HTML5 WebSocket

Building ***real-time*** Applications

Why WebSockets?

Building web applications with…

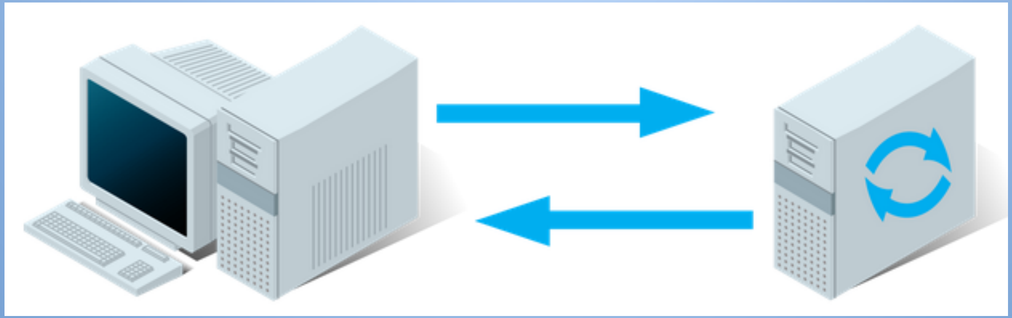
- → real-time component
- → low-latency experience
- → reduced network overhead

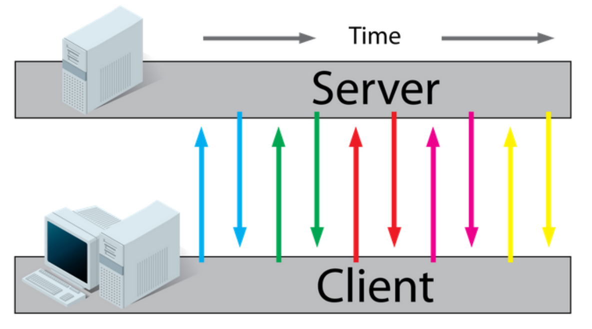
Use Cases

- → Browser-based multiplayer games (mmorpgs, FPS)
- → Real-time information (stock tickers, sports play-by-play)
- → Crowd-sourced applications (waze)
- → Social media
- → Second-screen applications

Real-time…HTTP?

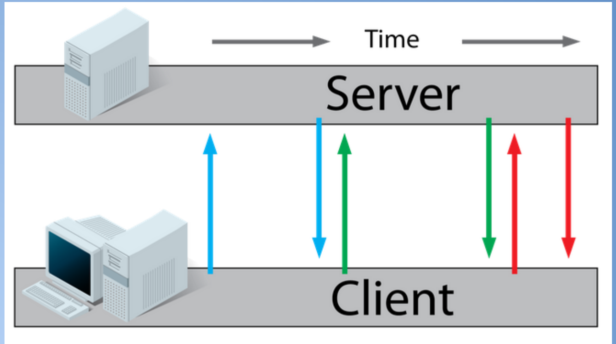
- Short-lived request/response interaction
- Traffic flows in one direction at a time
- Large amount of metadata generated per request





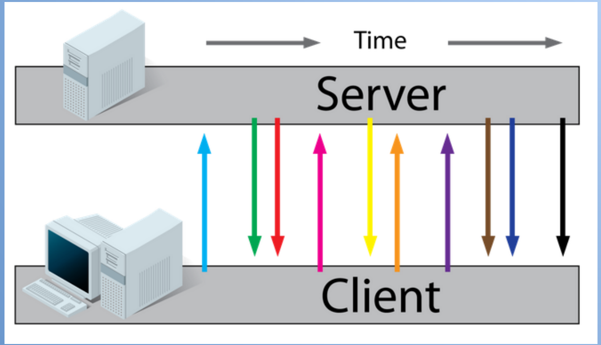
AJAX Polling:

- Unpredictable real-time data
- Insufficient (or unnecessary) # of requests



AJAX Long-polling:

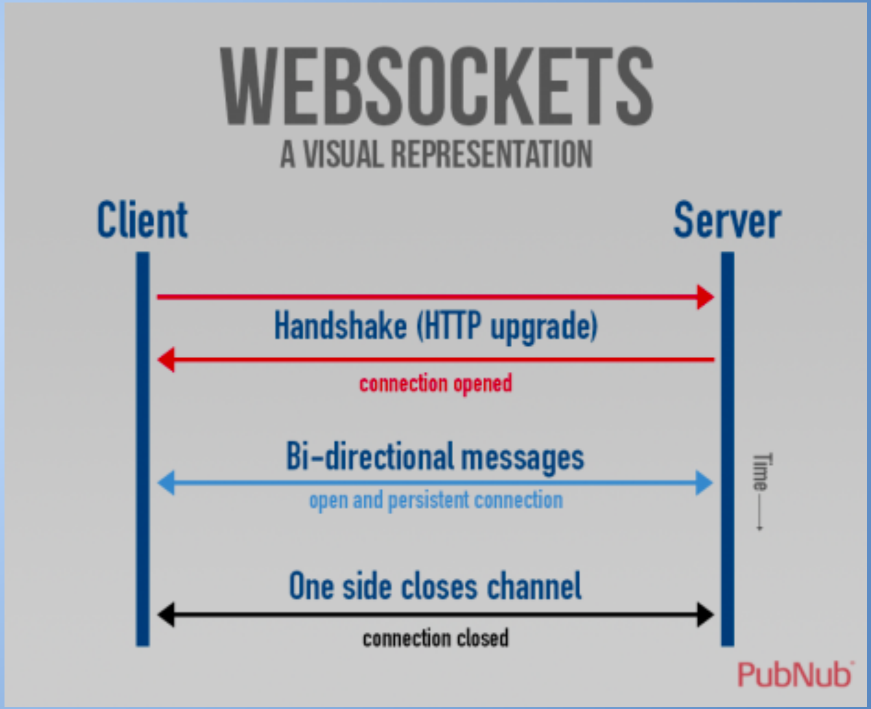
- Constant reconnections
- Timeouts if client-request is open too long.
- Requires a lot of server resources



WebSocket

...vs WebSocket

- → Persistent, long-running connection
- → Bi-directional communication
- → Full-duplex- traffic flows freely between
- → Scalable- low network overhead

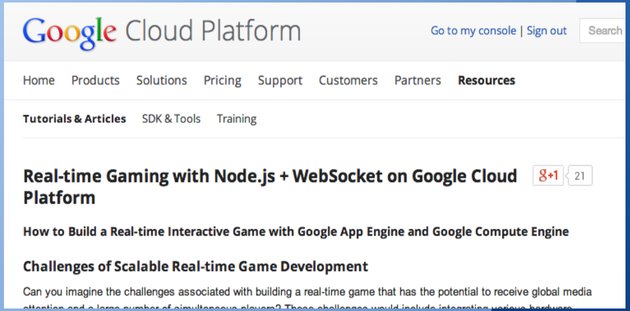


I’m convinced. Show me how it works.

Moving forward

- ➢ Web's standard for real-time communication
- ➢ Compatibility: Cross-browser fallback libraries
- ➢ Support: latest versions of all major browsers

Resources:
- Socket.io (JS library that leverages Node.js)
- MDN - Websockets
- The Definitive Guide to WebSockets (book)



Real-time Gaming with Node.js + WebSocket on Google Cloud Platform

How to Build a Real-time Interactive Game with Google App Engine and Google Compute Engine

Challenges of Scalable Real-time Game Development

Can you imagine the challenges associated with building a real-time game that has the potential to receive global media attention and a large number of simultaneous players? These challenges would include integrating various hardware