Using Websockets with ColdFusion

Raymond Camden | Developer Evangelist
Twitter: @cfjedimaster

Who am I?

- Raymond Camden
- Developer Evangelist for Adobe
- www.raymondcamden.com
- @cfjedimaster

What the heck is a Web Socket?

"WebSocket is a web technology providing for multiplexing bi-directional, full-duplex communications channels over a single TCP connection. "

-Wikipedia

WebSockets are a way to create JavaScript applications that have a true, open, connection to the server. This means they can receive updates instantly and broadcast to other clients.

Ok, dude, what?

```
setInterval(checkForStuff, 2000);
function checkForStuff() {
 someAjaxCallToTheServer();
}
```

Remember this guy?

The New Hotness

- Allow me to open a connection
- I'm told when new stuff is broadcast (server or other clients)
- I can broadcast my own stuff
- All kinds of fun filtering/organizing possible (think switchboard)

Communication via Phone

Your Browser is not worthy!

Web Sockets - **Working Draft**

Bidirectional communication technology for web apps

Resources: WebSockets information   Details on newer protocol   Wikipedia

*Usage stats:	Global
Support:	43.79%
Partial support:	7.55%
Total:	51.34%

	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Opera Mobile	Android Browser
		3.6						10.0	2.1
	6.0	9.0 Moz				3.2		11.0	2.2
	7.0	10.0 Moz				4.0-4.1		11.1	2.3
	8.0	11.0	17.0	5.0		4.2-4.3		11.5	3.0
Current	9.0	12.0	18.0	5.1	11.6	5.0	5.0-6.0	12.0	4.0
Near future	10.0	13.0	19.0	5.2	12.0				
Farther future		14.0	20.0						

Note: Partial support refers to the websockets implementation using an older version of the protocol and/or the implementation being disabled by default (due to security issues with the older protocol). Microsoft is currently experimenting with the technology.

Feedback

!websocket

- Fallback to Flash
- 100% of your code still works
- Support for showing a message to the poor saps left out…

The Details

- Client side and server side
- ColdFusion tags and JavaScript
- You will be writing JavaScript

Step One

- Application.cfc defines valid channels
- Channels are the most broad, most high level organization for WebSockets
- Defined as an array of structs

Example

```
this.wschannels = [{name="Sports"},
                   {name="Stocks"},
                   {name="Weather"}];
```

Step Two

- Your CFM defines a websocket via the new <cfwebsocket> tag
- Give it a name that sets up the JavaScript handle
- Tell it what to run when a message comes in
- Tell it what channel to connect to
- (There's more options)

Example

```
<cfwebsocket name="myWS"
onMessage="messageHandler"
subscribeTo="news">
```

Demo

- /example1

Using the JavaScript API

- publish - send a message (anything!)
- openConnection/closeConnection - pick up or hang up the phone
- subscribe/unsubscribe - connect (or disconnect) from a channel
- getSubcriptions - what I'm subscribed to
- getSubscriberCount - how many people are listening

More…

- authenticate - used for secured channels
- invoke and invokeAndPublish - used to communicate to a CFC

Demo

- /example2

Obligatory Chat Demo

CFC Handlers

- Give you server-side control over your websockets

- Must extend `CFIDE.websocket.ChannelListener`

- Some map to JavaScript functions

- Define the use of a handler in your Application.cfc

- Cached!!

Methods

- allowSubscribe - can I join the party?
- allowPublish - can I say something?
- beforePublish - format the message
- canSendMessage - can I hear something?
- beforeSendMessage - client specific formatting/modification
- afterUnsubscribe

Demo

- /example3

Server-Side Functions

- wsGetSubscribers(channel)
- wsPublish(channel, msg)
- wsGetAllChannels

Demo

- /example4

Filtering Options

- Multiple Channels
  - News, Weather, Sports, and Beer
- Manual processing
  - Messages can include custom data
- Subchannels
- Selectors

Subchannels

- You must subscribe to a channel defined in App.cfc, ala "news"

- But you can subscribe to a "dot path" under this: "news.sports"

- And go as far as you want: "news.sports.american.football"

Subchannels (2)

- You get messages for your subscription and "lower" nodes.

- Subscribed to news and you get news, news.sports, news.tech

- Subscribed to news.sports, you won't get news or news.tech

- Subscribed to news.sports, you will get news.sports.football

Demo

- /example5

Selectors

- Allow for more precise targeting
- Applies to publishing/receiving
- Selector is a basic conditional
- property <some comparison> value
- ColdFusion expressions, not JavaScript (no < or >)

Selector Example

- Ch: Products, Selector: price lt 100
- Ch: Stocks, Selector: change gt 10
- Ch: Scores, Selector: sport eq 'football'

Demo

- /example6

Security

Authentication

- Via onWSAuthenticate and JavaScript code (in other words, login with your fancy WebSocket app)

- Via an existing login, but cflogin only

onWSAuthenticate

- New method of Application.cfc
- Passed username, password, connectionInfo
  - connectionInfo is a struct you modify
  - must set .authenticated=true at minimum
- In JavaScript, use authenticate method
- Note! CFC handler MUST check in allowSubscribe

Demo

- /example7

SingleSignOn Mode

- Basically, this mode works with an *existing* authenticated user

- Must work along with cflogin-based security

Demo

- /example8

Security

- Remember - your WebSocket JavaScript variable is manipulable

- Show the console hack in chat

- This is all as secure as any other Ajax application

invokeAndPublish

- Used to run an adhoc CFC server to generate a message on a channel
- Usage:
  `myWS.invokeAndPublish("channel", "cfcname", "method", arrayOfArgs, structOfCustomHeaders)`
- Also runs `allowPublish()` in your CFC handler
- Cached

Demo

- /example9

Point2Point WebSocket

- Just you and the server (oh, how sweet)
- No channels involved - just a server CFC
- CFC can return messages, and make new ones via wsSendMessage
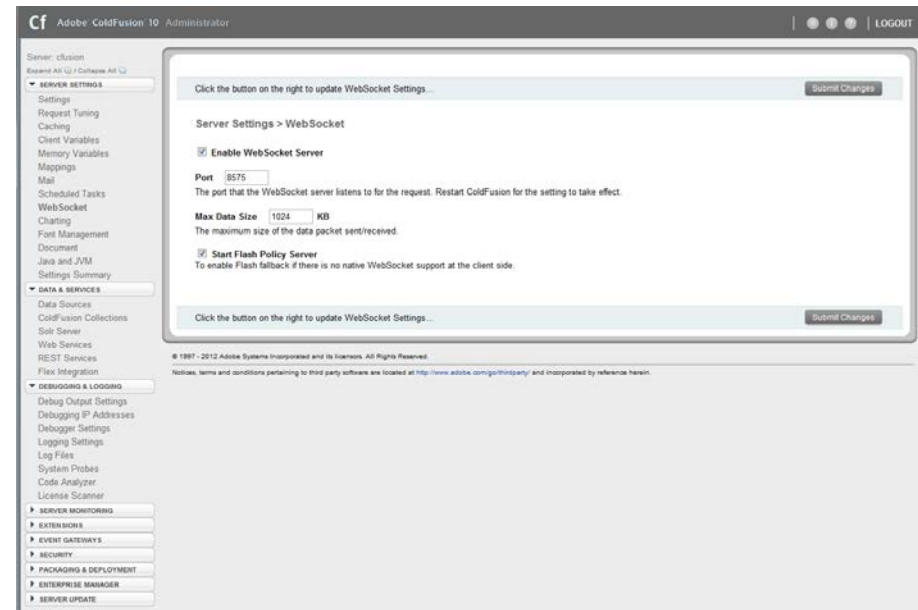
Demo

- /example10

Fallback

- If no websockets…
- Use Flash…
- If no Flash…
- messageHandler gets something
- or you can use onError

Demo

- /example11

CF Administrator Options

- Global enable/disable

- Set port and max data size

- Enable Flash fallback

Closing…

- Ajax dramatically improved the client experience
- This takes it even higher
- I'll be honest…