WebRTC hits Firefox, Android and iOS

By Sam Dutton
Sam is a Developer Advocate

A **lot** has happened with WebRTC over the last few weeks. Time for an update!

In particular, we're really excited to see WebRTC arriving on multiple browsers and platforms.

`getUserMedia` is available now in Chrome with no flags, as well as Opera, and Firefox Nightly/Aurora (though for Firefox you'll need to set preferences). Take a look at the cross-browser demo of `getUserMedia` at simpl.info/gum—and check out Chris Wilson's amazing examples of using `getUserMedia` as input for Web Audio.

`webkitRTCPeerConnection` is now in Chrome stable and it's flagless. TURN server support is available in Chrome 24 and above. There's an ultra-simple demo of Chrome's RTCPeerConnection implementation at simpl.info/pc and a great video chat application at apprtc.appspot.com. (A word of explanation about the name: after several iterations, it's currently known as `webkitRTCPeerConnection`. Other names and implementations have been deprecated. When the standards process has stabilised, the `webkit` prefix will be removed.)

WebRTC has also now been implemented for desktop in Firefox Nightly and Aurora, and for iOS and Android via the Ericsson Bowser browser.

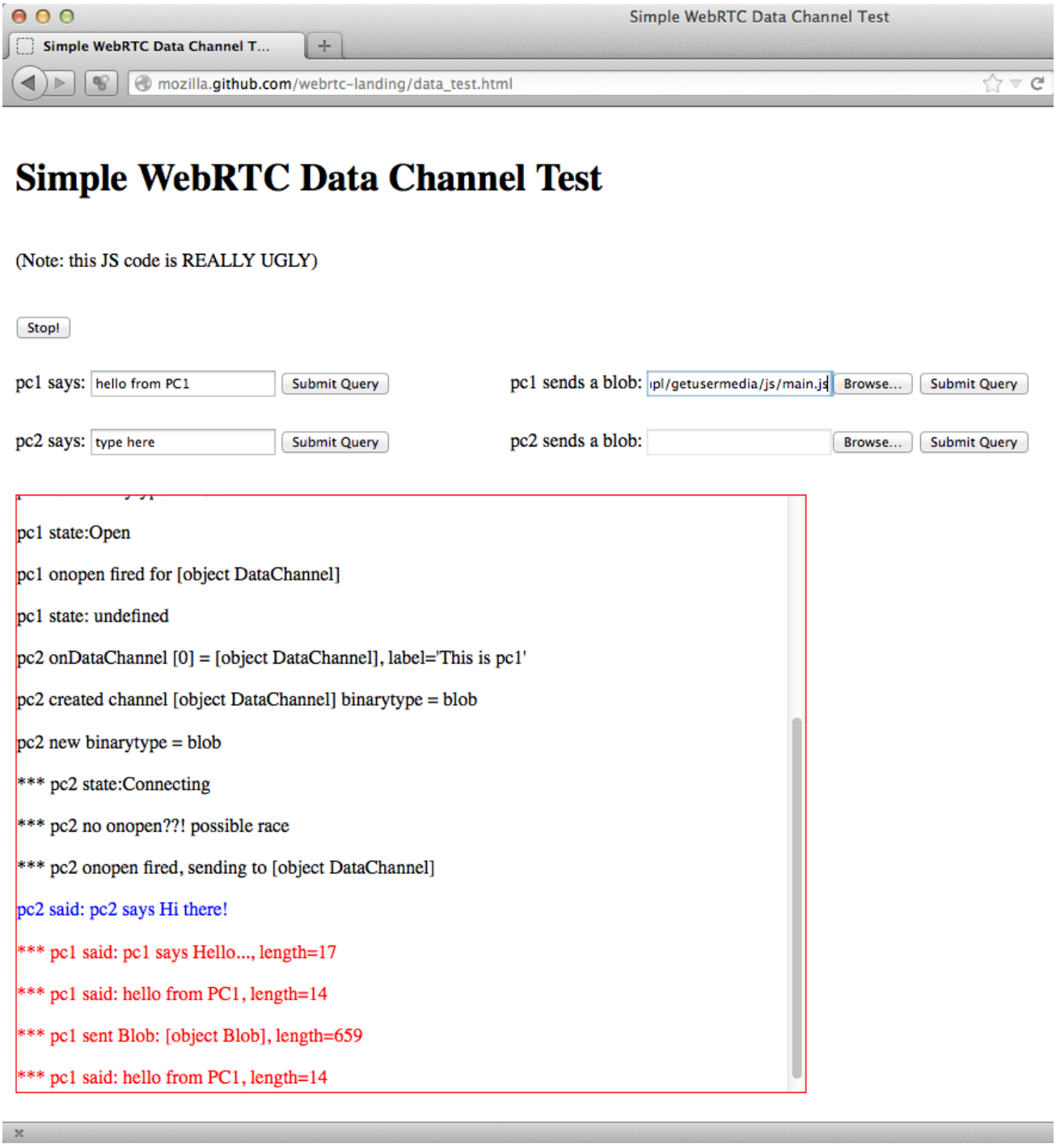
DataChannel

DataChannel is a WebRTC API for high performance, low latency, peer-to-peer communication of arbritary data. The API is simple—similar to WebSocket—but communication occurs directly between browsers, so DataChannel can be much faster than WebSocket even if a relay (TURN) server is required (when 'hole punching' to cope with firewalls and NATs fails).

DataChannel is planned for version 25 of Chrome, behind a flag – though it may miss this version. This will be for experimentation only, may not be fully functional, and communication won't be possible with the Firefox implementation. DataChannel in later versions should be more stable and will be implemented so as to enable interaction with DataChannel in Firefox.

Firefox Nightly/Aurora supports `mozGetUserMedia`, `mozRTCPeerConnection` and `DataChannel` (but don't forget to set your about:config preferences!)

Here's a screenshot of DataChannel running in Firefox:



This demo is at http://mozilla.github.com/webrtc-landing/data_test.html. Here's a code snippet:

```
pc1.onconnection = function() {
  log("pc1 onConnection ");
  dc1 = pc1.createDataChannel("This is pc1",{}); // reliable (TCP-like)
```

```
dc1 = pc1.createDataChannel("This is pc1",{outOfOrderAllowed: true, maxRetransm
log("pc1 created channel " + dc1 + " binarytype = " + dc1.binaryType);
channel = dc1;
channel.binaryType = "blob";
log("pc1 new binarytype = " + dc1.binaryType);

// Since we create the datachannel, don't wait for onDataChannel!
channel.onmessage = function(evt) {
  if (evt.data instanceof Blob) {
    fancy_log("*** pc2 sent Blob: " + evt.data + ", length=" + evt.data.size,"b
  } else {
    fancy_log('pc2 said: ' + evt.data, "blue");
  }
}
channel.onopen = function() {
  log("pc1 onopen fired for " + channel);
  channel.send("pc1 says Hello...");
  log("pc1 state: " + channel.state);
}
channel.onclose = function() {
  log("pc1 onclose fired");
};
log("pc1 state:" + channel.readyState);
    }
```

More information and demos for the Firefox implementation are available from the hacks.mozilla.org blog. Basic WebRTC support is due for release in Firefox 18 at the beginning of 2013, and support is planned for additional features including **`getUserMedia`** and createOffer/Answer constraints, as well as TURN (to allow communication between browsers behind firewalls).

For more information about WebRTC, see Getting Started With WebRTC. There's even a WebRTC book, available in print and several eBook formats.

Resolution Constraints

Constraints have been implemented in Chrome 24 and above. These can be used to set values for video resolution for **`getUserMedia()`** and RTCPeerConnection **`addStream()`** calls.

There's an example at simpl.info/getusermedia/constraints. Play around with different constraints by setting a breakpoint and tweaking values.

A couple of gotchas... **`getUserMedia`** constraints set in one browser tab affect constraints for all tabs opened subsequently. Setting a disallowed value for constraints gives a rather

cryptic error message:
navigator.getUserMedia error: NavigatorUserMediaError {code: 1, PERMISSION_DENIED: 1}

Likewise the error if you try to use **`getUserMedia`** from the local file system, not on a server!

Streaming screen capture

Tab Capture is now available in the Chrome Dev channel. This makes it possible to capture the visible area of the tab as a stream, which can then be used locally, or with RTCPeerConnection's **`addStream()`**. Very useful for sceencasting and web page sharing. For more information see the WebRTC Tab Content Capture proposal.

Keep us posted by commenting on this update: we'd love to hear what you're doing with these APIs.

...and don't forget to file any bugs you encounter at new.crbug.com!

Last updated July 2, 2018.