Google
Developer
Day2009

Google's HTML5 Work: What's Next?

Patrick Chanezon
June 5 2009

Google Developer Day2009

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

“it is Google which is driving web standards forward. That is why we at Zoho are firmly aligned with them, even if they are our primary competitor. We believe in an open web, there is plenty of opportunity for all of us. Could Google abuse its position? Well, I am sure they understand karma.”

Microsoft Silverlight vs Google Wave: Why Karma Matters | Zoho Blogs

Sridhar Vembu, CEO of AdventNet (Zoho)

Web vector graphics today

99%

W3C WORLD WIDE WEB consortium

33%

20%

Microsoft® Silverlight™

33%

HTML 5

Draft

4.8.11 The canvas element

Google Developer Day2009

<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
- 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)
```

OpenWeb vector graphics available in a browser near you

- Firefox, Safari (+iPhone), Opera, Chrome
- ~33% browsers natively
- Open Source JS Shims for Canvas and SVG (autumn 2009) support in IE
  - Much of SVG 1.1 Full supported by all browsers except IE
  - Most of Canvas (except for text extensions) supported by all browsers except IE
- No Flash in iPhone & Android

http://a.deveria.com/caniuse/

SVG (basic support) - Recommendation

Method of displaying basic Vector Graphics features using the embed or object elements

Resources: Wikipedia Sample files

	Internet Explorer	Firefox	Safari	Chrome	Opera
Far Past	6.0	2.0	3.1	0.2	9.0
Past	7.0				
Present	8.0	3.0	3.2	1.0	9.6
Near Future (2009)		3.5	4.0	2.0	10.0
Future (2010 or later)	9.0	4.0	4.*		10.*

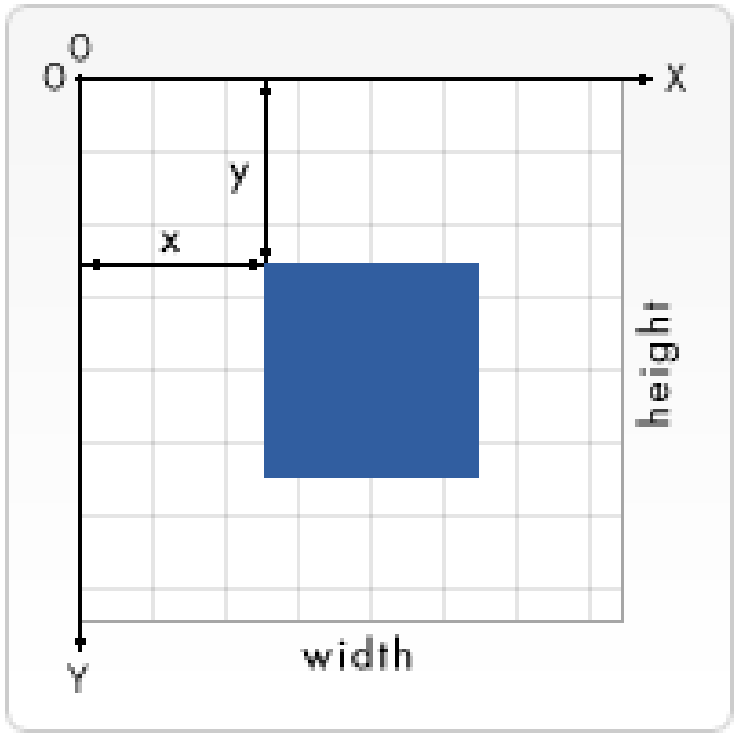
Canvas (basic support) - Working Draft

Method of generating dynamic graphics using JavaScript

Resources: Wikipedia Tutorial by Mozilla Animation kit experiment

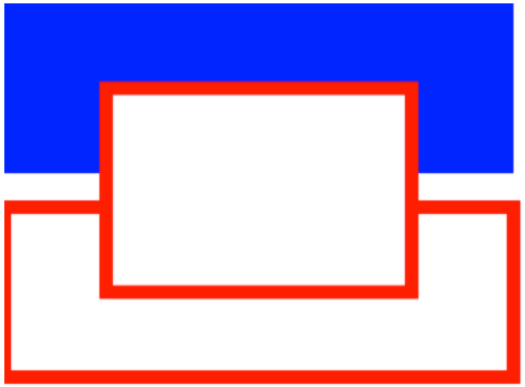
	Internet Explorer	Firefox	Safari	Chrome	Opera
Far Past	6.0	2.0	3.1	0.2	9.0
Past	7.0				
Present	8.0	3.0	3.2	1.0	9.6
Near Future (2009)		3.5	4.0	2.0	10.0
Future (2010 or later)	9.0	4.0	4.*		10.*

The Grid



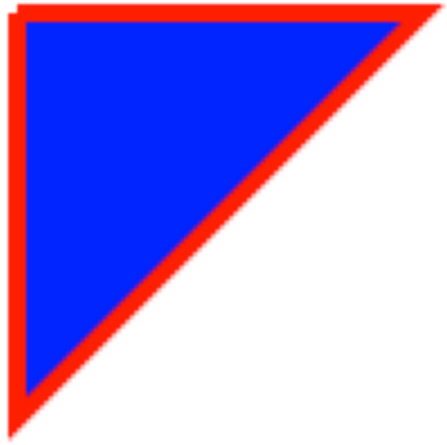
from https://developer.mozilla.org/en/Canvas_tutorial

fill, stroke, lines, Rect



```
context.fillStyle   = '#00f'; // blue
context.strokeStyle = '#f00'; // red
context.lineWidth   = 4;

// Draw some rectangles.
context.fillRect  (0,   0, 150, 50);
context.strokeRect(0,  60, 150, 50);
context.clearRect (30, 25,  90, 60);
context.strokeRect(30, 25,  90, 60);
```

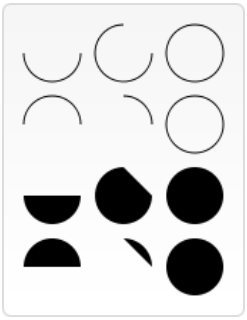
Path



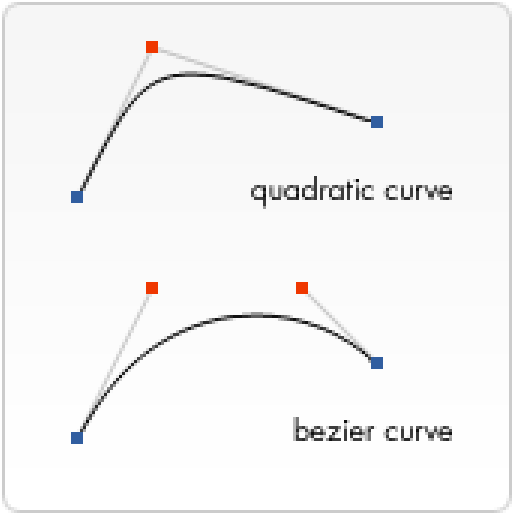
```
// Set the style properties.
context.fillStyle   = '#00f';
context.strokeStyle = '#f00';
context.lineWidth   = 4;
context.beginPath();
// Start from the top-left point.
context.moveTo(10, 10); // give the (x,y) coordinates
context.lineTo(100, 10);
context.lineTo(10, 100);
context.lineTo(10, 10);

// Done! Now fill the shape, and draw the stroke.
context.fill();
context.stroke();
context.closePath();
```

Arcs, Curves

```
arc(x, y, radius, startAngle, endAngle, anticlockwise)
```

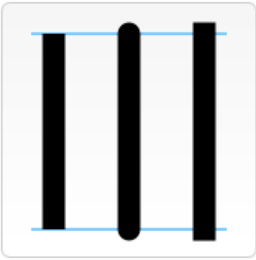


```
quadraticCurveTo(cp1x, cp1y, x, y) // (BROKEN in FF 3.5)
bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y)
```



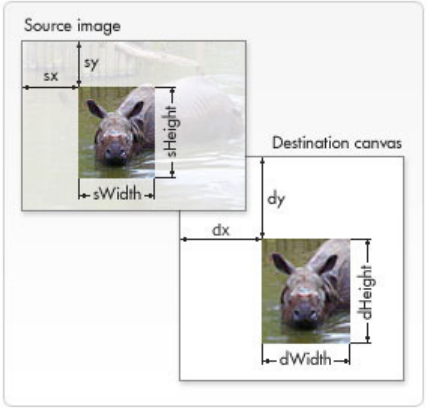
Styles and Colors

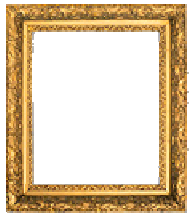
```
ctx.fillStyle = "orange";
ctx.fillStyle = "#FFA500";
ctx.fillStyle = "rgb(255,165,0)";
ctx.strokeStyle = "rgba(255,0,0,0.5)";

lineWidth = value
lineCap = type
lineJoin = type
miterLimit = value

createLinearGradient(x1,y1,x2,y2)
createRadialGradient(x1,y1,r1,x2,y2,r2)
addColorStop(position, color)

createPattern(image,type)
```



Images: draw, scale, slice

```
//drawImage(image, sx, sy, sWidth, sHeight, dx, dy,
dWidth, dHeight)// Draw slice
ctx.drawImage(document.getElementById('source'),
33,71,104,124,21,20,87,104);
```





```
// Draw frame
ctx.drawImage(document.getElementById('frame'),0,0);cont
ext.stroke();context.closePath();
```

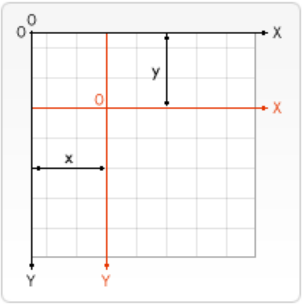
Pixel manipulation of Images, Video

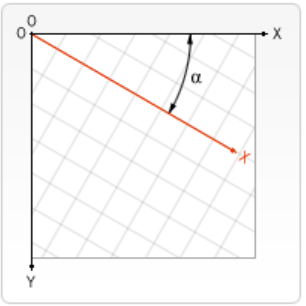
```
computeFrame: function() {
        this.ctx1.drawImage(this.video, 0, 0,
this.width, this.height);
        let frame = this.ctx1.getImageData(0, 0,
this.width, this.height);
        let l = frame.data.length / 4;

        for (let i = 0; i < l; i++) {
                let r = frame.data[i * 4 + 0];
                let g = frame.data[i * 4 + 1];
                let b = frame.data[i * 4 + 2];
                if (g > 100 && r > 100 && b < 43)
                        frame.data[i * 4 + 3] = 0;
        }
        this.ctx2.putImageData(frame, 0, 0);
        return;
}
```

https://developer.mozilla.org/en/manipulating_video_using_canvas

Tranformations

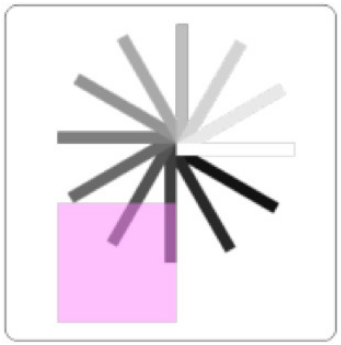
```
State: Canvas states are stored on a stack
save()restore()
translate(x, y)

rotate(angle)

scale(x, y)

transform(m11, m12, m21, m22, dx, dy)
setTransform(m11, m12, m21, m22, dx, dy)
```



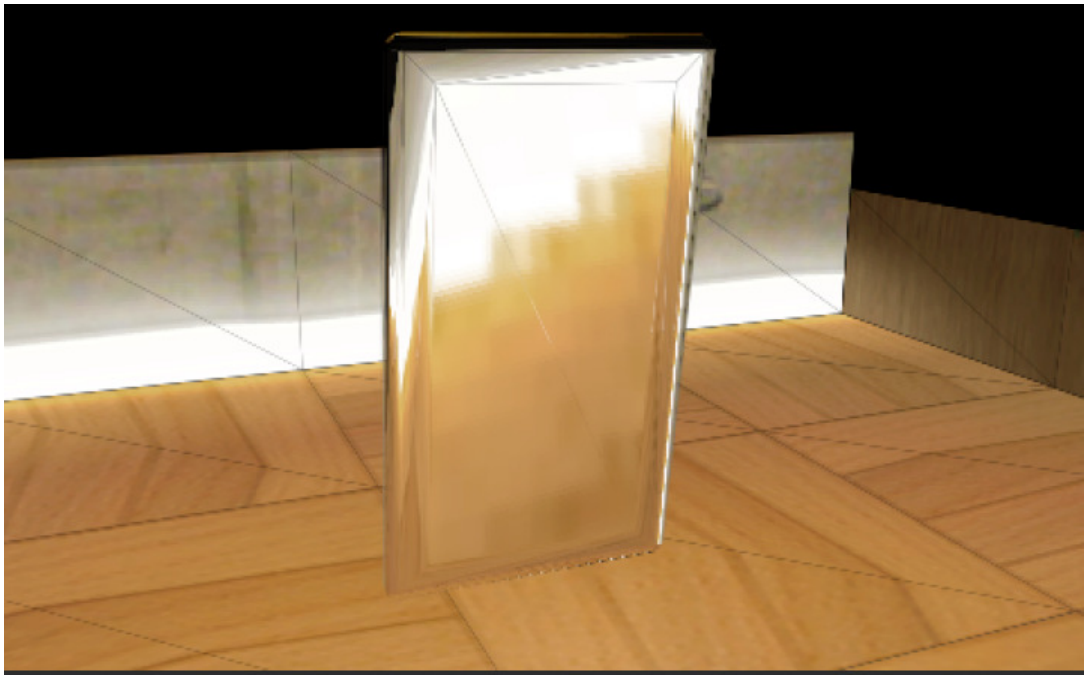


transform example

```
var i = 0;
var sin = Math.sin(Math.PI/6);
var cos = Math.cos(Math.PI/6);
ctx.translate(200, 200);
var c = 0;
for (i; i <= 12; i++) {
  c = Math.floor(255 / 12 * i);
  ctx.fillStyle = "rgb(" + c + "," + c + "," + c + ")";
  ctx.fillRect(0, 0, 100, 10);
  ctx.transform(cos, sin, -sin, cos, 0, 0);
}

ctx.setTransform(-1, 0, 0, 1, 200, 200);
ctx.fillStyle = "rgba(255, 128, 255, 0.5)";
ctx.fillRect(0, 50, 100, 100);
```



2D -> Triangles -> 3D



http://kawanet.blogspot.com/2009/02/incredible-javascriptcanvas-3d-demos.html

<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

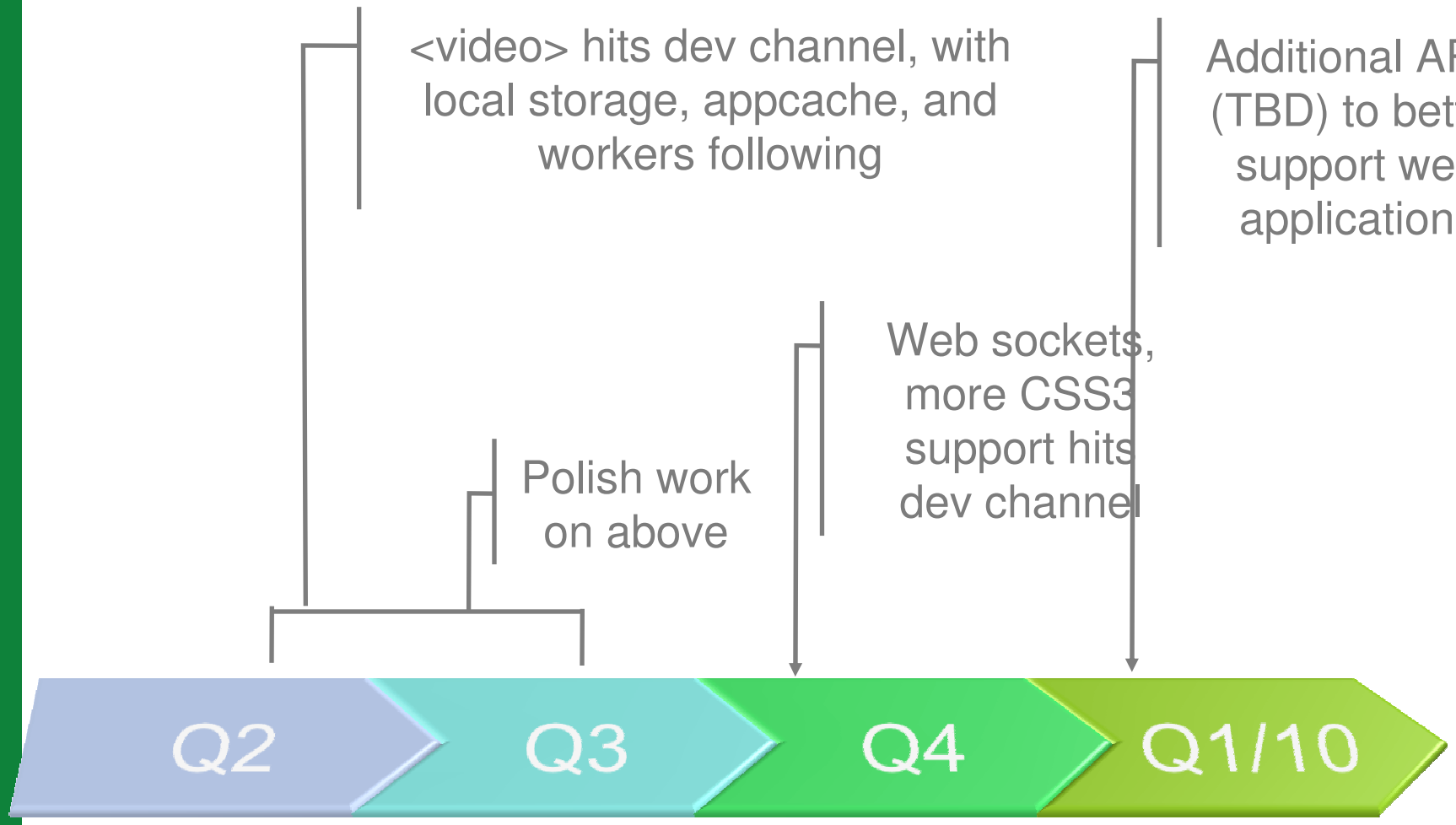
- Canvas 3D, developed by Mozilla, is a command-mode API that allows developers to make OpenGL calls via JavaScript
- O3D is an effort by Google to develop a retain-mode API where developers can build up a scene graph and manipulate via JavaScript, also hardware accelerated
- Discussion on the web and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

3D Demo

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



Resources

- Spec http://whatwg.org/html5
- http://code.google.com/chromium/
- http://code.google.com/apis/o3d/
- All demos at http://delicious.com/chanezon/openweb+demos
- https://developer.mozilla.org/en/Canvas_tutorial
- http://dev.opera.com/articles/view/html-5-canvas-the-basics/
- http://blog.mozbox.org/tag/demo
- https://bespin.mozilla.com/

Learn More at
http://code.google.com/chromium/

Q&A

Google
Developer
Day2009

Google
Developer
Day2009