goto; conference aarhus

HTML5 WEBSOCKETS

Brad Drysdale

Director of Technlogy

KAAZING

INTERNATIONAL
SOFTWARE DEVELOPMENT
CONFERENCE

Kaazing. Connect. Everything.

HTML5 WebSockets

The Web Communication Revolution

Brad Drysdale
Director of Technology - Kaazing

Next Generation Web-based…

Single Trader Desktop

Next Generation Web-based…

Gaming/Betting Platform

Single Trader Desktop

Next Generation Web-based…

Gaming/Betting Platform

Single Trader Desktop

Real-time Gambling

Next Generation Web-based…

Smart Metering

Gaming/Betting Platform

On-line Gaming

IPTV

Logistics &

Supply Chain

Single Trader Desktop

Social Networking

Real-time Gambling

RFID Tracking

eComm

Monitoring/Dashboards

Reaching the Masses…

W W W

Going big…

Extending your business across the Web means $$$

Yet you say…

“I can already do this today”

Hang on…

Can you *really*?

Is your proposed solution…

- Low Latency, Real-time Data ?
- Bandwidth Efficient ?
- Open Standards ?
- Require Plugins ? (Note: IE10)
- Platform Neutral ?
- Seamless support for Mobile/Tablet OS ?
- Cloud Ready ?
- Future Proofed ?
- Web Scale ?

Is your proposed solution…

- Low Latency, Real-time Data ?
- Bandwidth Efficient ?
- Open Standards ?
- Require Plugins ? (Note: IE10)
- Platform Neutral ?
- Seamless support for Mobile/Tablet OS ?
- Cloud Ready ?
- Future Proofed ?
- Web Scale ?
- **Truly Web Competitive ???**

Here we go…

So what’s new…

Here we go…

Here’s how you get
Web Competitive

Welcome HTML5

- HTML5 is the next set of W3C HTML standards
- Offers new and enhanced features as building blocks for next generation RIAs
- Industry standard backed by Google, Apple, Mozilla, Microsoft, Cisco, etc
- Many of the browser vendors have already implemented several of these features
- The race is on to implement the rest and be the best

HTML5 Features

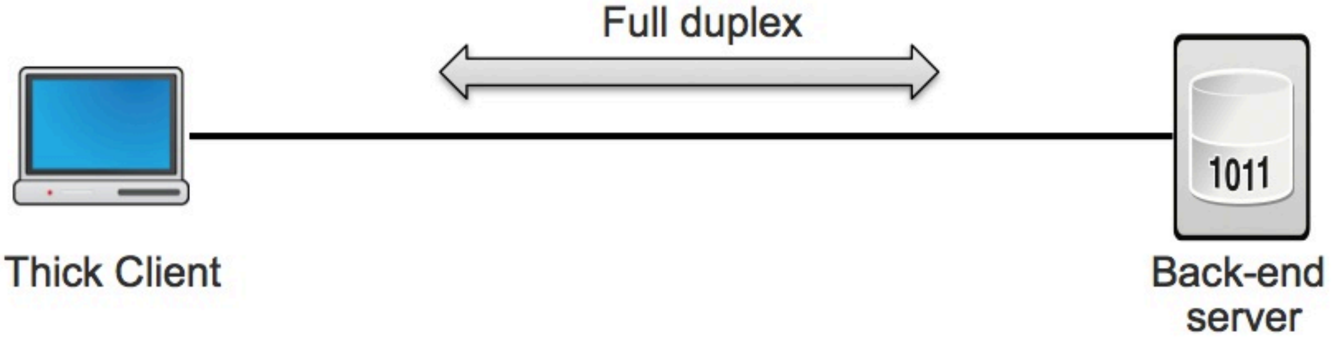
- HTML5 features:
  - New forms and media (audio/video) elements
  - New APIs
    - Canvas
    - Web Workers
    - Geolocation
    - Offline storage
    - **WebSockets**
    - Communication APIs
  - Lots of other cool stuff which is content for a different talk

Once upon a time…

Let's revisit the
good old days…

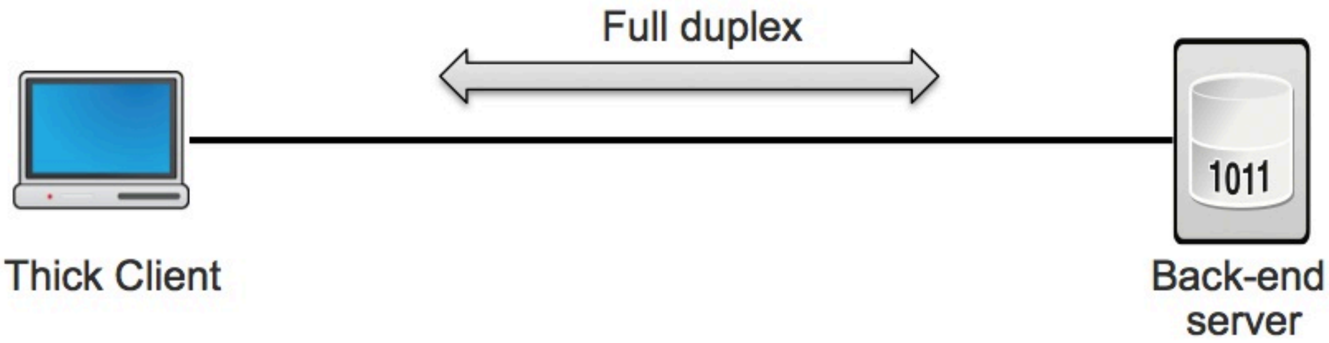
Client-Server Architecture

TCP socket



Client-Server Architecture

TCP socket

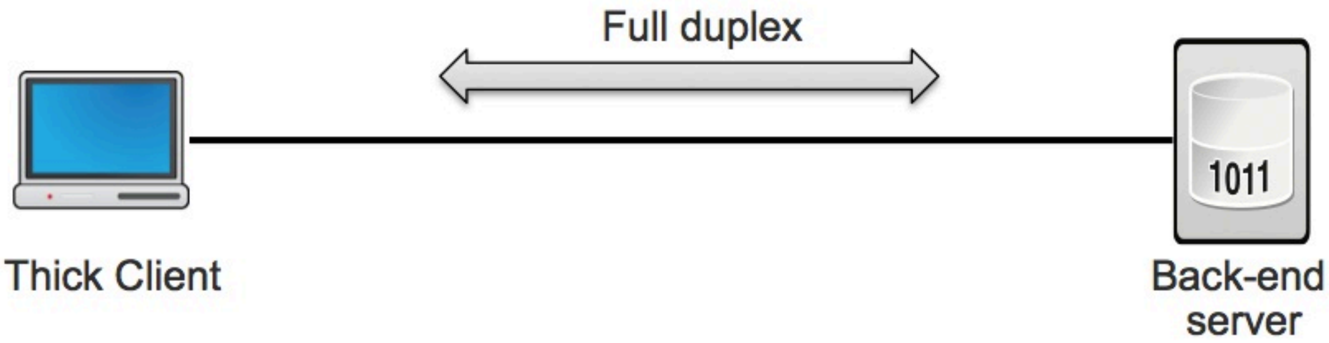


Full duplex transmission of rich business protocols between server to client

Client-Server Architecture

Now let’s extend this to the Web!

TCP socket



Full duplex transmission of rich business protocols between server to client

Out spending again…

Middleware.

Out spending again…

JBoss™
ORACLE®
Microsoft®
bea™
IBM®

What is this stuff?

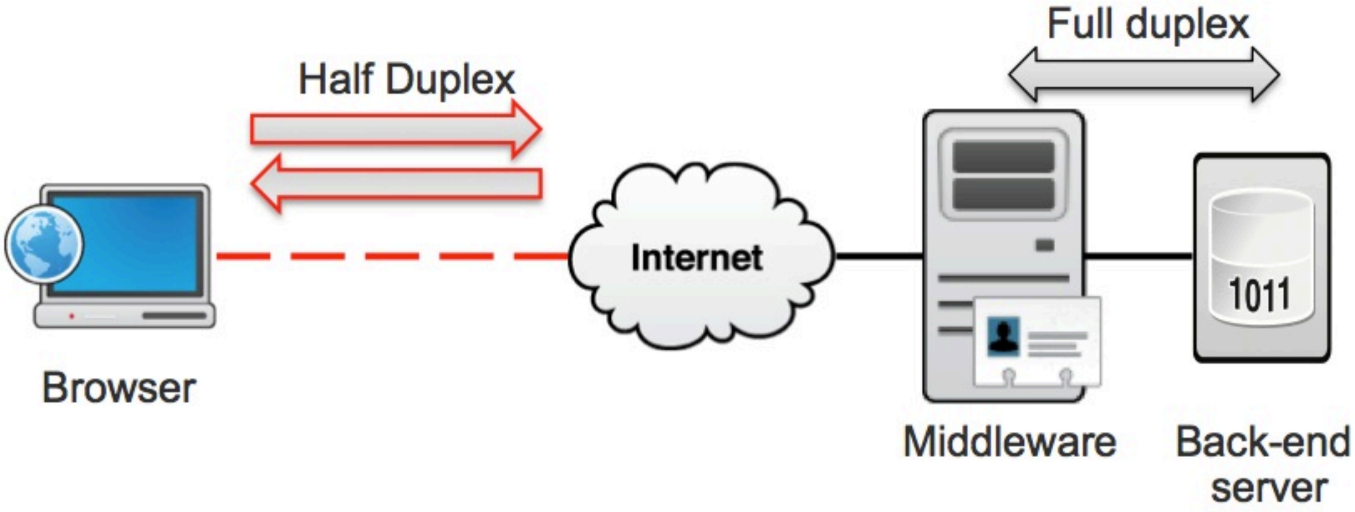
Middleware.

Hint is in the name…

Middleware.

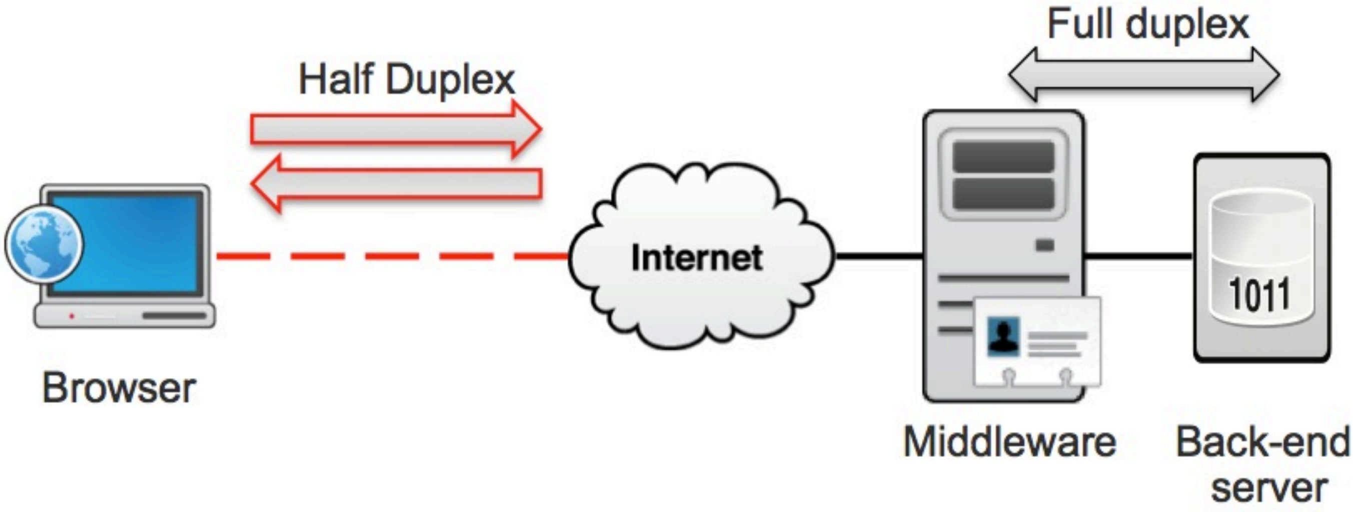
HTTP Web Architecture

HTTP



HTTP Web Architecture

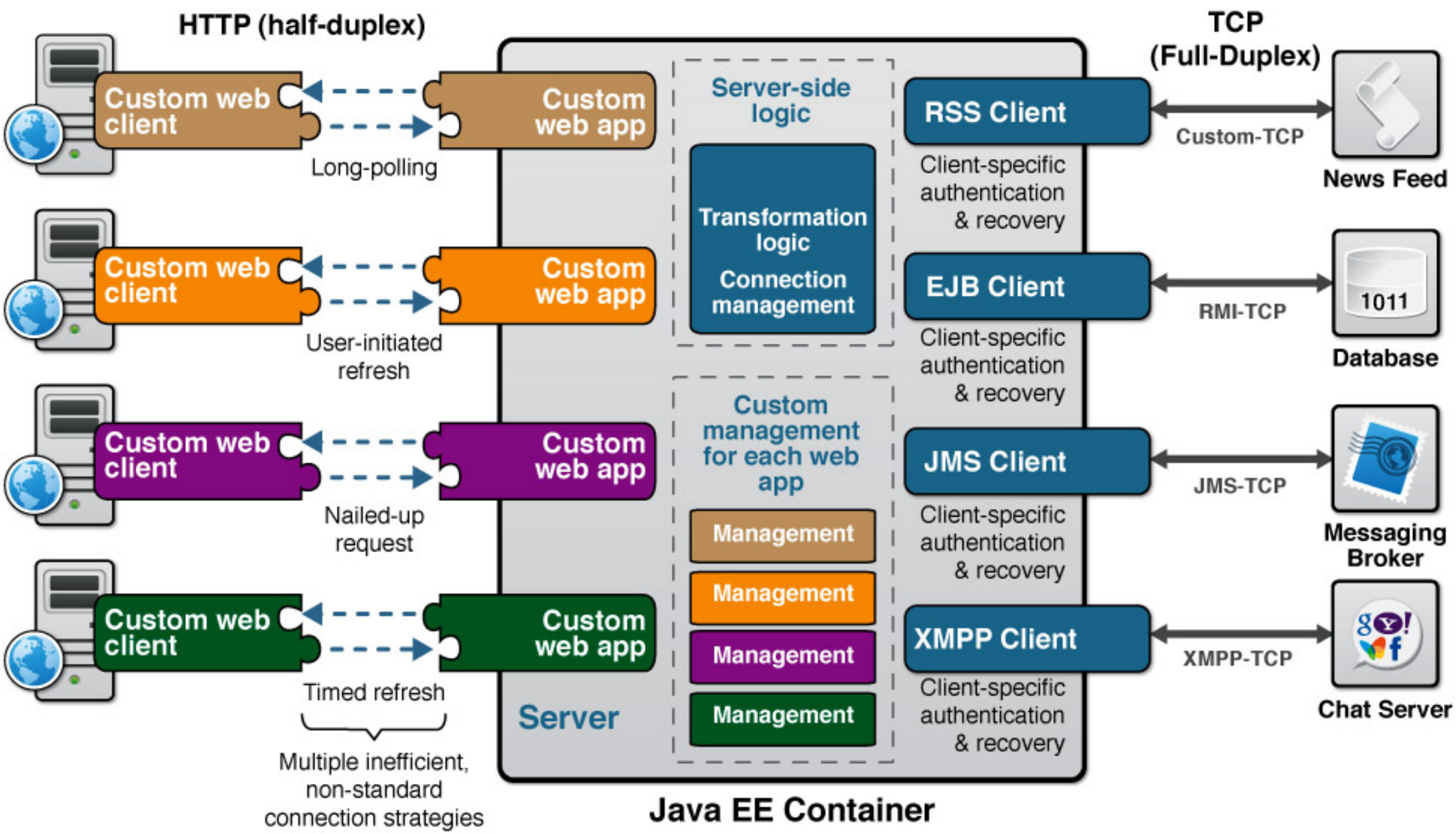
HTTP



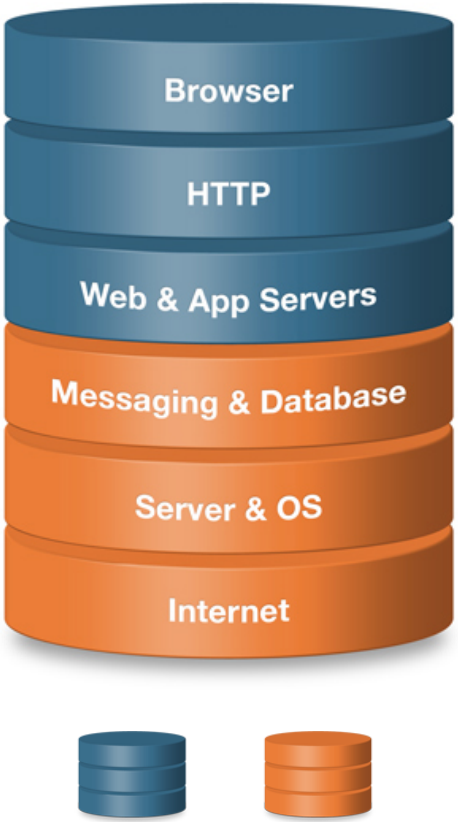
Middleware is the glue between HTTP and TCP

HTTP Is Not Full Duplex

Half-Duplex Web Architecture



The Legacy Web Stack



Half duplex Full duplex

- Designed to serve static documents
  - HTTP
  - Half duplex communication
- High latency
- Bandwidth intensive
  - HTTP header traffic approx. 800 to 2000 bytes overhead per request/response
- Complex architecture
  - Not changed since the 90’s
  - Plug-ins
  - Polling / long polling
  - Legacy application servers
- Expensive to “Webscale” applications

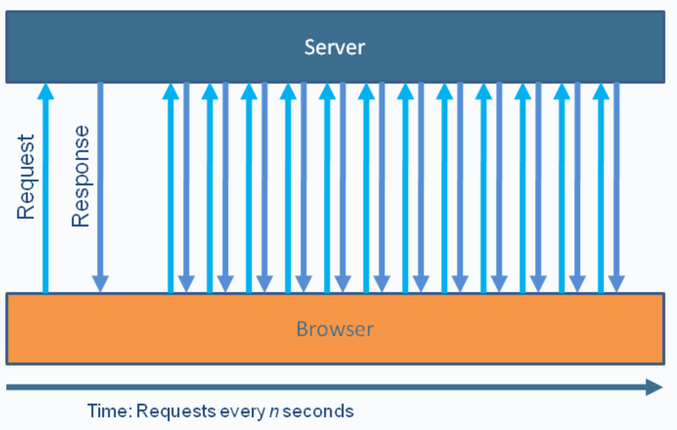
HTTP Shows It's Age

Squeeze every
last drop…

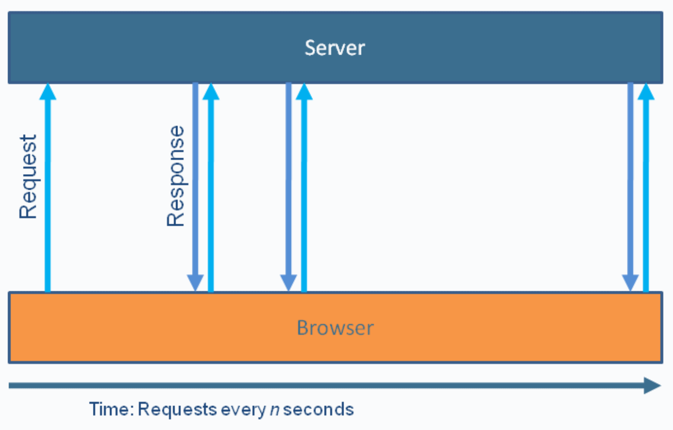
Hack the Web for Real-Time

- Ajax applications use various “hacks” to simulate real-time communication
  - Polling - HTTP requests at regular intervals and immediately receives a response
  - Long Polling - HTTP request is kept open by the server for a set period
  - Streaming - More efficient, but not complex to implement and unreliable
- Excessive HTTP header traffic, significant overhead to each request response

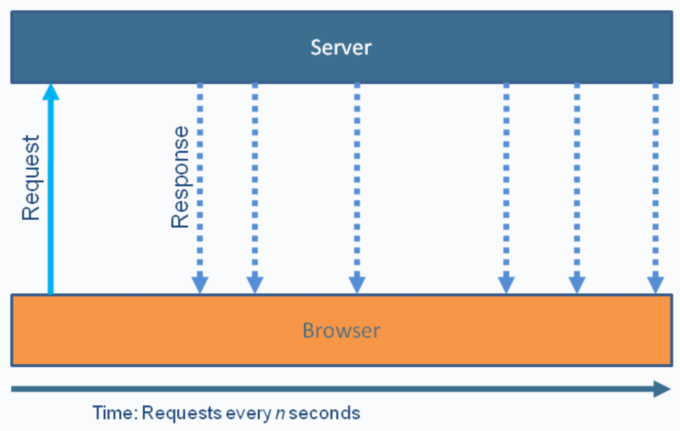
Hack the Web for Real-Time



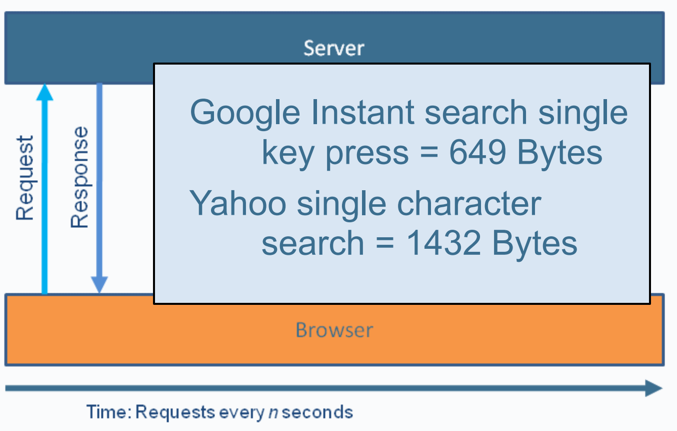
Polling



Long-Polling



Streaming



Google Instant search single key press = 649 Bytes

Yahoo single character search = 1432 Bytes

Request Response Overhead

HTTP Request Headers

Client

```
GET /PollingStock//PollingStock HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:
1.9.1.5) Gecko/20091102 Firefox/3.5.5
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/
*;q=0.8
Accept-Language: en-us
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://localhost:8080/PollingStock/
Cookie: showInheritedConstant=false;
showInheritedProtectedConstant=false; showInheritedProperty=false;
showInheritedProtectedProperty=false; showInheritedMethod=false;
showInheritedProtectedMethod=false; showInheritedEvent=false;
showInheritedStyle=false; showInheritedEffect=false;
```

HTTP Response Headers

Server

```
HTTP/1.x 200 OK
X-Powered-By: Servlet/2.5
Server: Sun Java System Application Server 9.1_02
Content-Type: text/html;charset=UTF-8
Content-Length: 321
Date: Sat, 07 Nov 2009 00:32:46 GMT
```

- Total (unnecessary) HTTP request and response header information overhead: 871 bytes (example)
- Overhead can be as much as 2000 bytes

HTTP Header Traffic Analysis

- Example network throughput for HTTP request and response headers associated with polling
  - **Use case A**: 1,000 clients polling every second:
    - Network throughput is (871 x 1,000) = 871,000 bytes = 6,968,000 bits per second (**~6.6 Mbps**)
  - **Use case B**: 10,000 clients polling every second:
    - Network throughput is (871 x 10,000) = 8,710,000 bytes = 69,680,000 bits per second (**~66 Mbps**)
  - **Use case C**: 100,000 clients polling every second:
    - Network throughput is (871 x 100,000) = 87,100,000 bytes = 696,800,000 bits per second (**~665 Mbps**)

About Ajax and Comet

- Great toilet cleaners…
- Ajax (Asynchronous JavaScript and XML) is used to build highly interactive Web apps
  - Content can change without loading the entire page
  - User-perceived low latency
- "Real-time" often achieved through polling and long-polling
- Comet lack of a standard implementation
- Comet adds lots of complexity

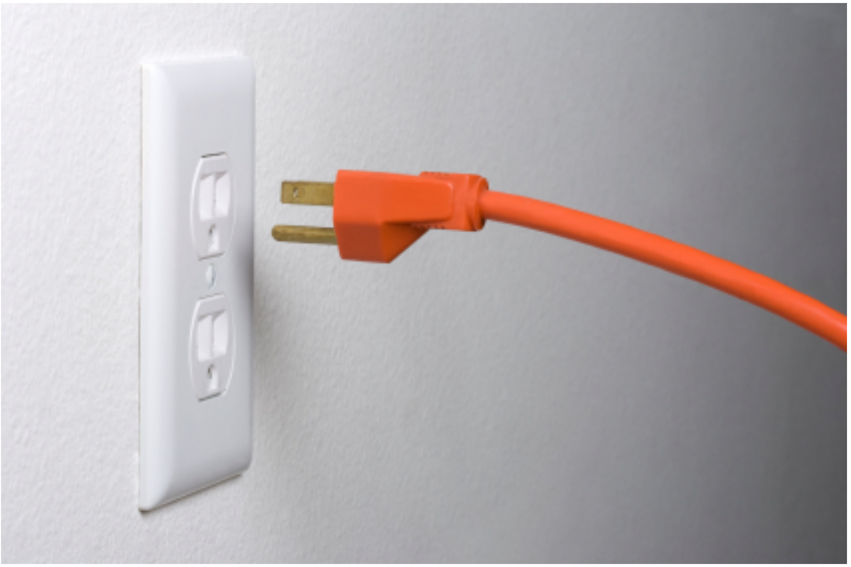
Traditional vs Web

- Traditional Computing
  - Full-duplex bidirectional TCP sockets
  - Access any server on the network
- Web Computing
  - Half-duplex HTTP request-response
  - HTTP polling, long polling fraught with problems
  - Lots of latency, lots of bandwidth, lots of server-side resources
  - Bespoke solutions became very complex over time

Complexity does not scale

The Web gets a new Superhero

Enter HTML5 WebSocket!

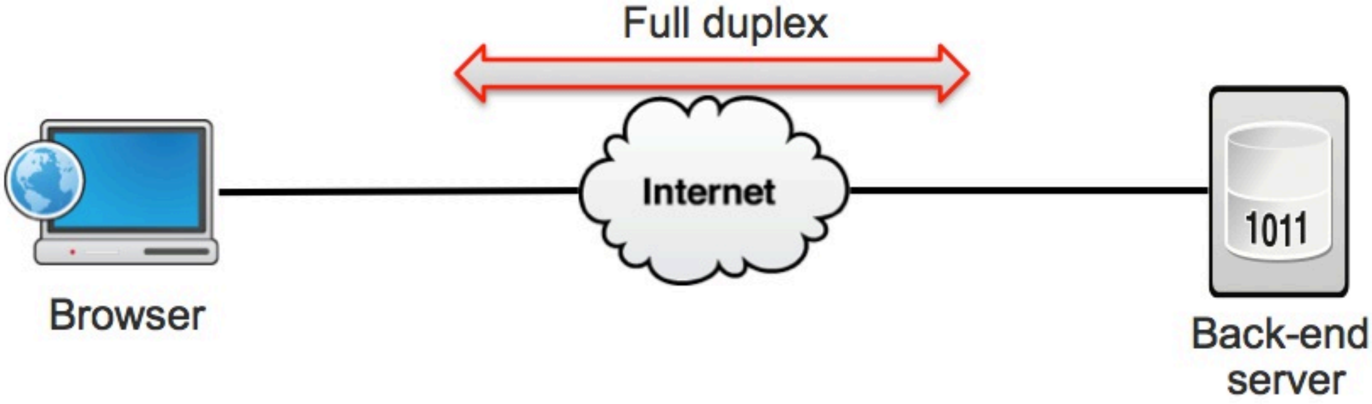


HTML5 WebSocket

- WebSockets provide an improved Web Comms fabric
- Consists of W3C API and IETF Protocol
- Provides a full-duplex, single socket over the Web (even using ports 80 and 443)
- Traverses firewalls, proxies, and routers seamlessly
- Leverages Cross-Origin Resource Sharing
- Share port with existing HTTP content
- Can be secured with TLS (much like HTTPS)

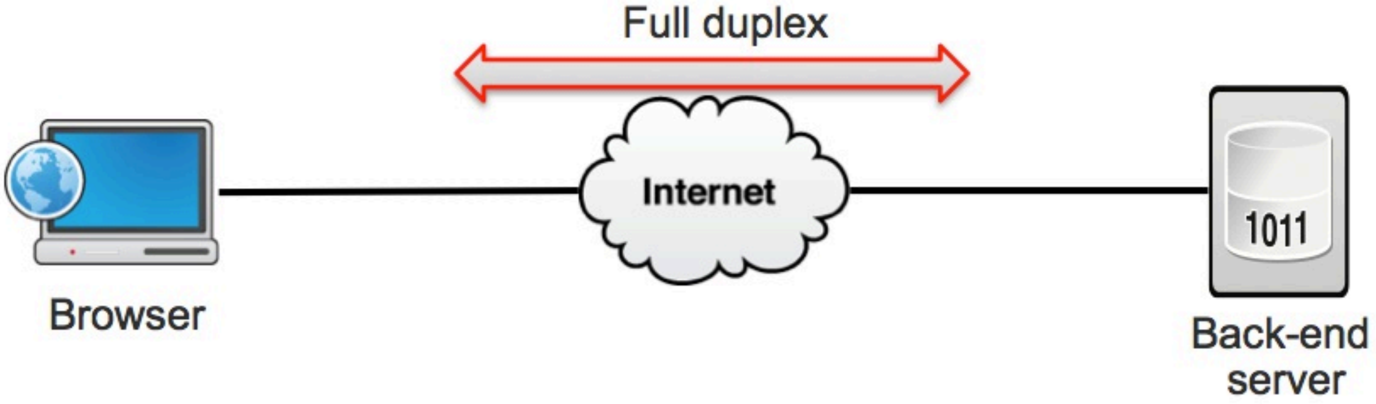
The New Web Architecture

WebSocket



The New Web Architecture

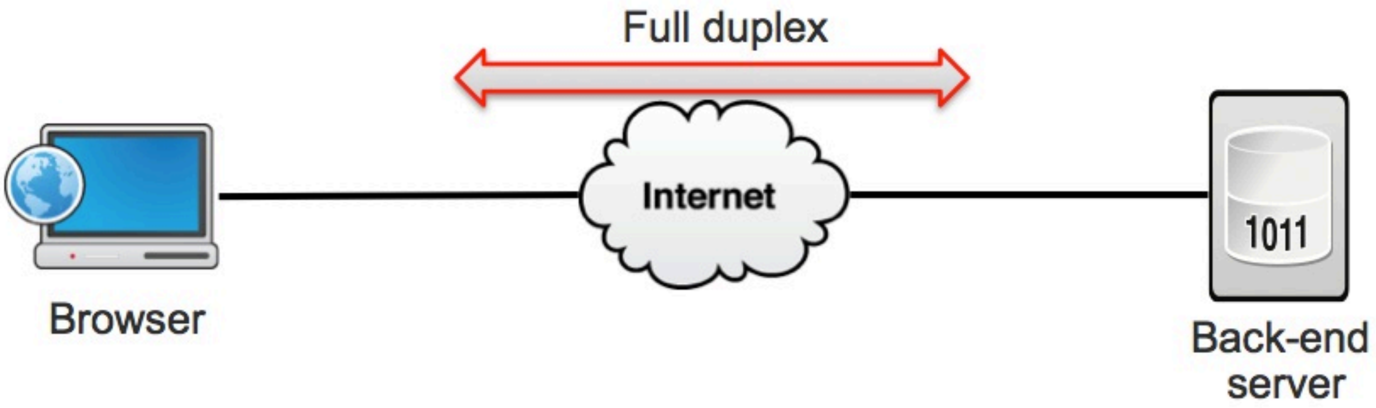
WebSocket



Regain the full duplex transmission of rich business protocols between server to client

The New Web Architecture

WebSocket



Regain the full duplex transmission of rich business protocols between server to client, **across the Web, across the Cloud**

Checking For Browser Support

JavaScript

```
//Checking for browser support
if (window.WebSocket) {
    document.getElementById("support").innerHTML =
    "HTML5 WebSocket is supported";
  } else {
     document.getElementById("support").innerHTML =
     "HTML5 WebSocket is not supported";
  }
```

Current Browser Support

Browser Support for WebSocket

- Chrome
- Safari
- Firefox (need to turn on)
- Opera 10.7 (need to turn on)
- Internet Explorer 9+ Beta

WebSocket Emulation

- Kaazing WebSocket Gateway
  - http://www.kaazing.com/download
  - Makes WebSocket work in all browsers today (including I.E. 6)
- Flash WebSocket implementation
  - http://github.com/gimite/web-socket-js
  - Requires opening port on the server's firewall

How do I use: WebSocket API

JavaScript

```
//Create new WebSocket
var mySocket = new WebSocket("ws://
www.WebSocket.org");

// Associate listeners
mySocket.onopen = function(evt) {
       alert("Connection open…");
};

mySocket.onmessage = function(evt) {
       alert("Received message: " + evt.data);
};

mySocket.onclose = function(evt) {
       alert("Connection closed…");
};
```

Using the WebSocket API

JavaScript

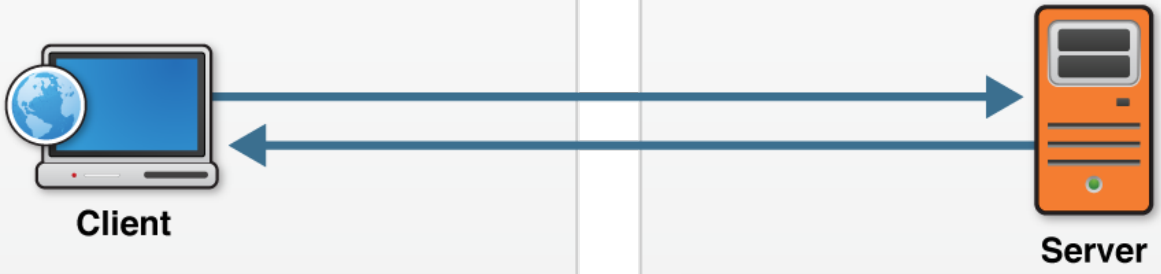
```
// Sending data
mySocket.send("WebSocket Rocks!");

// Close WebSocket
mySocket.close();
```

WebSocket Handshake

Client wants *ws://example.com/chat*

Server accepts



REQUIRED

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-Websocket-Key: 16-byte nonce, base64 encoded
Sec-Websocket-Version: 6
```

OPTIONAL

```
Sec-Websocket-Origin: http://example.com
Sec-Websocket-Protocol: protocol [,protocol]*
Sec-Websocket-Extensions: extension [,extension]*
Cookie: cookie content & other cookie-related headers
```

REQUIRED

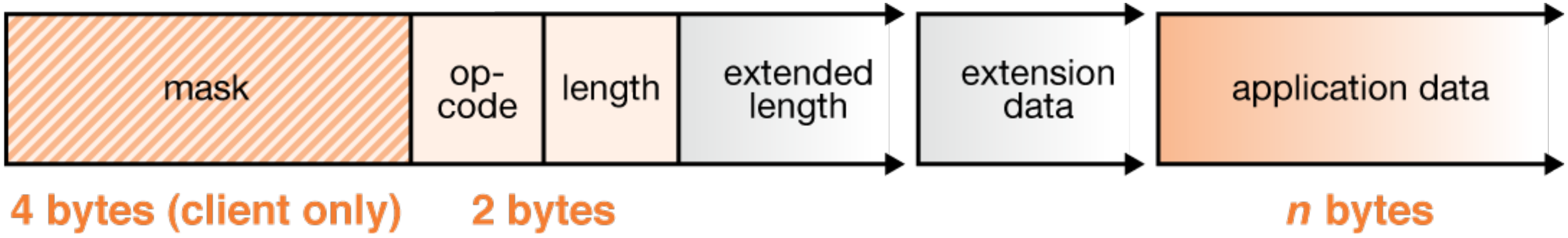
```
HTTP/1.1 101 "Switching Protocols" or other description
Upgrade: websocket
Connection: Upgrade
Sec-Websocket-Accept: 20-byte MD5 hash in base64
```

OPTIONAL

```
Sec-Websocket-Protocol: protocol
Sec-Websocket-Extensions: extension [,extension]*
```

WebSocket Frames

- Frames have a few header bytes
- Data may be text or binary
- Frames from client to server are masked (XORed w/ random value) to avoid confusing proxies



Reduction in Network Traffic

- With WebSocket, each frame has only several bytes of packaging (a 500:1 or even 1000:1 reduction)
- No latency involved in establishing new TCP connections for each HTTP message
- Dramatic reduction in unnecessary network traffic and latency
- Remember the Polling HTTP header traffic? *665 Mbps network throughput for just headers*

HTTP Header Traffic Analysis

Client	Overhead Bytes	Overhead Mbps
1,000	871,000	~6,6*
10,000	8,710,000	~66
100,000	87,100,000	~665

`* 871,000 bytes = 6,968,000 bits = ~6.6 Mbps`

WebSocket Framing Analysis

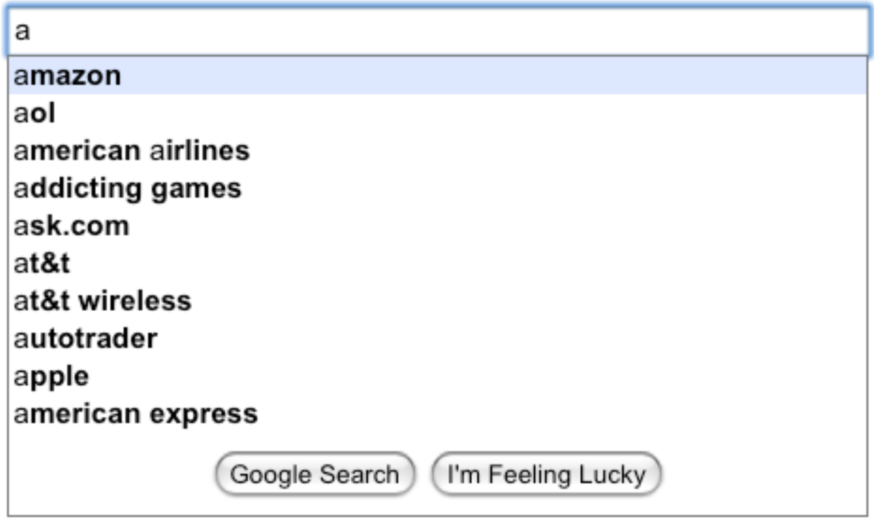
Client	Overhead Bytes	Overhead Mbps
1,000	2,000	~0.015*
10,000	20,000	~0.153
100,000	200,000	~1.526

* 2,000 bytes = 16,000 bits (~0.015 Mbps)

HTTP versus WebSockets

KAAZING

Example: Entering a character in a search field with auto suggestion



a

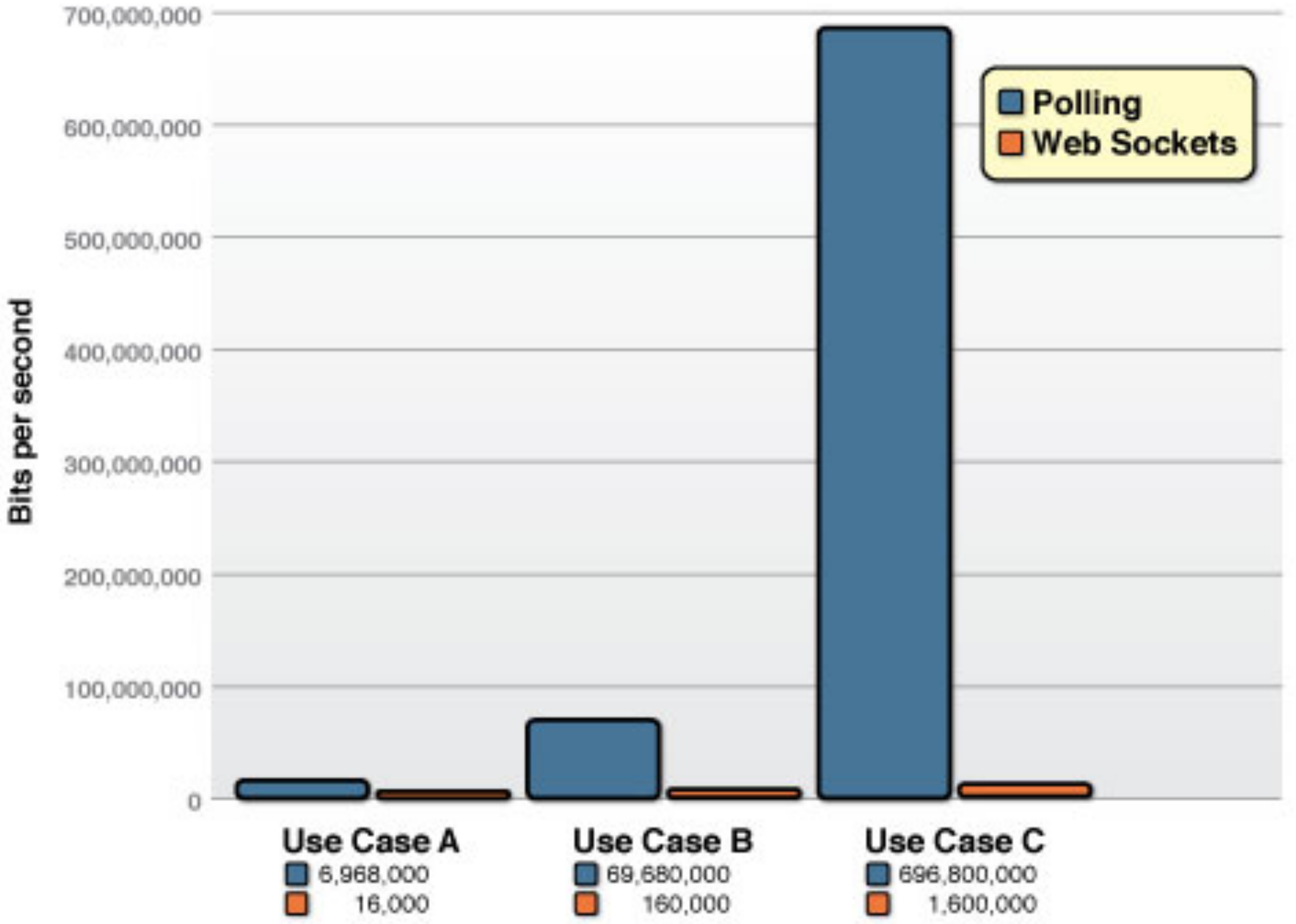
amazon
aol
american airlines
addicting games
ask.com
at&t
at&t wireless
autotrader
apple
american express

	HTTP traffic*	WebSocket Traffic*
Google	788 bytes, plus 1 byte	2 bytes, plus 1 byte
Yahoo	1737 bytes, plus 1 byte	2 bytes, plus 1 byte

* Header information for each character entered into search bar

WebSockets reduces bandwidth overhead up to 1000x

Polling vs. Web Sockets



Overheard…

“Reducing kilobytes of data to 2 bytes…and reducing latency from 150ms to 50ms is far more than marginal. In fact, these two factors alone are enough to make WebSocket seriously interesting to Google.”

—Ian Hickson (Google, HTML5 spec lead)

Verbatim

“The world is moving to HTML5”

—Apple

“The Web has not seen this level of transformation, this level of acceleration, in the past ten years… we're betting big on HTML5”

—Vic Gundotra, VP of Engineering, Google

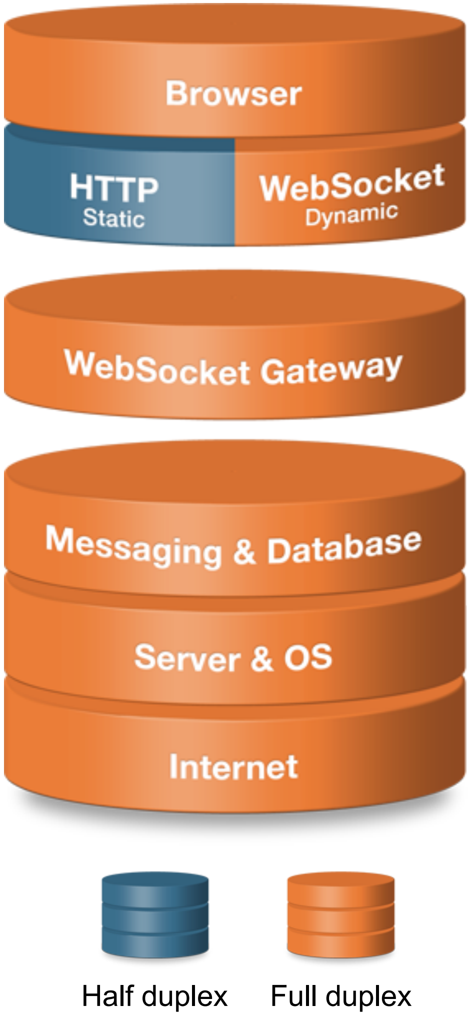
“In a nutshell, we love HTML5, we love it so much we want it to actually work.

—Dean Hachamovitch, General Manager for Internet Explorer, Microsoft

“I had no idea there was so much HTML5 already in play”

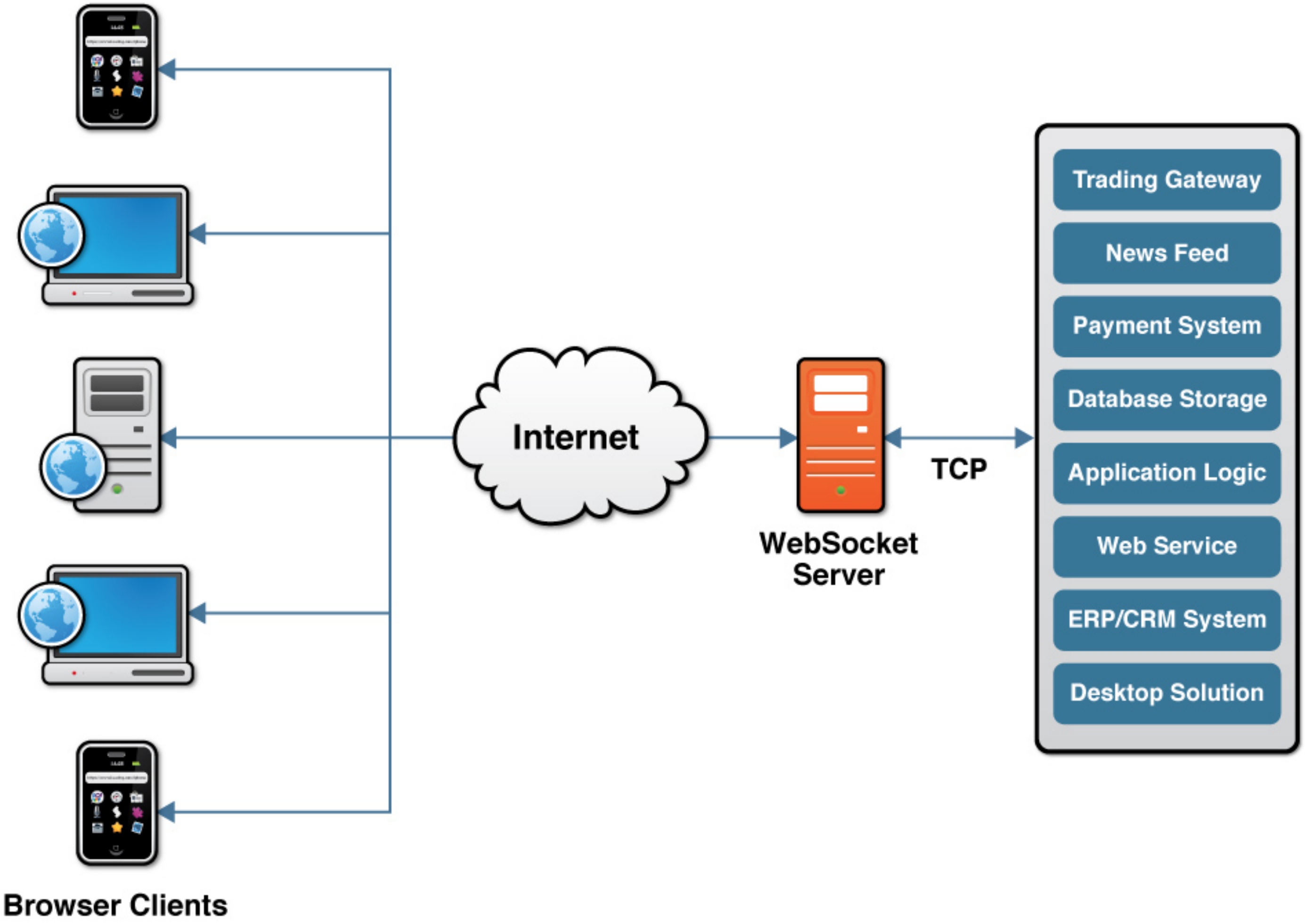
—Tim O’Reilly

The New Web Stack



- Designed for full-duplex high performance transactional Web
  - HTTP & HTML5 WebSocket
  - Full duplex communication
- Lower latency
- Reduced bandwidth
- Simplified architecture
- Massive scalability

WebSockets Architecture



Current Browser Support

Browser Support for WebSocket

- Chrome
- Safari
- Firefox (need to turn on)
- Opera 10.7 (need to turn on)
- Internet Explorer 9+ Beta

Dev channel

Server Support

- Kaazing WebSocket Gateway
- Apache mod_pywebsocket
- Jetty
- phpwebsockets
- web-socket-ruby
- Yaws (Erlang)
- Node.js / Socket.io

- This slide is forever out of date…

Got WebSocket !

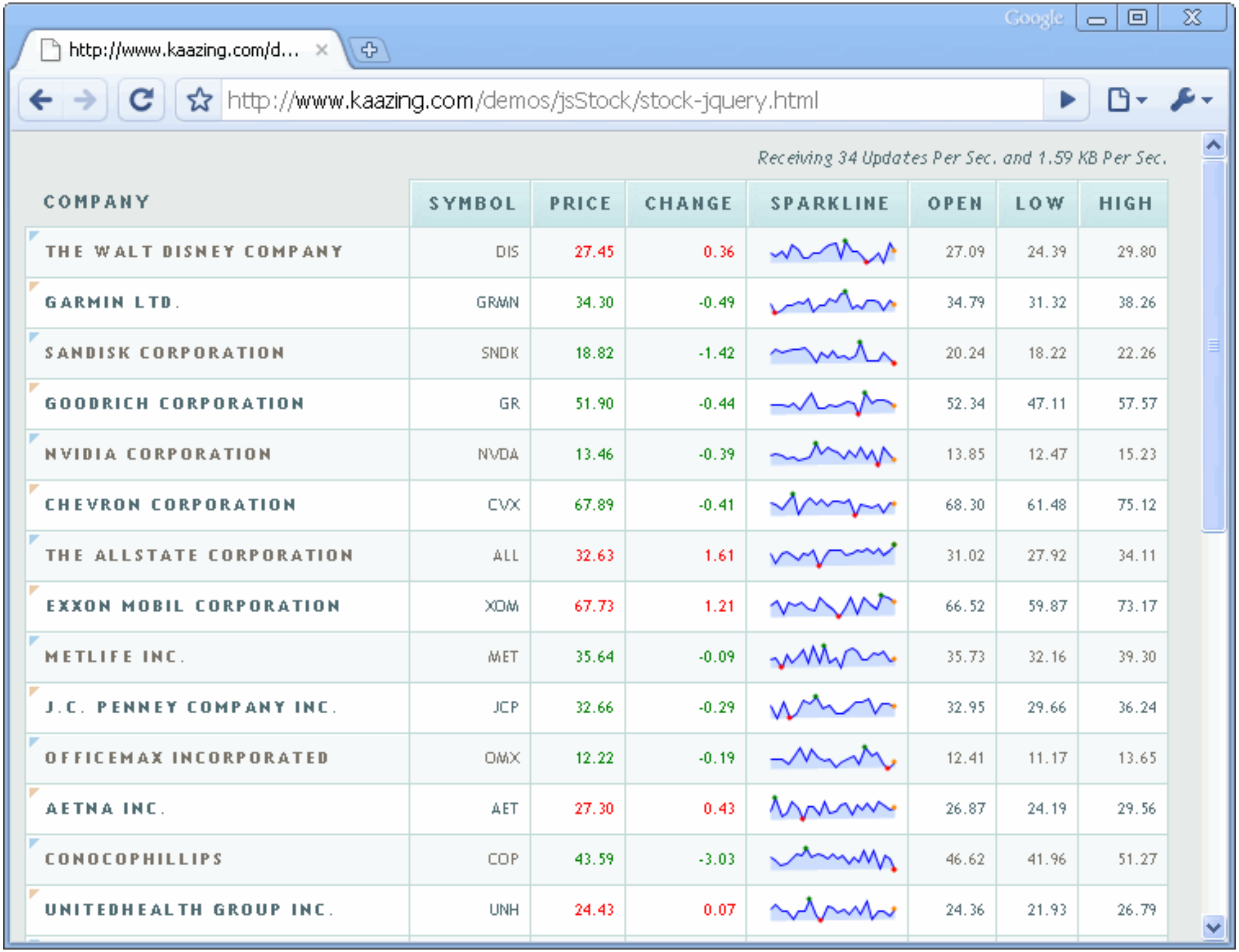
Now what ?

Discovering WebSockets

Got WebSocket. Now What?

- Major upgrade for web traffic, use it!
- Build high performance, scalable messaging for web apps
- Extend the reach of *any* TCP-based protocol you want, all the web to the browser
- The browser is a true client of that protocol – powerful paradigm shift
- Aggregate data and apply business logic at the client

Example: Financial Apps

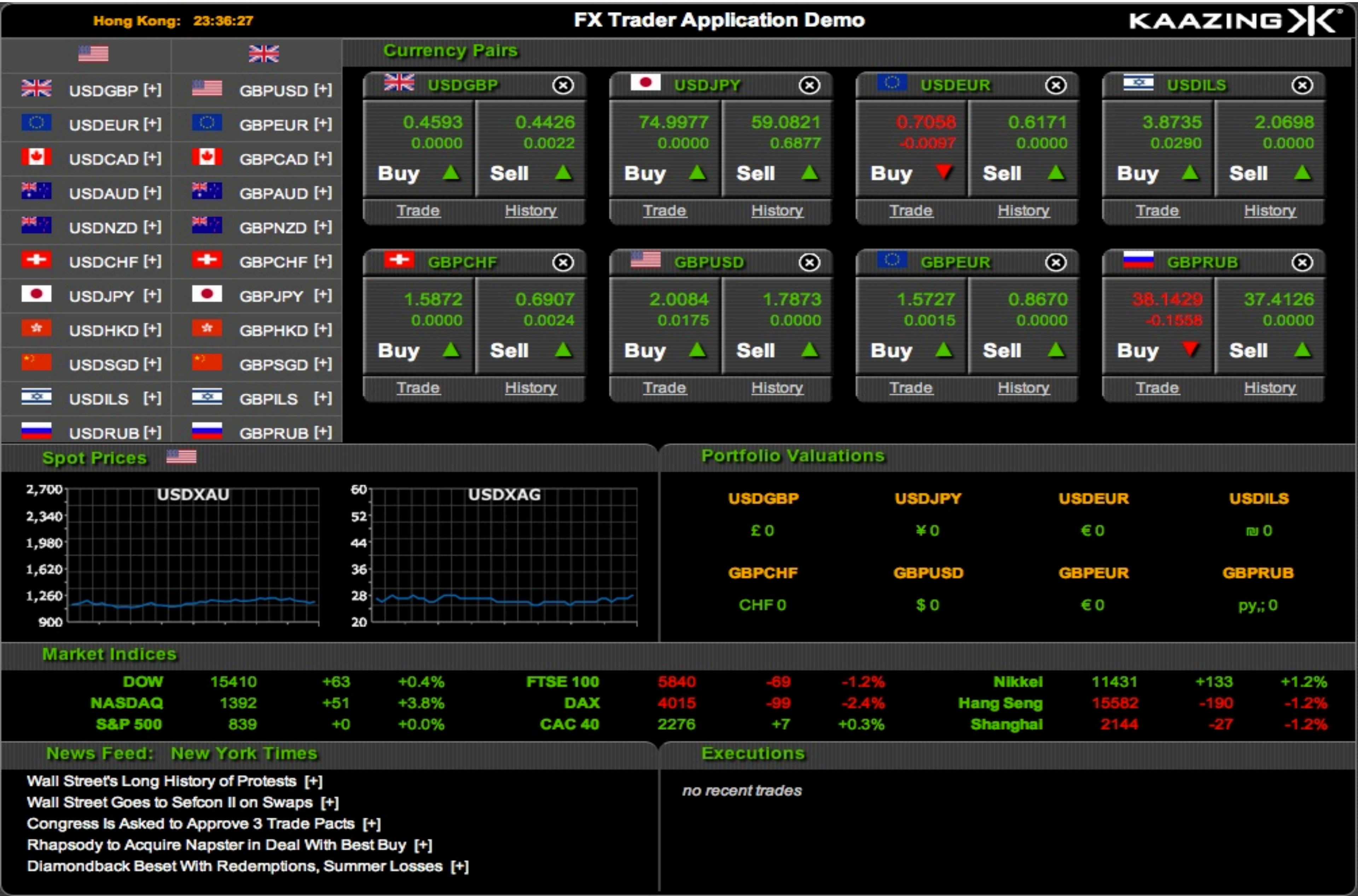


http://www.kaazing.com/demos/jsStock/stock-jquery.html

Receiving 34 Updates Per Sec. and 1.59 KB Per Sec.

COMPANY	SYMBOL	PRICE	CHANGE	SPARKLINE	OPEN	LOW	HIGH
THE WALT DISNEY COMPANY	DIS	27.45	0.36		27.09	24.39	29.80
GARMIN LTD.	GRMN	34.30	-0.49		34.79	31.32	38.26
SANDISK CORPORATION	SNDK	18.82	-1.42		20.24	18.22	22.26
GOODRICH CORPORATION	GR	51.90	-0.44		52.34	47.11	57.57
NVIDIA CORPORATION	NVDA	13.46	-0.39		13.85	12.47	15.23
CHEVRON CORPORATION	CVX	67.89	-0.41		68.30	61.48	75.12
THE ALLSTATE CORPORATION	ALL	32.63	1.61		31.02	27.92	34.11
EXXON MOBIL CORPORATION	XOM	67.73	1.21		66.52	59.87	73.17
METLIFE INC.	MET	35.64	-0.09		35.73	32.16	39.30
J.C. PENNEY COMPANY INC.	JCP	32.66	-0.29		32.95	29.66	36.24
OFFICEMAX INCORPORATED	OMX	12.22	-0.19		12.41	11.17	13.65
AETNA INC.	AET	27.30	0.43		26.87	24.19	29.56
CONOCOPHILLIPS	COP	43.59	-3.03		46.62	41.96	51.27
UNITEDHEALTH GROUP INC.	UNH	24.43	0.07		24.36	21.93	26.79

Example: Financial Apps



Hong Kong: 23:36:27

FX Trader Application Demo

USDGBP [+]	GBPUSD [+]
USDEUR [+]	GBPEUR [+]
USDCAD [+]	GBPCAD [+]
USDAUD [+]	GBPAUD [+]
USDNZD [+]	GBPNZD [+]
USDCHF [+]	GBPCHF [+]
USDJPY [+]	GBPJPY [+]
USDHKD [+]	GBPHKD [+]
USDSGD [+]	GBPSGD [+]
USDILS [+]	GBPILS [+]
USDRUB [+]	GBPRUB [+]

Currency Pairs

USDGBP ⊗
0.4593 0.0000 Buy ▲ | 0.4426 0.0022 Sell ▲
Trade History

USDJPY ⊗
74.9977 0.0000 Buy ▲ | 59.0821 0.6877 Sell ▲
Trade History

USDEUR ⊗
0.7058 -0.0097 Buy ▼ | 0.6171 0.0000 Sell ▲
Trade History

USDILS ⊗
3.8735 0.0290 Buy ▲ | 2.0698 0.0000 Sell ▲
Trade History

GBPCHF ⊗
1.5872 0.0000 Buy ▲ | 0.6907 0.0024 Sell ▲
Trade History

GBPUSD ⊗
2.0084 0.0175 Buy ▲ | 1.7873 0.0000 Sell ▲
Trade History

GBPEUR ⊗
1.5727 0.0015 Buy ▲ | 0.8670 0.0000 Sell ▲
Trade History

GBPRUB ⊗
38.1429 -0.1558 Buy ▼ | 37.4126 0.0000 Sell ▲
Trade History

Spot Prices

Portfolio Valuations

USDGBP	USDJPY	USDEUR	USDILS
£ 0	¥ 0	€ 0	₪ 0
GBPCHF	GBPUSD	GBPEUR	GBPRUB
CHF 0	$ 0	€ 0	руб 0

Market Indices

DOW	15410	+63	+0.4%	FTSE 100	5840	-69	-1.2%	Nikkei	11431	+133	+1.2%
NASDAQ	1392	+51	+3.8%	DAX	4015	-99	-2.4%	Hang Seng	15582	-190	-1.2%
S&P 500	839	+0	+0.0%	CAC 40	2276	+7	+0.3%	Shanghai	2144	-27	-1.2%

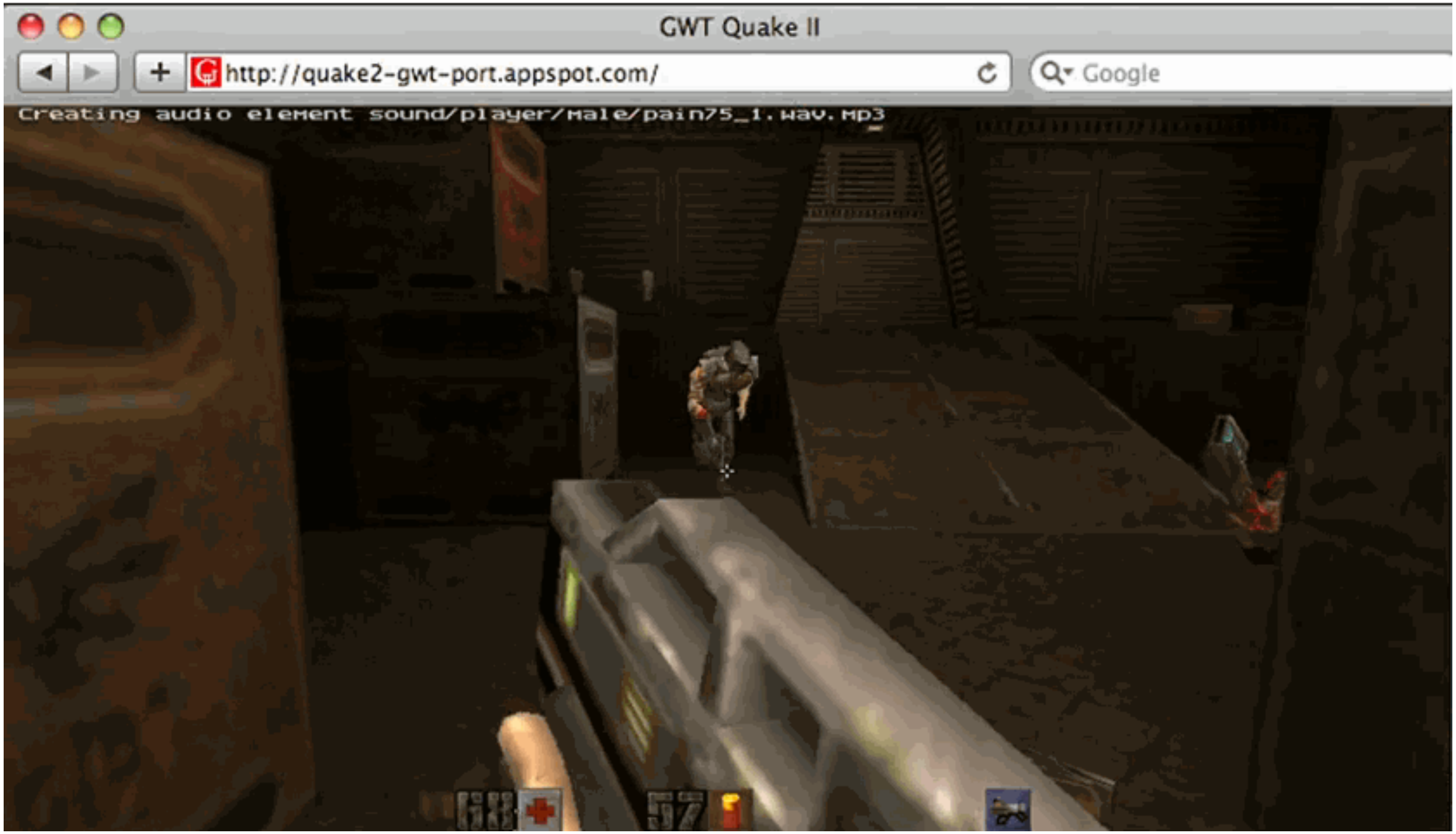
News Feed: New York Times

Wall Street's Long History of Protests [+]
Wall Street Goes to Defcon II on Swaps [+]
Congress Is Asked to Approve 3 Trade Pacts [+]
Rhapsody to Acquire Napster in Deal With Best Buy [+]
Diamondback Beset With Redemptions, Summer Losses [+]

Executions

no recent trades

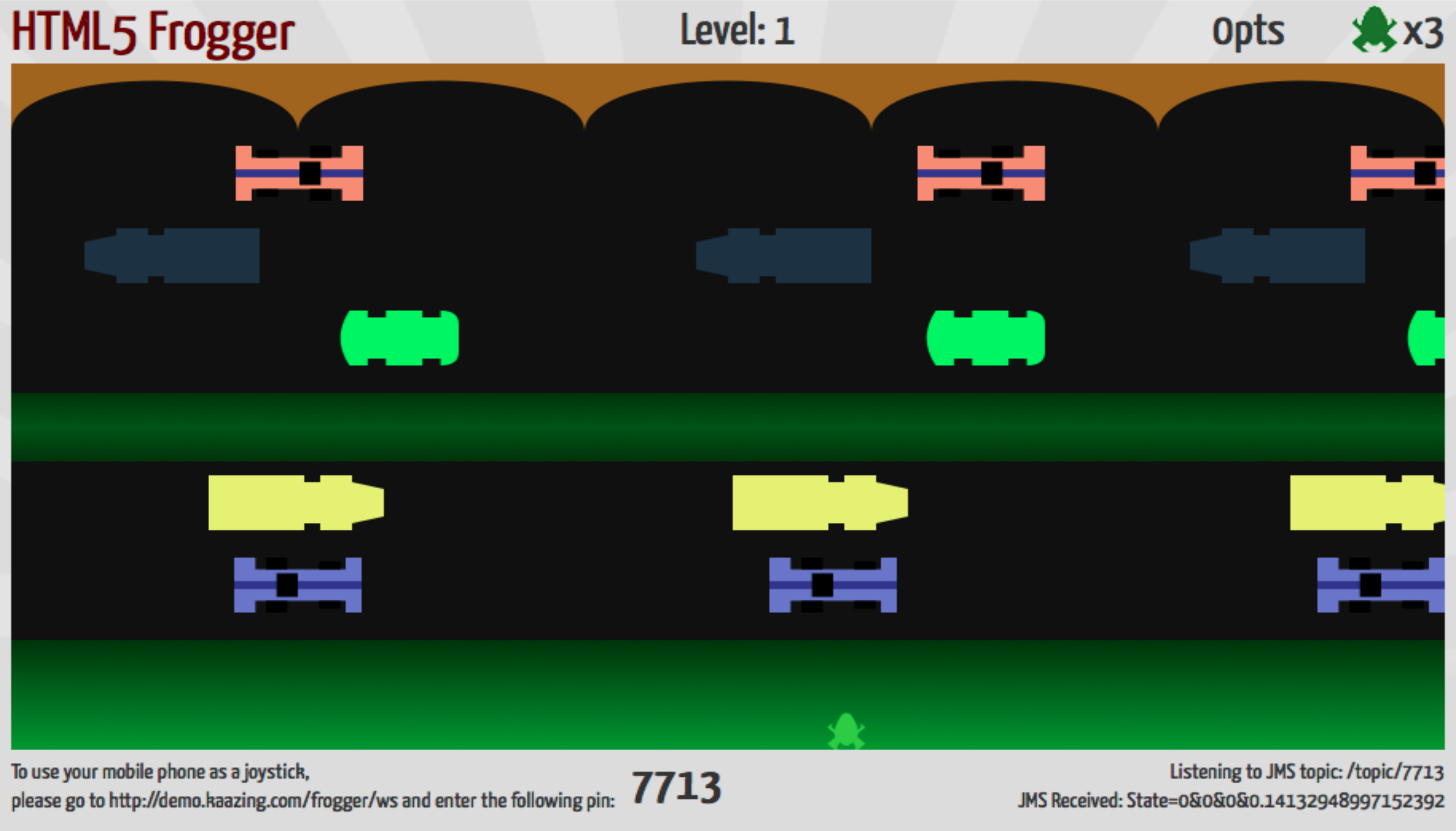
WebSocket-Based Quake II



```
Creating audio element sound/player/male/pain75_1.wav.mp3
```

http://code.google.com/p/quake2-gwt-port

Example: HTML5 Frogger



To use your mobile phone as a joystick,
please go to http://demo.kaazing.com/frogger/ws and enter the following pin: 7713

Listening to JMS topic: /topic/7713
JMS Received: State=0&0&0&0.14132948997152392

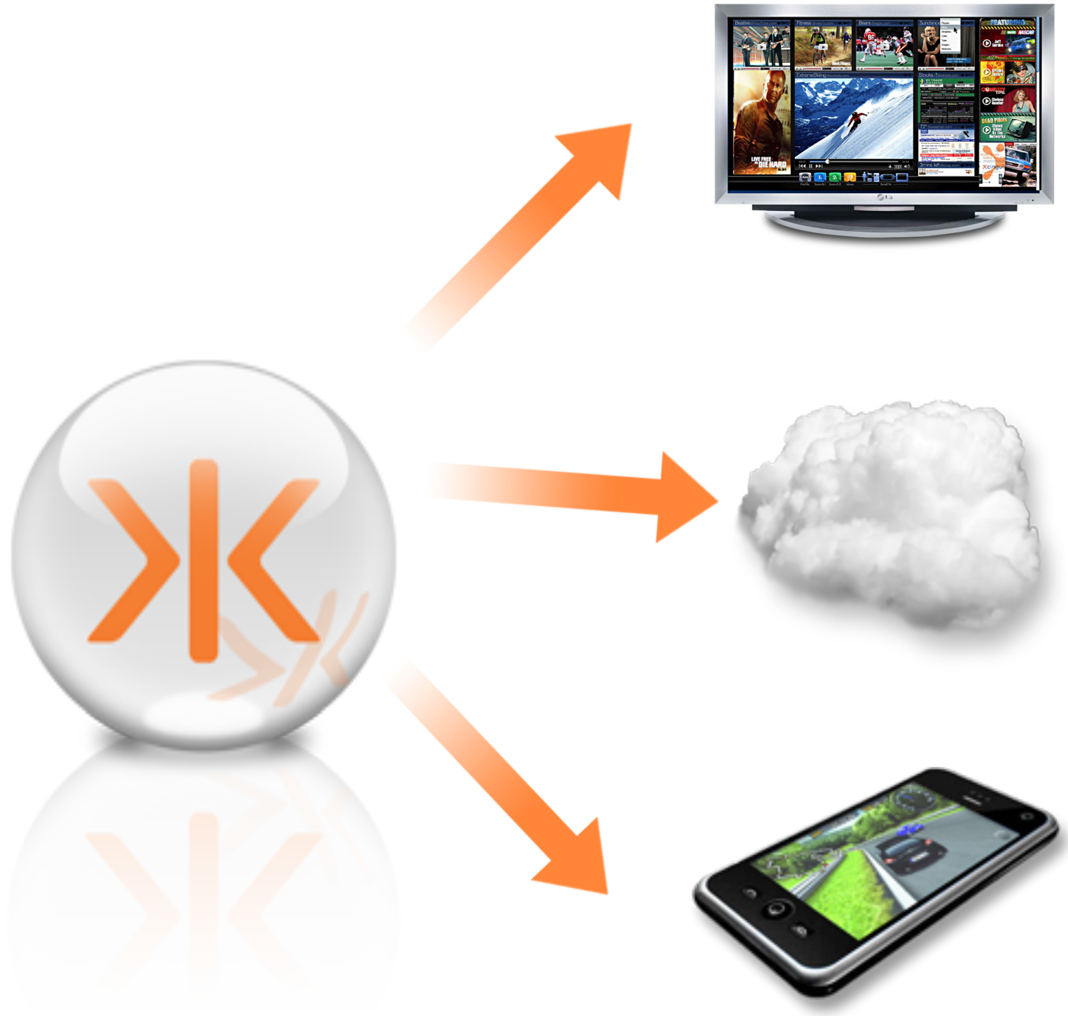
http://demo.kaazing.com/frogger

Possibilities…

- Low latency Financial and Trading apps
- Online in-game betting and live auctions
- Social networking
- Performance and monitoring dashboards
- RFID and GPS Tracking
- Sports and news broadcasting applications
- Supply chain and inventory management
- Smart meters
- Next generation web application of your choice!

Your cool [HTML5 WebSocket] App Here...

http://iseeaday.blogspot.com/



Unconstrained Web

- Financial Services
- Transportation and Logistics
- Telecommunications
- Utilities
- Social Networking

Cloud Computing

- Server to Server communication
- Distributed Internet applications over any TCP protocol
- Services on demand

3G & 4G Mobile Networking

- Significant bandwidth reduction
- New Service Delivery
- New Customer Experience

QUESTIONS
ANSWERS

KAAZING™