Comet Never More!
(HTML 5 WebSockets in Theory and Practice)

Jonas Jacobi & John R. Fallows
Founders
Kaazing
Speaker’s qualifications

- John and Jonas are founders of Kaazing, provider of the open source HTML 5 Websocket gateway
- Both dudes are frequent writers for magazines such as JDJ, JavaMagazine, and AjaxWorld
- John and Jonas speak frequently at conferences Worldwide on HTML 5 Communication
- Authors of Pro JSF and Ajax: Building Rich Internet Components
Overall Presentation Goal

Learn how to architect and build full-duplex Web applications using HTML 5 communication
If we were not restricted by the traditional limitations of HTTP, what type of Web applications would we build?
Web Networking Use cases for Work and Play
Part 1 – Comet and the Future
- Real-time Web
- Comet/Reverse Ajax
- HTML 5 Communication

Break (30 minutes)

Part 2 – Full-duplex Web Today
- Networking Review
- HTML 5 WebSockets
- Kaazing Gateway
Defining Real-Time Web

Web Applications Typically Not Real-Time
Defining Real-Time Web

- Web Clients Receive Server Updates
  - Server-initiated communication
- End-Users Receive Updates Simultaneously
  - Collaboration
Defining Real-Time Web

Or, is it just nearly, nearly real-time?
Defining Real-Time Web

Or, is it just nearly, nearly real-time?
Ajax (XHR)

- Updates Limited To Preset Interval
  - Message buffering increases memory usage
  - Near real-time updates achieved with shorter intervals

- Shorter Updates Cause
  - Increased network traffic
  - Higher frequency connection setup & teardown
Push Technology History

Push technology has been around for a while:

- Pushlets (2002)
- Bang Networks (early adopter)

Previous attempts failed, because:

- Scalability Limitations (Cost etc…)
- Not general purpose
- No standard
Push Technology

- **Server-Initiated Message Delivery**
  - Clients are listening
  - Clients behind firewalls
- **Techniques such as Comet/Reverse Ajax**
- **Delays Completion of HTTP Response**
- **Generally Implemented in JS**
Long Polling and Streaming

Current Comet implementations center around two major areas:

- Long Polling
- Streaming
Long Polling

- Also known as asynchronous-polling
- Request open for a set period
- HTTP headers often account for more than half of the network traffic
Long Polling HTTP Request

From client (browser) to server:

GET /long-polling HTTP/1.1\r\nHost: www.kaazing.com\r\nUser-Agent: Mozilla/5.0 (X11; U; Linux x86_64; en-US; rv:1.9) Gecko/2008061017 Firefox/3.0\r\nAccept: text/html,application/xhtml+xml,application/xml;q=0.9,/*;q=0.8\r\nAccept-Language: en-us,en;q=0.5\r\nAccept-Encoding: gzip,deflate\r\nAccept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\nKeep-Alive: 300\r\nConnection: keep-alive\r\nCache-Control: max-age=0\r\n\r

Long Polling HTTP Response

From server to client (browser):

Date: Tue, 16 Aug 2008 00:00:00 GMT
Server: Apache/2.2.9 (Unix)
Content-Type: text/plain
Content-Length: 12

Hello, world
HTTP Streaming

- Persistent HTTP Connection
  - Pending POST

- Minimizes Latency

- Reduction in Network Traffic

- Optimizes Connection Setup & Tear-Down
  - Keep-alive
  - Security
What’s Missing?

- Not a standard
- No true bi-directional communication
- No guaranteed message delivery
- Complex middle-tier architecture
  - Adds unnecessary latency
Evolving the Web

Ajax

Server

Browser

Comet/Reverse Ajax

Server

Browser

Firewall

WebSockets

Server

Browser

Firewall
HTML 5 Overview

- Next generation application platform
  - Communication (sockets, cross-site)
  - Graphics (2D)
  - Drag ‘n’ drop
  - Storage (transient, persistent)
  - Offline
  - Compatibility
HTML 5 WebSockets

The Communication section:

- WebSockets
- Server-sent events

Not New; TCPConnection API and protocol were initially drafted over two years ago
HTML 5 Server-Sent Events

- Standardizes and formalizes how a continuous stream of data can be sent from a server to a browser
- Introduces `eventsource`—a new DOM element
Connects to a server URL to receive an event stream:

```html
<eventsource src="http://stocks.kaazing.com"
onmessage="alert(event.data)">
```
HTML 5 Server-Sent Events

- Server can add the **ID** header so that clients add a **Last-Event-ID** header
- Used to guarantee message delivery
- Server specify an optional retry header as part of an event in the event stream
HTML 5 WebSockets

- Defines full-duplex communications
  - Operates over a single socket
- Traverses firewalls and routers seamlessly
- Allows authorized cross-domain communication
- Integrates with:
  - Cookie-based authentication
  - Existing HTTP load balancers
HTML 5 WebSockets

- Connection established by upgrading from the HTTP protocol to the WebSocket protocol
- WebSocket data frames can be sent back and forth between the client and the server in full-duplex mode
HTML 5 WebSockets

- Supports a diverse set of clients
- Cannot deliver raw binary data to JavaScript
  - Binary data is ignored if the client is JavaScript
- Enables direct communication with backend systems
HTML 5 WebSockets

- Detects presence of proxy servers
- A tunnel is established by issuing an HTTP CONNECT statement
- Secure Web sockets over SSL can leverage the same HTTP CONNECT technique
Simplified Architecture

WebSocket Server

- RMI - TCP (Full Duplex)
- IMAP - TCP (Full Duplex)
- Jabber - TCP (Full Duplex)
- Custom - TCP (Full Duplex)
- TCP over HTTP (Full Duplex)

Java EE

- EJB
- JMS
- JDBC - TCP (Full Duplex)

Database

- IMAP Server
- Stock Trading Feed

IM Server

RMI - TCP (Full Duplex)

JDBC - TCP (Full Duplex)

Creating a WebSocket instance:

```javascript
var myWebSocket = new WebSocket("ws://www.websocket.org");
```
**HTML 5 WebSockets**

Associating listeners:

```javascript
myWebSocket.onopen = function(evt)
    { alert("Connection open ..." newX); };

myWebSocket.onmessage = function(evt)
    { alert("Received Message: "+ e1t.data); };

myWebSocket.onclose = function(evt)
    { alert("Connection closed."); };
```
Sending messages:

```javascript
myWebSocket.postMessage("Hello WebSocket!");
myWebSocket.postMessage("Goodbye Comet!");
myWebSocket.disconnect();
```
HTML 5 WebSockets

- Enables full-duplex communication to any TCP-based back-end service:
  - JMS
  - Jabber
  - Stomp
  - etc…
Stomp Protocol

- Simple Text-Oriented Messaging Protocol (STOMP)
  - Publish and Subscribe
  - Transactional
  - Acknowledgements
- ActiveMQ, RabbitMQ, …
var myStomp = new StompClient();

myStomp.onopen = function(headers)
    { myStomp.send("Hello STOMP!", "/topic/destination"); }  

myStomp.onmessage = function(headers, body) { alert(body); } 

myStomp.connect("ws://www.websocket.org/stomp");

myStomp.subscribe(destination.value);
XMPP client
HTML 5 – When??

- 2022 AD (Not really)
- Opera already has Server Sent Events
- Mozilla/Firefox patch available – Bug 338583
- Kaazing.org provides this NOW!!!
  - IE 5.5+, Firefox 1.5+, Chrome 0.2+, Safari 3.1+, Opera 9.0+
Summary

- Server-Sent Events standardize and formalize Comet/Reverse Ajax
- WebSockets provides full-duplex communication
- Do I have to wait till 2022 AD?
Comet Never More!
(HTML 5 WebSockets in Theory and Practice)

John R. Fallows & Jonas Jacobi
Founders
Kaazing
Part 1 – Comet and the Future

- Real-time Web
- Comet/Reverse Ajax
- HTML 5 Communication

Break (30 minutes)

Part 2 – Full-duplex Web Today

- Networking Review
- HTML 5 WebSockets
- Kaazing Gateway
Networking Review

Desktop Networking
- Full-duplex bidirectional TCP sockets
- Access any server on the network

Browser Networking
- Half-duplex HTTP request-response
- HTTP polling, long polling, streaming
- Same-origin HTTP requests
Half-Duplex Architecture

Browser

HTTP (Half Duplex)

Servlet

Application Transport Logic

Java EE Container

RMI - TCP (Full Duplex)

EJB

Database

RMI

JavaMail

IMAP Server

IMAP - TCP (Full Duplex)

JABBER - TCP (Full Duplex)

JMS - TCP (Full Duplex)

JMS

Stock Trading Client

Stock Trading Feed

RMI

JDBC - TCP (Full Duplex)

JMS

Custom - TCP (Full Duplex)
HTML 5 Communication

- WebSocket
  - Proxy-friendly text socket for your browser
- Server-Sent Events
  - Standardized HTTP streaming (downstream)
- Cross-Site XMLHttpRequest
  - Secure cross-site remote communication
- postMessage
  - Secure inter-iframe communication
DEMO

Java Messaging Service
Kaazing Protocols

- Protocols
- ByteSocket
- WebSocket
- Server-Sent Events
- Cross-Site XHR
- postMessage
- XHR
- IFrame
Kaazing ByteSocket

- Provides binary socket abstraction
- Leverages text-based WebSocket
  - Encodes payload using base64
- Send and receive ByteBuffers
  - JavaScript has no byte or ByteArray type (yet)
- Kaazing Gateway converts base64
var location = "ws://www.kaazing.org/binary";
var socket = new ByteSocket(location);
socket.onmessage = function(event) {
    alert(event.data.getInt());
}

var buf = new ByteBuffer();
buf.putString("Hello, world", Charset.UTF8);
socket.postMessage(buf.flip());
HTML 5 WebSocket

- Provides Full-Duplex Text Socket
- Send and Receive Strings
- Enables Streaming to Server Too
- Browser Support
  - None (yet)
HTML 5 WebSocket Schemes

ws://www.kaazing.org/text

wss://www.kaazing.org/encrypted-text
var location = "ws://www.kaazing.org/text";
var socket = new WebSocket(location);

socket.onopen = function(event) {
    socket.send("Hello, WebSocket");
}

socket.onmessage =
    function(event) { alert(event.data); }

socket.onclose =
    function(event) { alert("closed"); }
HTML 5 WebSocket Handshake

GET /text HTTP/1.1\r\nUpgrade: WebSocket\r\nConnection: Upgrade\r\nHost: www.kaazing.org\r\n...\r\n
HTTP/1.1 101 WebSocket Protocol Handshake\r\nUpgrade: WebSocket\r\nConnection: Upgrade\r\n...\r\n
Frames can be sent full-duplex
- Either direction at any time

Text Frames use terminator
\x80Hello, WebSocket\0xff

Binary Frames use length prefix
\x00\0x10Hello, WebSocket

Text and binary frames on same WebSocket
Kaazing WebSocket

Protocols

ByteSocket

WebSocket

Server-Sent Events

Cross-Site XHR

postMessage

XHR

IFrame

www.devoxx.com
HTML 5 Server-Sent Events

- Standardizes Comet
  - JavaScript API
  - Wire protocol
    - Encourages adoption by servers
- Browser Support
  - Patch under review for Firefox trunk
**HTML 5 Server-Sent Events**

**HTML DOM Element**

```html
<eventsource src="http://www.kaazing.org/sse"
            onmessage="alert(event.data)" />
```

**HTML DOM API**

```javascript
var es =
document.createElement("eventsource");
es.addEventListener("message",
    function(event) { alert(event.data); },
    false);
es.addEventSource("http://www.kaazing.org/sse");
```
HTML 5 Server-Sent Events

GET /sse HTTP/1.1\r\nHost: www.kaazing.org\r\nLast-Event-ID: 9\r\n...\r\n
200 OK HTTP/1.1\r\n...\r\n
:comment\nid: 10\n
data: Hello, Server-Sent Events

www.devoxx.com
HTML 5 Server-Sent Events
Cross-Site XMLHttpRequest

- W3C Technical Report
  - Access Control for Cross-Site Requests
  - Published Sept 12, 2008
  - http://www.w3.org/TR/access-control/

- Browser Support
  - Firefox 3.1-beta
  - IE8 XDomainRequest (similar)
  - Opera, Safari, Chrome coming
Cross-Site XMLHttpRequest

GET / HTTP/1.1\r\nHost: www.w3.org\r\nOrigin: http://www.kaazing.org\r\n...

200 OK HTTP/1.1\r\nAllow-Origin: http://www.kaazing.org\r\n...

Cross-site XMLHttpRequest
Kaazing Cross-Site XHR

Protocols

ByteSocket

WebSocket

Server-Sent Events

Cross-Site XHR

XHR

Iframe

postMessage
HTML 5 postMessage

- Send Strings Between HTML Documents
  - Documents may be served by different sites

- Standard API
  
  ```javascript
  targetWindow.postMessage(message, targetOrigin)
  window.onmessage = function(event) {
    alert(event.data);
  }
  ```

- Browser Support
  - IE 8, FF 3, Opera 9, WebKit nightlies
DEMO

HTML 5 postMessage
Kaazing postMessage

- Protocols
  - ByteSocket
  - WebSocket
  - Server-Sent Events
  - Cross-Site XHR
    - postMessage
    - XHR
    - Iframe
Kaazing postMessage

- **HTML 5**
  - `targetWindow.postMessage(message, targetOrigin)`

- **Kaazing**
  - `postMessage0(targetWindow, message, targetOrigin)`

- Documents may be served by any domain
  - http://www.kaazing.org:8000/org-domain
  - https://www.kaazing.net:9000/secure-net-domain
Kaazing Protocols Support

Protocols

ByteSocket

WebSocket

Server-Sent Events

Cross-Site XHR

postMessage

XHR

IFrame
Kaazing Protocols

- Text or Binary
  - Stomp
  - XMPP
  - IRC
  - Telnet
  - IMAP
  - SMTP
  - Custom…
Kaazing Gateway Scalability

- Based on SEDA (Staged Event-Driven Architecture)
  - Leverages Java New I/O (NIO)

- Concurrency
  - Proportional to bandwidth not connections

- Latency
  - Socket integration, bytes-in, bytes-out

- Stateless
  - Minimal memory usage, balancing, failover
Kaazing Gateway Security

Web Page
- www.server.com
- stock.server.com
- chat.server.com

Chat
- wss://stock.server.com:443
- tcp://chat.internal.net:5222

Stock
- wss://stock.server.com:443
- tcp://stock.internal.net:61613

Public Internet
- IP Addresses: 65.43.2.1

DMZ
- IP Addresses: 10.0.0.0/24
- 172.16.0.0/20
- 192.168.0.0/16

Internal Network
- IP Addresses: 192.168.0.0/16

Director Server

Kaazing Gateway
- WSS: wss://gateway.dmz.net:8080
- TCP: tcp://gateway.dmz.net:9090
- tcp://stock.internal.net:61613
- tcp://chat.internal.net:5222
Kaazing Enterprise Gateway

Features

- Adobe Flex APIs
- Flash runtime detection
- EncryptedKeyring
- Single sign-on
- Protocol Validation
- Protocol Security Enhancements
- Management
DEMO

Kaazing XMPP Client
Summary

- HTML 5 Communication is here
- WebSockets and SSE standardize Comet
- Avoid vendor lock-in, the standards shall set you free!
- Kaazing Community – www.kaazing.org
If HTTP did not restrict your creativity, what Web application would YOU create?
Q&A
Thanks for your attention!

http://www.kaazing.org

http://www.w3c.org