WebSockets

A quick tour and demo

Gaurav Oberoi

Dec 18, 2010 - Chennai Geeks

Who am I?

- **Entrepreneur**: started and sold 2 companies
  - BillMonk (consumer)
  - Precision Polling (SaaS)
- **Programmer, Product Manager**:
  - Amazon, Obopay, Xmarks, SurveyMonkey
- **General geekery**:
  - Seattle Tech Startups founder (~2200 member list)
  - TechStars mentor, advisor to startups

What is this talk about?

- **What** are WebSockets?
- **Why** do we need them?
- **How** do we use them?
- What **limitations** do they have?

Disclaimer

Haven’t deployed production code using WebSockets.

Not an expert!

What are WebSockets?

+

= ?

According to the textbook:

WebSocket is a technology for providing bi-directional full duplex communication channels over a single TCP socket.

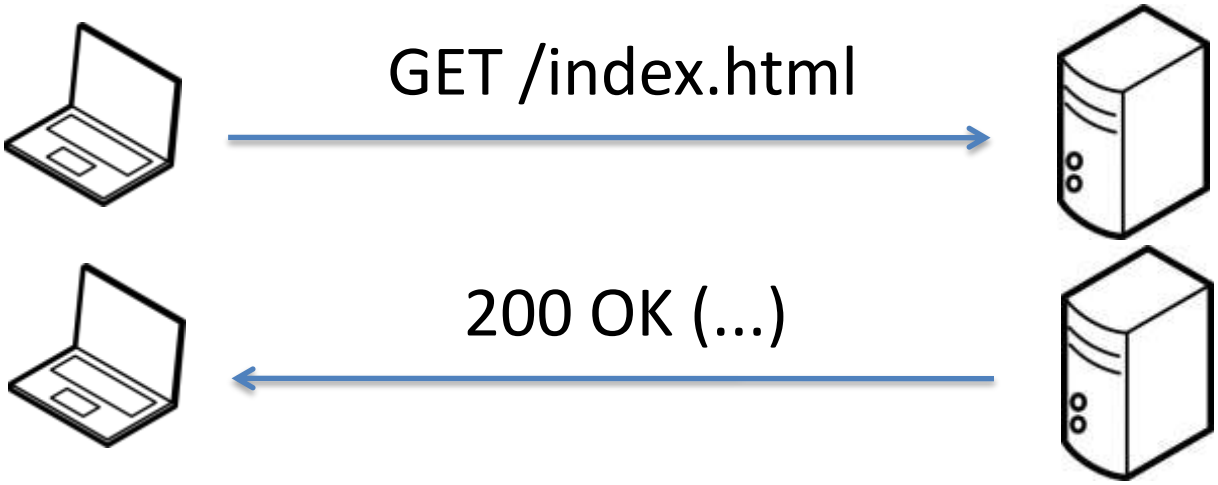
Ahem…

Put simply, WebSockets:

Are a new feature in HTML5 (draft) that let you stream data to and from web browsers.

For comparison:

Here's good ol' HTTP request and response:



With sockets, send data back and forth anytime:

More juicy details

- New protocol (ws:// and wss://)
  - W3C and IETF standard
  - Currently a draft, more on this later
- Uses ports 80 and 443
  - 443 recommended for proxy traversal
- Starts as HTTP, then switches to WS
- Supports >1 connections over 1 TCP socket

More juicy details: the handshake

Client sends:

```
GET /demo HTTP/1.1
Host: example.com
Connection: Upgrade
Sec-WebSocket-Key2: 12998 5 Y3 1  .P00
Sec-WebSocket-Protocol: sample
Upgrade: WebSocket
Sec-WebSocket-Key1: 4 @1  46546xW%0l 1 5
Origin: http://example.com

^n:ds[4U
```

Server responds:

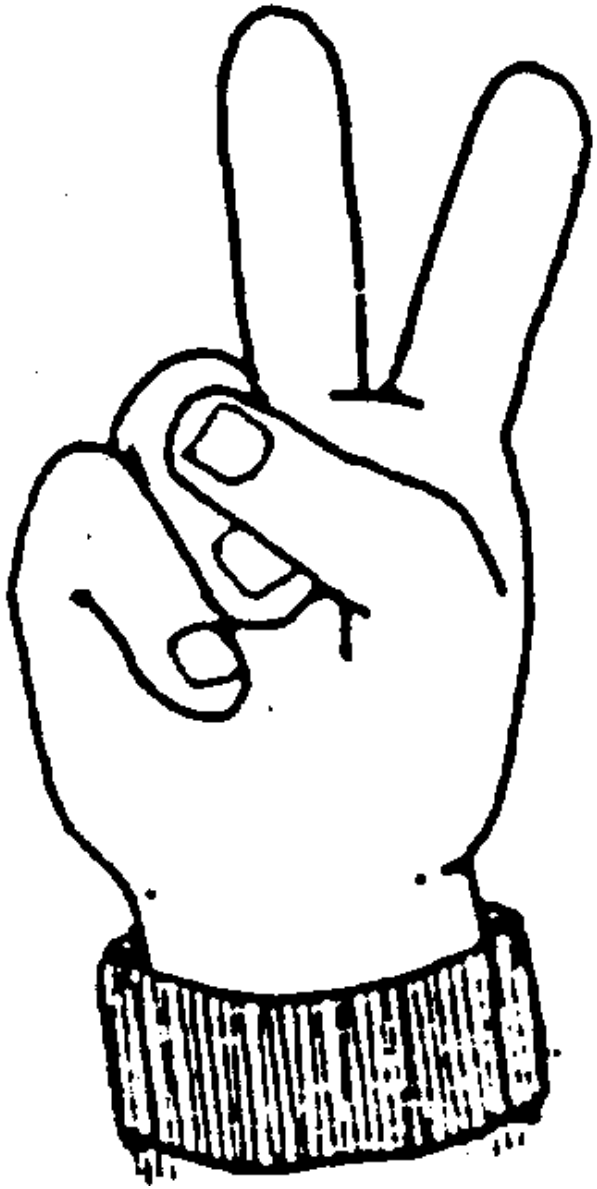
```
HTTP/1.1 101 WebSocket Protocol Handshake
Upgrade: WebSocket
Connection: Upgrade
Sec-WebSocket-Origin: http://example.com
Sec-WebSocket-Location: ws://example.com/demo
Sec-WebSocket-Protocol: sample

8jKS'y:G*Co,Wxa-
```

Yawn…

… so why do we need WebSockets?

2 good reasons



First: desire for real-time

Want low latency 2-way communication for:

- Gaming (pong)
- Collaboration (live wikis)
- Dashboards (financial apps)
- Tracking (watch user actions)
- Presence (chat with customer support)
- More!

Second: HTTP doesn't deliver

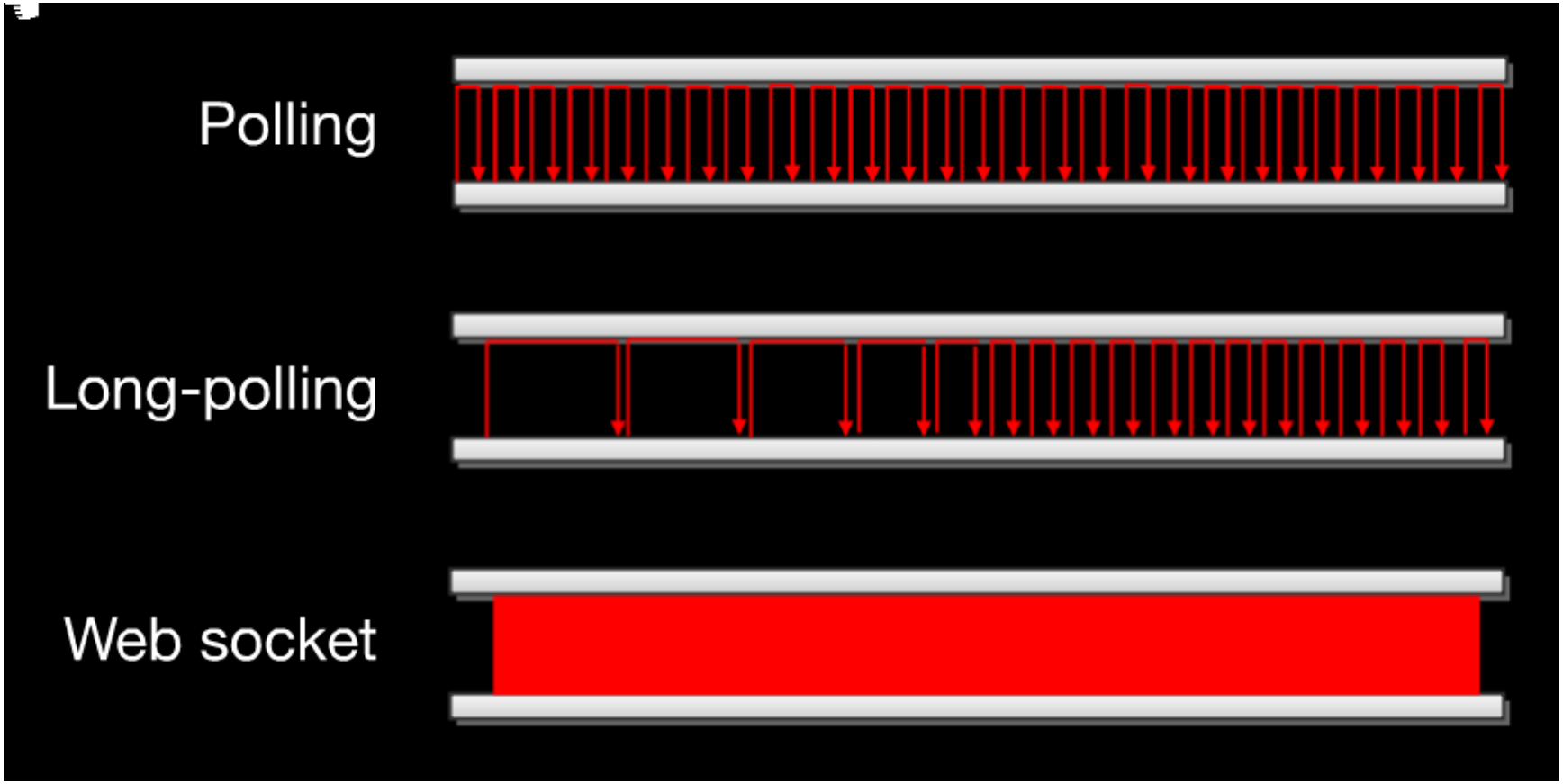
People hack around this (see "Comet")

- Polling, long-polling, stream via hidden iframe
- BUT these are slow, complex, and bulky

Or rely on plugins:

- Flash, Silverlight, Java applets
- BUT these don't work everywhere (phones)

Damn, this is hairy:



Source: http://www.slideshare.net/ismasan/websockets-and-ruby-eventmachine

Vs. HTTP hacks, Websockets provide:

- **Lower latency**: no new TCP connections for each HTTP request
- **Lower overhead:** for each message sent (2 bytes vs. lines of HTTP header junk)
- **Less traffic**: since clients don't need to poll, messages only sent when we have data

How do we use WebSockets?

What you'll need:

1. WebSockets compliant browser
   - WebKit: Chrome, Safari (works on iOS)

2. Client Javascript API

3. Server-side API

Client API in Javascript

Initialize:

```
varws = new WebSocket(“ws://example.com”)
```

Send data:

```
send(data)
```

Event-handlers:

```
onopen
onmessage
onclose
onerror
```

Server-side API

Threaded servers can be used but not recommended:

- 1 thread/process per socket → memory hog!

Evented-servers are ideal:

- 1 process handles all requests
- Waits for I/O before processing requests (reactor pattern)

Suggested servers:

- Ruby: Event Machine + em-websocket
- Javascript: Node.js + WebSocket module
- Python: Twisted + txWebSocket

DEMO TIME!

What limitations do WebSockets have?

First, the big ones

- **Not all browsers** support them
  - No support in Firefox 4, IE9, Opera
  - So you need to fallback to something that works
  - See Socket.IO: client and server library for graceful fallback

- **Draft spec in jeopardy** after Dec 8 security announcement by Mozilla
  - Security vulnerabilities with transparent proxies
  - Not caused by WebSockets, but they expose issue

My guess: we'll get by this with some spec changes

And the other things

WebSockets need maintenance and care:

- **Re-open conn**if network hiccup or timeout
- **Back off** if server is down, don’t keep trying
- **Keep alive** if your connection times out
- **Buffer and re-send** msgs in the above cases

My guess: libraries will fill in these gaps

In Conclusion

- WebSockets == “TCP for the Web”
- Easy JS library, evented-servers on the back
- Not prod ready, but soon (fingers crossed)

Check out some demos:

http://websockets.org/demos.html

twitter.com/goberoi