline and it just “works”
- Web pages can provide a “manifest” of files that should be cached locally
- These pages can be accessed offline
- Enables web pages to work without the user being connected to the Internet
- Implemented in WebKit, implementation ongoing in Google Chrome

<video>

- Allows a page to natively play video
  - No plugins required
  - As simple as including an image - <video src="video.mp4">
- Has built-in playback controls
  - Stop
  - Pause
  - Play
- Scriptable, in case you want your own dynamic control
- Chrome will support MP4 (H.264 + AAC), and Ogg (Theora + Vorbis)
- Implemented in WebKit, implementation under way in Google Chrome (now in dev channel).

<video> Demo

Rich Text Editing

- contentEditable implementations vary widely between browsers
- Behaviour for simple things, such as selection ranges, is not well defined in any spec
- Currently helping to define a specification for existing behaviour, as well as improvements for new functionality
  - Specify exactly what selections select
  - Add execCommand to additional events where it makes sense
  - getData('text/html') for paste and drop events
- No reason web apps should have to ship 200KB to do rich text editing

Notifications

- Alert() dialogs are annoying, modal, and not a great user experience
- Provide a way to do less intrusive event notifications
- Work regardless of what tab / window has focus
- Provide more flexibility than an alert() dialog
- Not currently in any standard, we're currently prototyping

Web Sockets

- Allows bi-directional communication between client and server in a cleaner, more efficient form than hanging gets (or a series of `XMLHttpRequest`s)
- Specification is under active development

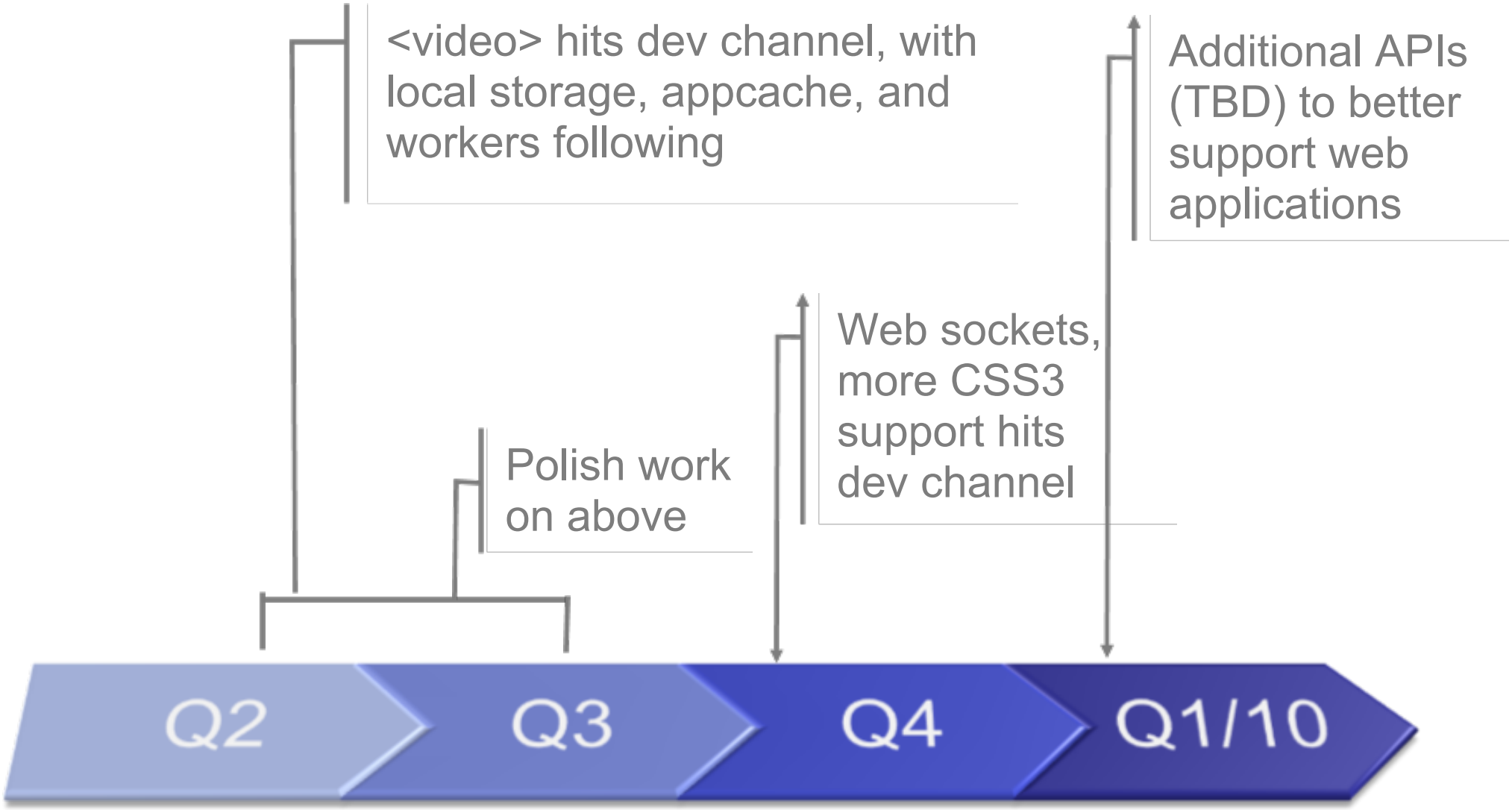
3D APIs

- Canvas 3D, developed by Mozilla, is a command-mode API that allows developers to make OpenGL calls via JavaScript
- O3D is an effort by Google to develop a retain-mode API where developers can build up a scene graph and manipulate via JavaScript, also hardware accelerated
- Discussion on the web and in standards bodies to follow

And So Much More…

- There's much work that we haven't even started yet.
- Some is well defined
  - Geolocation
  - Forms2
  - Datagrid
- Many things less defined
  - P2p APIs
  - Better drag + drop support
  - Webcam / Microphone access
  - O/S integration (protocol / file extension handlers and more)
  - Managing uploads, "blobs", file APIs
  - And more

Our Work in a Nutshell



Q & A

Post your questions for this talk on Google Moderator:
code.google.com/events/io/questions


Google™
09
I/O