# Bringing Real-Time Collaboration to CQ5 Authors

Code Distillery
Julian Sedding, September 2011

# What you get:

- comet technology overview
  - protocols
  - server components
  - client components

# What's the matter?

- concurrent edits
  - problems
    - conflicts
  - solutions
    - locking
    - real-time updates
    - real-time information

# Problems

- conflicts
  - authors cannot see each other
  - overwritten/lost changes
  - accidental activations

# Locking

- pros
  - prevents concurrent edits
- cons
  - requires explicit action
  - blocks others
  - forgotten locks
  - page-level granularity

# Real-Time Updates

- pros
  - reduction of concurrent edits
  - paragraph-level granularity
  - early visibility of conflicts
- cons
  - can be intrusive
  - doesn't prevent conflicts

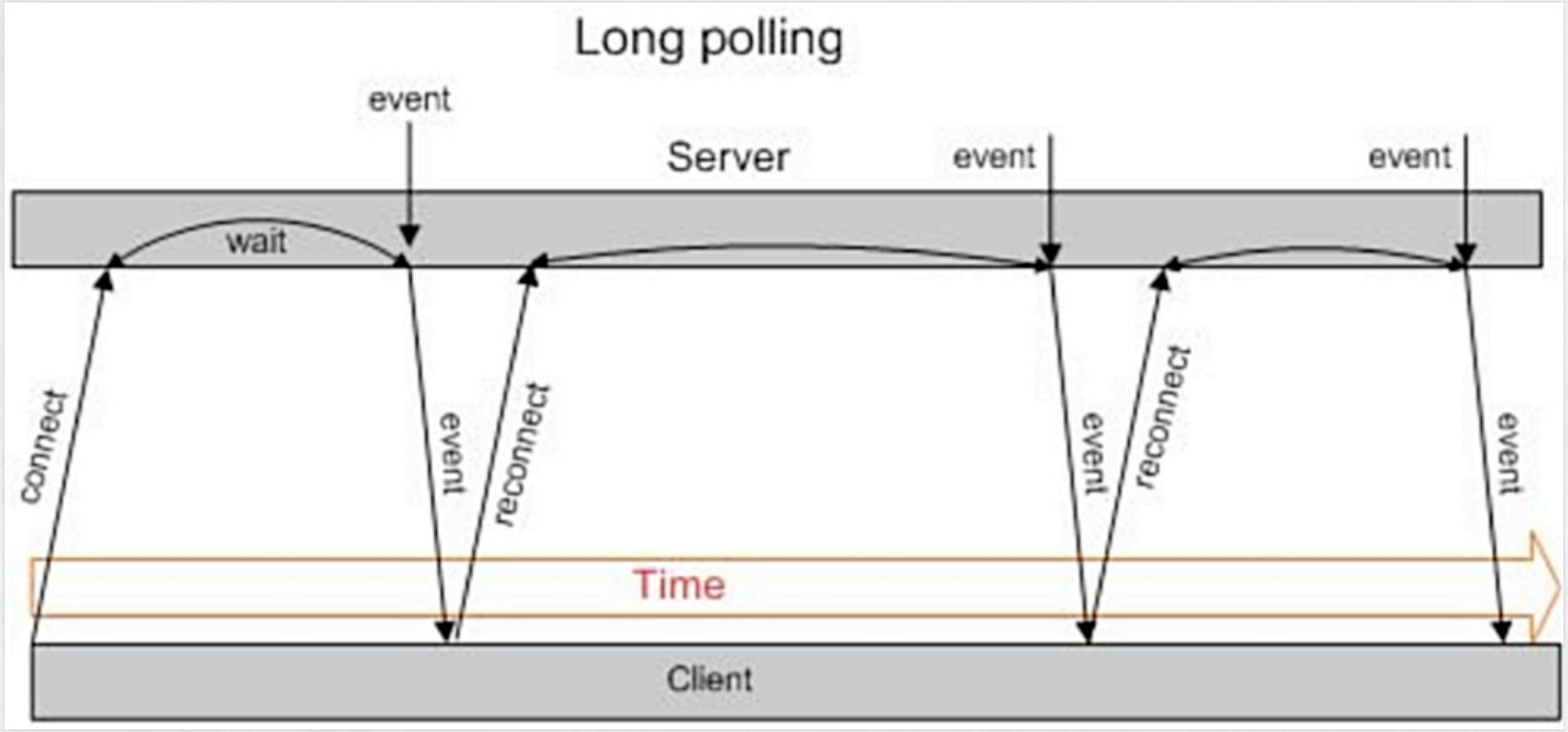
# Real-Time Information

- pros
  - awareness of other authors
  - least intrusive
- cons
  - doesn't prevent conflicts

# Envisaged Solution

- strategies
  - real-time updates
  - real-time information
- technologies
  - buzz: comet
  - non-buzz
    - javascript
    - http long-polling

# Comet …or HTTP long-polling via XHR




source: http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.ajax.devguide.help/docs/PureAjax_pubsub_clients.html

# Server Considerations

- characteristics
  - one open connection per client
  - connections are mostly idle
- implications
  - idle connections need to be cheap
  - scalability of thread per connection model is limited
  - asynchronous processing model scales better

# Client Considerations

- browsers allow a limited number of connections per host
  - implementations should take care to only use one
- exponential backoff strategies in case of failure allow recovery of server
- javascript's same-origin-policy

# Scope

- must-have
  - who else is editing this page?
  - which paragraphs are being edited?
  - what changed?
- nice-to-have (out of scope)
  - chat
  - color coded users

# Technical Requirements

- protocol (pub/sub)
- server
- client (javascript library)

# Protocol

- bayeux
  - json messages
    - simple message passing
- xmpp over bosh
  - xml messages
  - presence management
  - guaranteed message delivery

xmpp: extensible message and presence protocol
bosh: bidirectional streams over synchronous http

# Server Implementations

- bayeux
  - apache felix (since FELIX-1796)
  - jetty
- xmpp
  - openfire
  - ejabberd

# Javascript Client Libraries

- bayeux
  - cometd (from cometd.org)
- xmpp
  - strophe
  - jsjac
  - many more…

# Choice of Technology

- protocol: bayeux
- server: apache felix
- client: cometd(jquery bindings)
- reasons
  - server is an OSGi bundle, allowing for fast dev, test and demo setups
- disclaimer
  - not recommended for production

# Demo

# Integration into CQ

- purely a javascript application
- observe state changes via event listeners
  - CQ.Ext could offer more hooks
- send state changes via comet
- get state changes via comet events
- update CQ's UI (i.e. refresh) based on state changes

# Conclusion

- technology is available
- little experience with scalability
- interesting possibilities, e.g.
  - client to client notifications
  - server to client notifications
  - live content-finder updates, anyone?

# Questions?