

Backend & Frontend architecture scalability & websockets

09-03-2016
Searchlite HQ

Anne Jan Brouwer
NoProtocol

About me

- Anne Jan Brouwer
- Professional developer since 2001
 - C++
 - Sysadminning
 - PHP 2004
 - NodeJS 2010
- NoProtocol
 - Realization
 - Full stack implementation



Backend & Frontend architecture scalability & websockets

- About this talk:
 - Language agnostic
 - Mostly . .
 - Architecture focussed
- For anyone involved with web
 - Sites
 - Apps
- Fun

Kinds of applications

- Content driven websites
- Realtime data driven
 - Physical data
 - News
 - Chat
- ??
- Most common are multi-user CMS-like systems
 - Right??

Backend & Frontend

- Back in the olden days
 - Didn't used to be like that
 - Everything was done serverside
 - Frontend was just styling
- These days
 - Code running serverside gets smaller
 - Code running clientside gets larger
 - Everything is (ran through) an API

Scalability

- What
 - More iron
 - Distribution (CDNs etc)
- How?
 - Handling state?
 - Getting rid of state?
 - Different models?
- Performance

Performance



Scalability



High availability

- What are we talking about?
 - Failovers?
 - Handovers?
- How do we achieve that?
 - Handling application state?
 - Getting rid of state?
 - Different models / patterns / paradigms?

Websockets

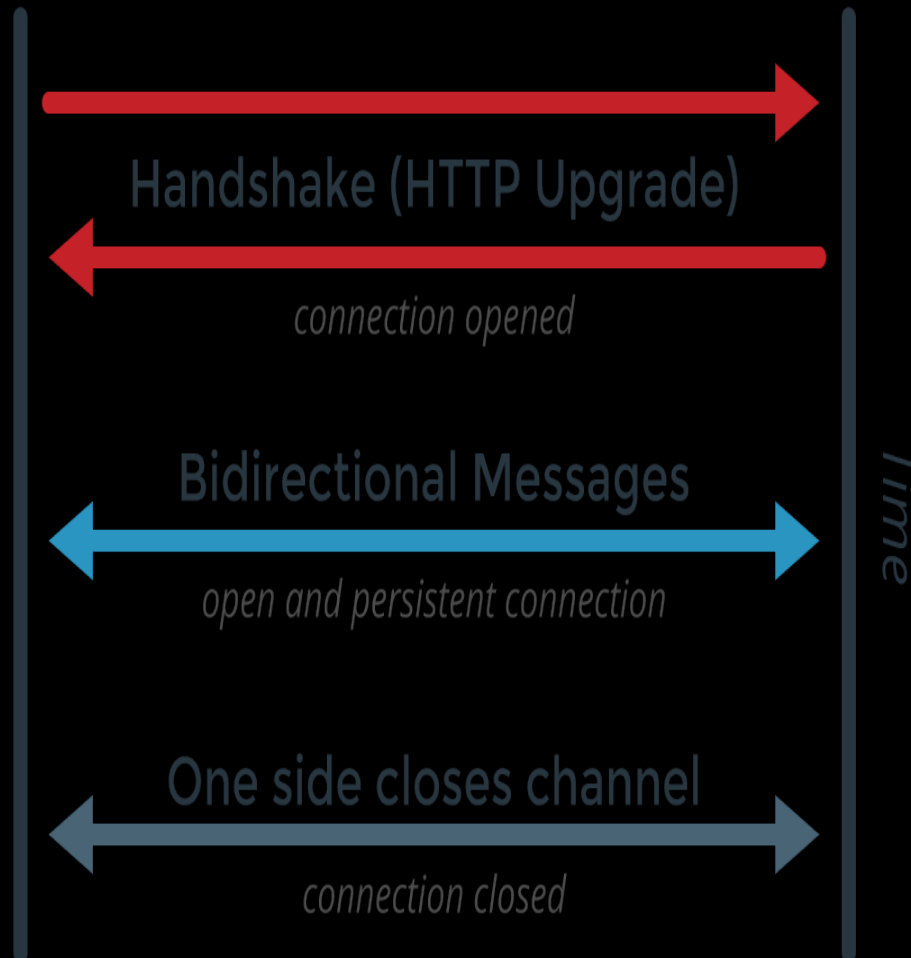
- Back in the olden days
 - Polling
 - Long Polling
- Standard since 2011
 - RFC6455
- Implementations available for most programming languages
 - Yes, even for PHP



Websockets

Client

Server



Websockets

- They're nice for things like chat and games
- But what about serious applications?
 - Etherpad
 - Ethercalc
 - Etherdraw
- Google docs

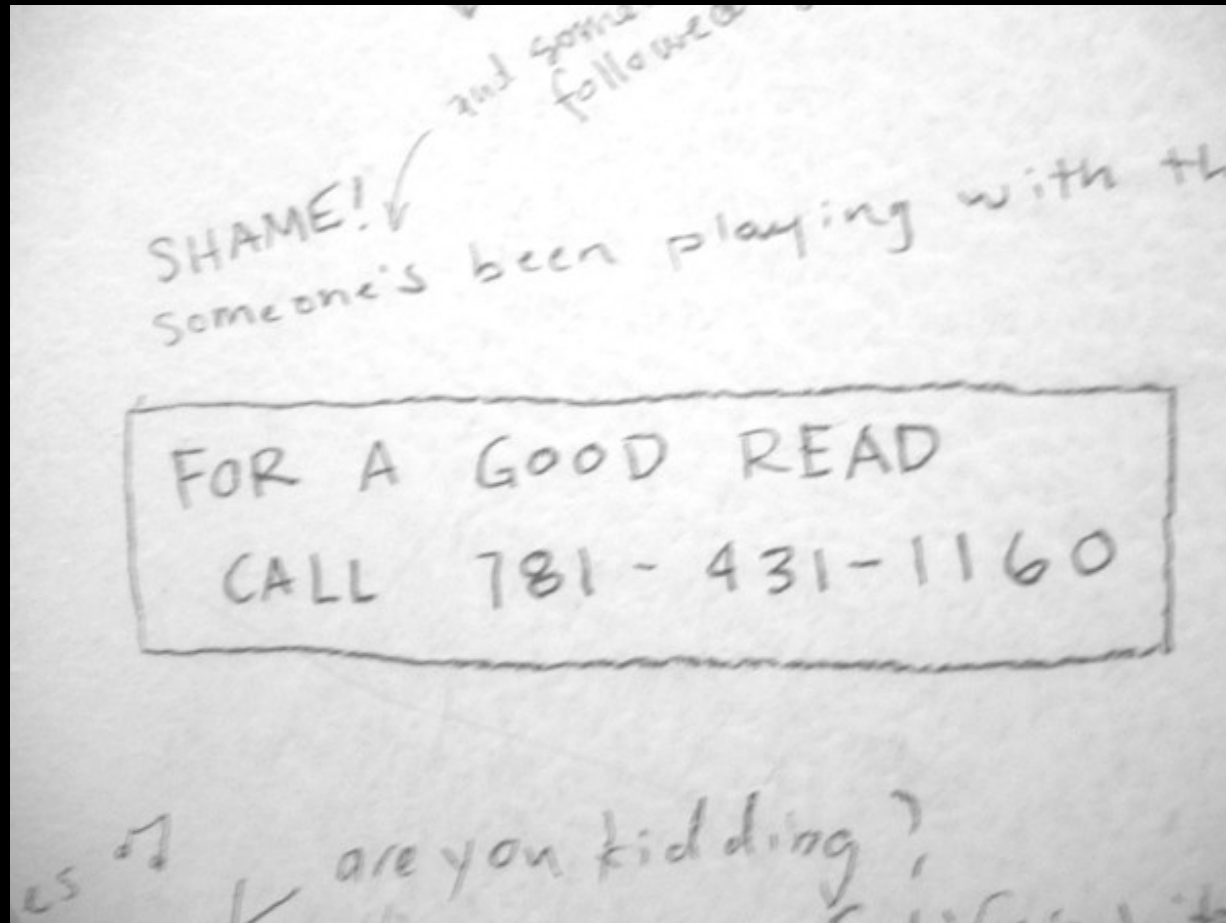
Websockets

- But how does that apply to my CMS
- What about application state?
- Websockets have inherent state
- Coming from classic website model?
 - You probably need a different pattern / paradigm

pub/sub

- Publish–subscribe pattern
- Does literally what it says
- First described in 1987
 - In a talk at the SOSF conference
 - As part of “news” subsystem of Isis toolkit
- Not one standard but a pattern
- WAMP
 - Web Application Messaging Protocol (draft)

pub/sub



pub/sub

- Publish
- .. to a channel
 - sometimes called a topic
- Most implementations return number of subscribers

pub/sub

- Subscribe to (one or more) channels
- Subscribe to (one or more) patterns
 - For example: news.*
- Some servers / libraries
 - Faye
 - RabbitMQ
 - Redis
 - many more . .

Tying it all together



Tying it all together

- What is it good for?
 - Absolutely everything
- Subscribing to channels for pages and widgets
 - Your “user” list view
 - Any widget really
- Subscribing to alerts
- Collaborating

Tying it all together

- Take a look at your “CMS”
- Any table could be an “auto-updating” list
 - Get initial data old-school page-load or via API
 - Paginated
 - Subscribe to data channel
 - Go fancy; subscribe to channel + pattern
- Any form could be a “multi-player” form
 - Implement field locking for bonus points

Tying it all together

- You (client) send event to API (sync)
- API (server) broadcasts to channel (async)
- “Other” clients pick up on change
 - Only when applicable

Implementation



On the backend

- Redis (or other messaging bus)
- NodeJS (or other socket server)
- Some other backend language (optionally)
 - Legacy reasons
- Your favorite database
 - For long-term storing/logging

Interesting Redis perks

- Key-value store
 - With TTL
 - Distributed API rate-limiting
- Partitioning
- Distributed locks
- Cache store
 - Auto-eviction
- Lua scripting

On the frontend

- RxJS
- Angular
 - Listeners, data objects
 - With RxJS
- React
 - Something something Flux
 - Not a lib but a pattern
 - Redux
- PubSubJS
 - Nice and small

But will it scale?



But will it scale?

- Subscribers are “cheap”
- Load balancing is simplified
 - Easiest round-robin setup mostly suffices
- Requests are lighter (less data)
 - Only wanted (subscribed) data gets transmitted
- Easy to add more (micro) servers
- Works very well with automatic scaling (AWS)

How gracefully will it fail?

- Reload drops current subscriptions
- You start again
 - Get initial data
 - Subscribe to channel
- Handling a closed socket
 - Opportunity for an “event”
- Fail-fast

What about security?



What about security?

- WSS
 - Mandatory when serving over https
- JWT
 - Just as safe as session cookies
- Subscriptions dropped when connection closes
 - Safer than token based systems
 - Have a TTL

What about security?

- Unit-testable
 - Via mockups
 - Echo service
- Graceful degradation
 - Since (in most cases) it's just an addition
- Ledger
 - Accountability
 - Why not a blockchain?

Retrofitting



Retrofitting

- Fitting pub/sub into existing systems
- A-sync all the things?
- Starting small
 - Polling widget was hated during writing
 - Usually a retrofit in itself
- Things fall into place
 - On initial load get data directly (old-school) or via API
 - Subscribe and receive changes

Wrapping up



Wrapping up

Use these modern web paradigms

Over 95% browser support*

* Probably even better depending on your target audience

Questions?

AnneJan.com

[@annejanbrouwer](https://twitter.com/annejanbrouwer)
github.com/annejan

NoProtocol.com