Realtime Web with Gevent and Server-Sent Events

@stevenewey

Why not WebSockets?

Why not WebSockets?

- hixie-75, hixie-76, hybi-00, hybi-07, hybi-10

Why not WebSockets?

- hixie-75, hixie-76, hybi-00, hybi-07, hybi-10
- Takes my lovely HTTP connection and does unspeakable things to it.

Why not WebSockets?

- hixie-75, hixie-76, hybi-00, hybi-07, hybi-10
- Takes my lovely HTTP connection and does unspeakable things to it.
- Hard to put behind nginx.

Why not WebSockets?

- hixie-75, hixie-76, hybi-00, hybi-07, hybi-10
- Takes my lovely HTTP connection and does unspeakable things to it.
- Hard to put behind nginx.
- Also: proxies, older browsers.

http://www.flickr.com/photos/binusarina/3889528397/

What now?

Let's think about what we actually need.

http://www.flickr.com/photos/ecristescu/3785682290/

Turns out...

Turns out...

- The conversation I want to have is pretty one sided. Because...

Turns out...

- The conversation I want to have is pretty one sided. Because...
- Any time my client has information for the server, it makes an XHR request. That works pretty well.

Turns out...

- The conversation I want to have is pretty one sided. Because...
- Any time my client has information for the server, it makes an XHR request. That works pretty well.
- So all I really need is for the server to be able to push events to the client.

Server-Sent Events

- The Vikings were the first people to discover SSE.
- First implemented in Opera way back in 2006.

http://www.flickr.com/photos/24736216@N07/7181127193/

Server-Sent Events

Over the wire: It's just a GET request.

```
GET /my_event_source HTTP/1.1
```

```
data: exciting new information\n\n
(some time later...)
data: Good news everyone!\n\n
```

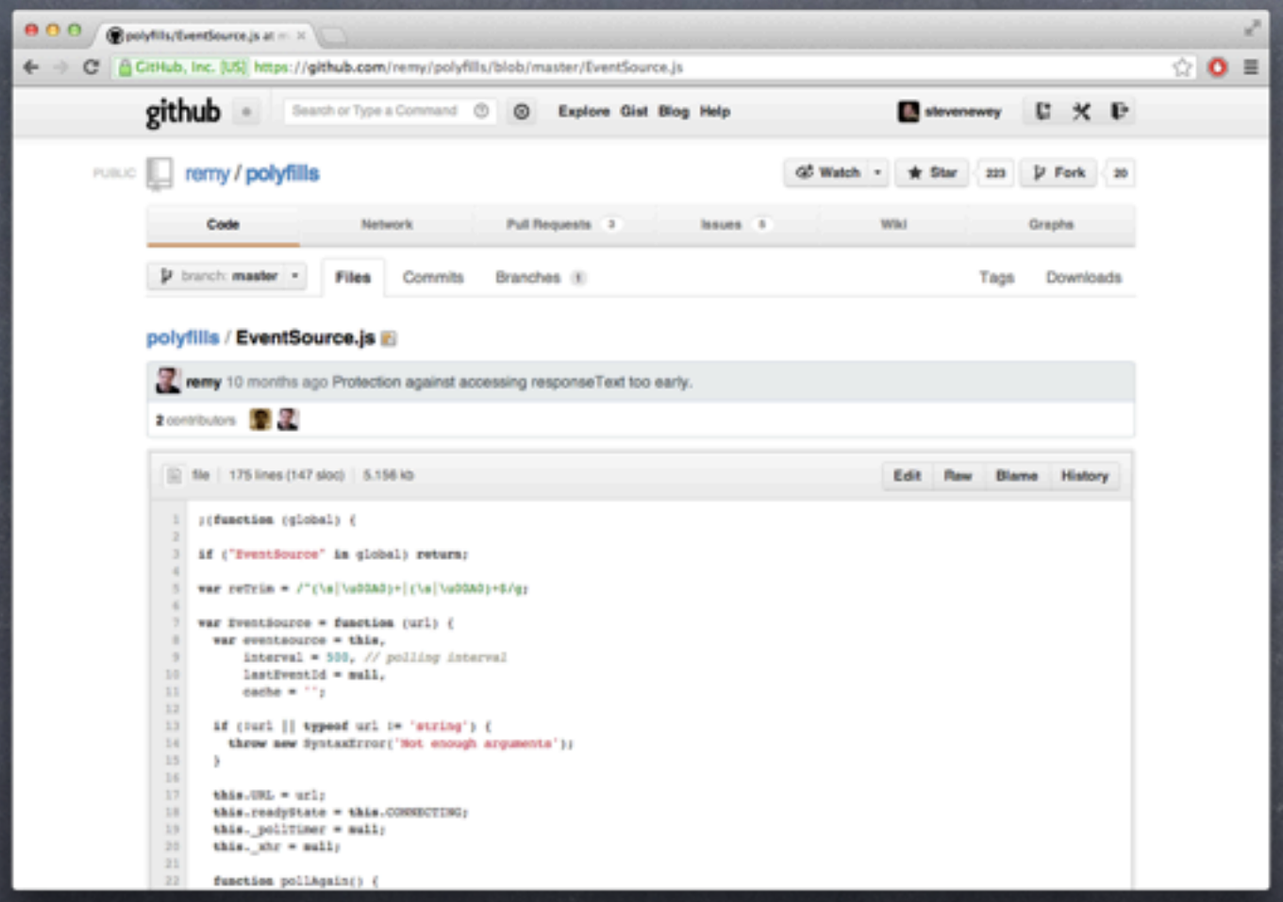
Server-Sent Events

In the browser:

```
var SSE = new EventSource(URL);
SSE.onmessage = function(message) {
    console.log('OMG, a message!');
    $.shoveItInTheDOM(message.data);
}
```

Server-Sent Events

When you need to support IE...



5k (unminified, uncompressed) Polyfill

https://github.com/remy/polyfills/blob/master/EventSource.js

The Server

```
from gevent.pywsgi import WSGIServer

monkey.patch all the things
```

Serve a simple Flask app

```
def event_stream():
    count = 0
    while True:
        gevent.sleep(2)
        yield 'data: %s\n\n' % count
        count += 1
```

```
@app.route('/my_event_source'):
def sse_request():
    return Response(
        event_stream(),
        mimetype='text/event-stream')
```

DEMO

http://www.flickr.com/photos/dieselbug2007/369649914/

Gotchas

Gotchas

- The spec says EventSource should respect CORS. In practice, it doesn't today, so it's handy that we can stick nginx in front of our server.
- Speaking of nginx, be sure to set proxy_buffering to off and set a healthy proxy_read_timeout or implement some kind of regular ping.

With 25 more minutes...

- Using Redis PubSub as a source for events.
- Building a client app that can change it's PubSub channels without having to reconnect the EventSource.
- How to combine this with a RESTful API and a ~~Javascript~~ CoffeeScript M*C framework for epic win.

Thanks

- Twitter/App.Net: @stevenewey
- s@s-n.me
- https://github.com/stevenewey/ssedemo