

# chloe and the RTW

---

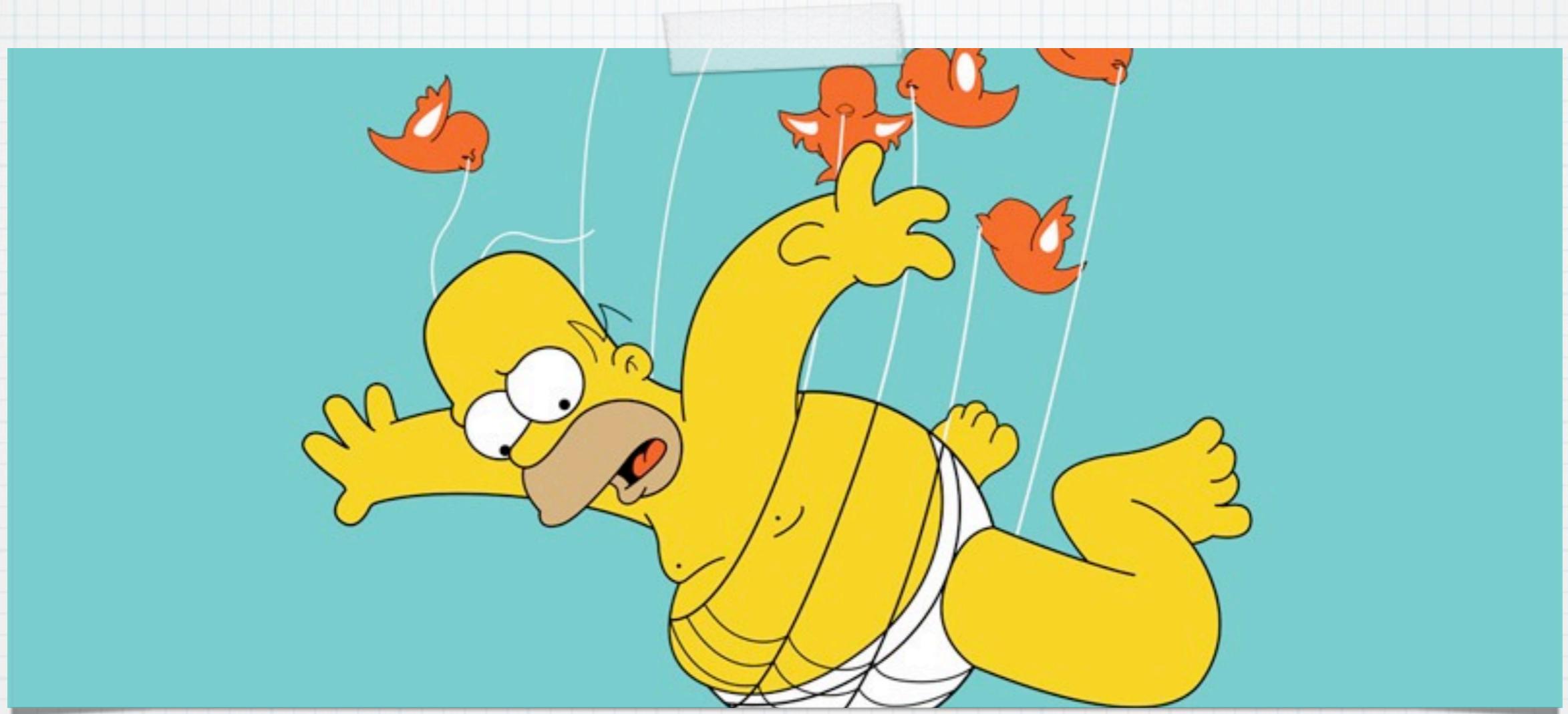
Trotter Cashion  
Strange Loop 2011

For those  
looking only  
at slides!



**MASHION**

Co-Founder



@cashion

[github.com/  
trotter](https://github.com/trotter)



# Go Get It!

---

<http://github.com/mashion/chloe>

Why?

The node.js logo is displayed on a dark grey rectangular background. The word "node" is in white lowercase letters, and ".js" is in yellow uppercase letters. A small white cloud-like shape is positioned between the "e" and the ".js".

node.js

Knockout!



# SIMULCHART



Real-time charts you can share and embed.

Create Your Own ➔

REST API



Simulchart



Give Us Data

Embed Script

Real-Time Chart

simulchart.com



# Knockout!

Made me cry :-)

# invisible race car



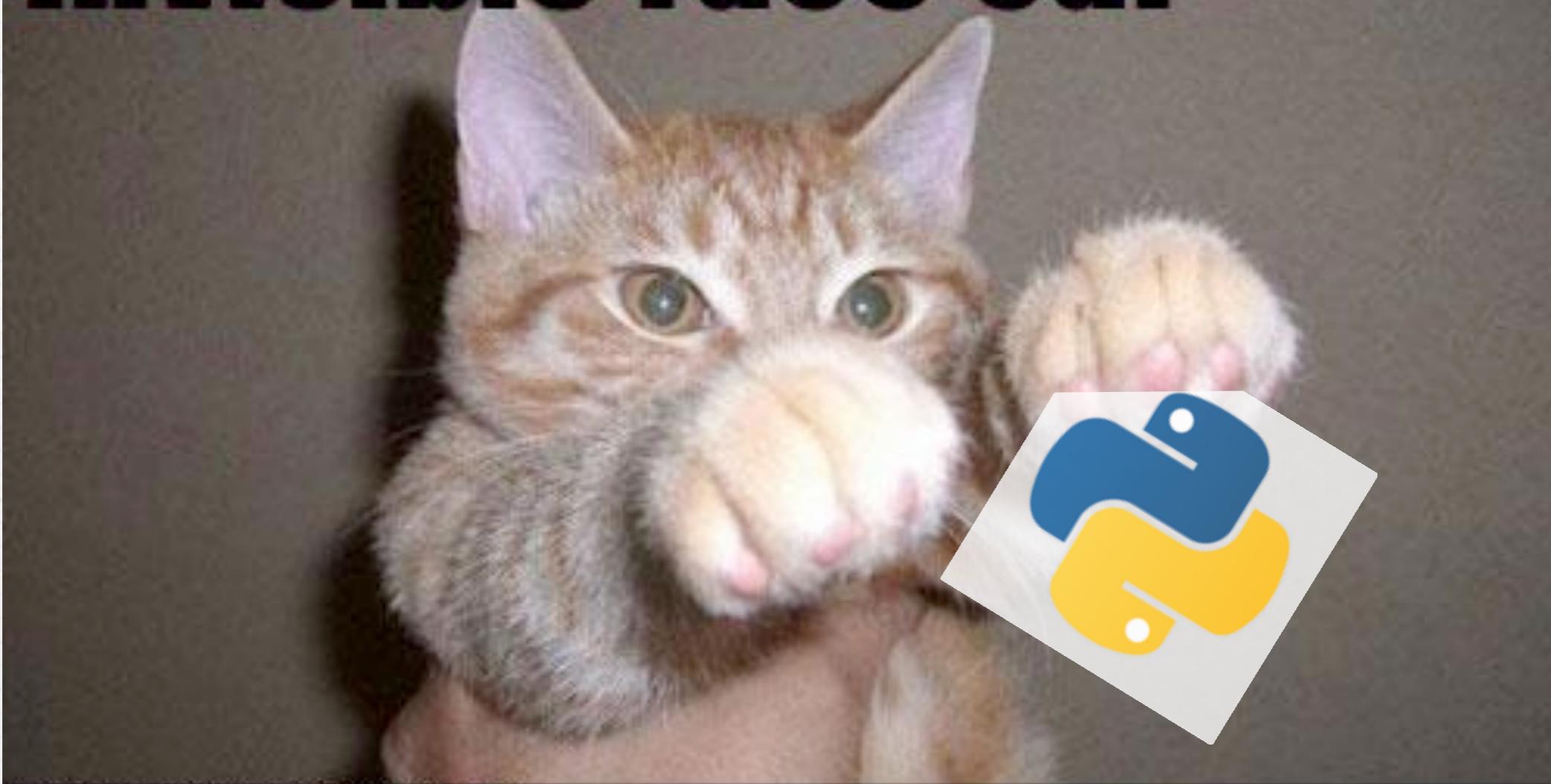
We can Haz?

**invisible race car**



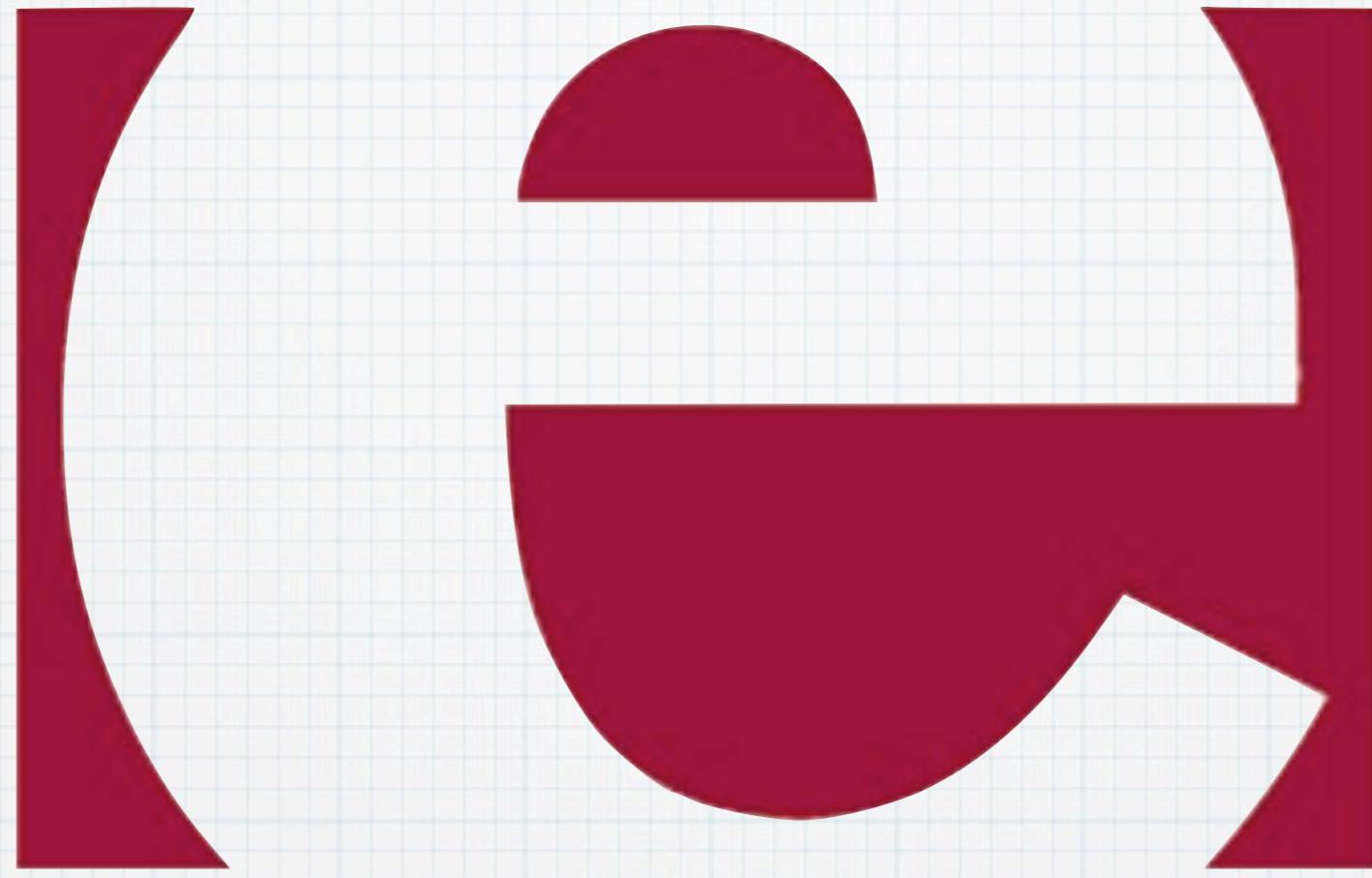
We can Haz?

**invisible race car**



We can Haz?

Got me thinking...



E R L A N G

Chloe

A realtime web-server... that doesn't suck.

# Cool History, Right?

---

<http://github.com/mashion/chloe>

# Usage...

---

## A Simple Chat Service

```
<html>
  <head>
    <script type="text/javascript" src="/jquery-1.6.4.min.js"></script>
    <script type="text/javascript" src="http://localhost:8901/chloe.js">
      </script>
    <script type="text/javascript" src="/chat.js"></script>
    <title>Ajax Chat</title>
  </head>
  <body>
    <form>
      <input name="message" id="message" type="text"/>
      <input name="submit" type="submit" value="Send"/>
    </form>
    <ul id="messages"></ul>
  </body>
</html>
```

```
<html>
  <head>
    <script type="text/javascript" src="/jquery-1.6.4.min.js"></script>
    <script type="text/javascript" src="http://localhost:8901/chloe.js">
      </script>
    <script type="text/javascript" src="/chat.js"></script>
    <title>Ajax Chat</title>
  </head>
  <body>
    <form>
      <input name="message" id="message" type="text"/>
      <input name="submit" type="submit" value="Send"/>
    </form>
    <ul id="messages"></ul>
  </body>
</html>
```



```
<html>
  <head>
    <script type="text/javascript" src="/jquery-1.6.4.min.js"></script>
    <script type="text/javascript" src="http://localhost:8901/chloe.js">
      </script>
    <script type="text/javascript" src="/chat.js"></script>
    <title>Ajax Chat</title>
  </head>
  <body>
    <form>
      <input name="message" type="text" />
      <input name="submit" type="submit" value="Send" />
    </form>
    <ul id="messages"></ul>
  </body>
</html>
```



```
$($function () {
  var form = $('form'),
      container = $('#messages');
  chloe = new Chloe({host: 'localhost', port: 8901});

  chloe.connect(function () {
    form.submit(function () {
      chloe.send(form.serialize());
      $('#message').val('');
      return false;
    });
  });

  chloe.onmessage(function (message) {
    container.prepend("<li>" + message + "</li>");
  });
});
```

```
$ (function () {
  var form = $('form'),
      container = $('#messages');
  chloe = new Chloe({host: 'localhost', port: 8901});

  chloe.connect(function () {
    form.submit(function () {
      chloe.send(form.serialize());
      $('#message').val('');
      return false;
    });
  });

  chloe.onmessage(function (message) {
    container.prepend("<li>" + message + "</li>");
  });
});
```

Instantiate Chloe

```
$function () {
  var form = $('form'),
      container = $('#messages');
  chloe = new Chloe({host: 'localhost', port: 8901});

  chloe.connect(function () {
    form.submit(function () {
      chloe.send('connect');
      $('#mess
      return false;
    });
  });

  chloe.onmessage(function (message) {
    container.prepend("<li>" + message + "</li>");
  });
});
```

**Connect to the Server**

```
$($.function () {
  var form = $('form'),
      container = $('#messages');
  chloe = new Chloe({host: 'localhost', port: 8901});

  chloe.connect(function () {
    form.submit(function () {
      chloe.send(form.serialize());
      $('#message').val('');
      return false;
    });
  });

  chloe.onmessage(function (message) {
    container.prepend("<li>" + message + "</li>");
  });
});
```

Send data to the server

```
$function () {
  var form = $('form'),
      container = $('#messages');
  chloe = new Chloe({host: 'localhost', port: 8901});

  chloe.connect(function () {
    form.submit(function () {
      chloe.send(form.serialize());
      $('#message').val('');
      return f
    });
  });

  chloe.onmessage(function (message) {
    container.prepend("<li>" + message + "</li>");
  });
}

});
```

**Run this function when a message arrives**

```
get '/' do
  erubis :index
end

post '/updates' do
  data = URI.decode_www_form(request.body.read)
  params = Hash[data]
  uri = URI.parse("http://localhost:8901/send")
  Net::HTTP.post_form(uri,
    {"data" => params["message"]})
  'ok'
end
```

```
get '/' do
  erubis :index
end
post '/up' do
  data = URI.decode_www_form(request.body.read)
  params = Hash[data]
  uri = URI.parse("http://localhost:8901/send")
  Net::HTTP.post_form(uri,
    {"data" => params["message"]})
  'ok'
end
```

**Get the html from earlier**



```
get '/' do
  erubis :index
end

chloe.send(form.serialize()));

post '/updates' do
  data = URI.decode_www_form(request.body.read)
  params = Hash[data]
  uri = URI.parse("http://localhost:8901/send")
  Net::HTTP.post_form(uri,
    {"data" => params["message"]})
  'ok'
end
```

```
get '/' do
  erubis :index
end

post '/updates' do
  data = URI.decode_www_form(request.body.read)
  params = Hash[data]
  uri = URI.parse("http://localhost:8901/send")
  Net::HTTP.post_form(uri,
    {"data" => params["message"]})
  'ok'
end
```

Ruby trick!

```
get '/' do
  erubis :index
end

post '/updates' do
  data = URI.decode_www_form(request.body.read)
  params = Hash[data]
  uri = URI.parse("http://localhost:8901/send")
  Net::HTTP.post_form(uri,
    'ok'          ↗
    "Chloe broadcast/multicast url"  ↗
    s[ "message" ] ) )
end
```

```
get '/' do
  erubis :index
end

post '/updates' do
  data = URI.parse  
Send the data to Chloe
  params = Hash[parse  
test.body.read)
  uri = URI.parse("http://localhost:8901/send")
  Net::HTTP.post_form(uri,
    {"data" => params["message"]})
  'ok'
end
```

# How It Works



Chrome

Chrome

Server

Chrome

Server \*

\*could be Java, Ruby, Scala, Clojure, Node...

Chrome

Server \*

Chloe

\*could be Java, Ruby, Scala, Clojure, Node...

Chrome

Server \*

Chloe

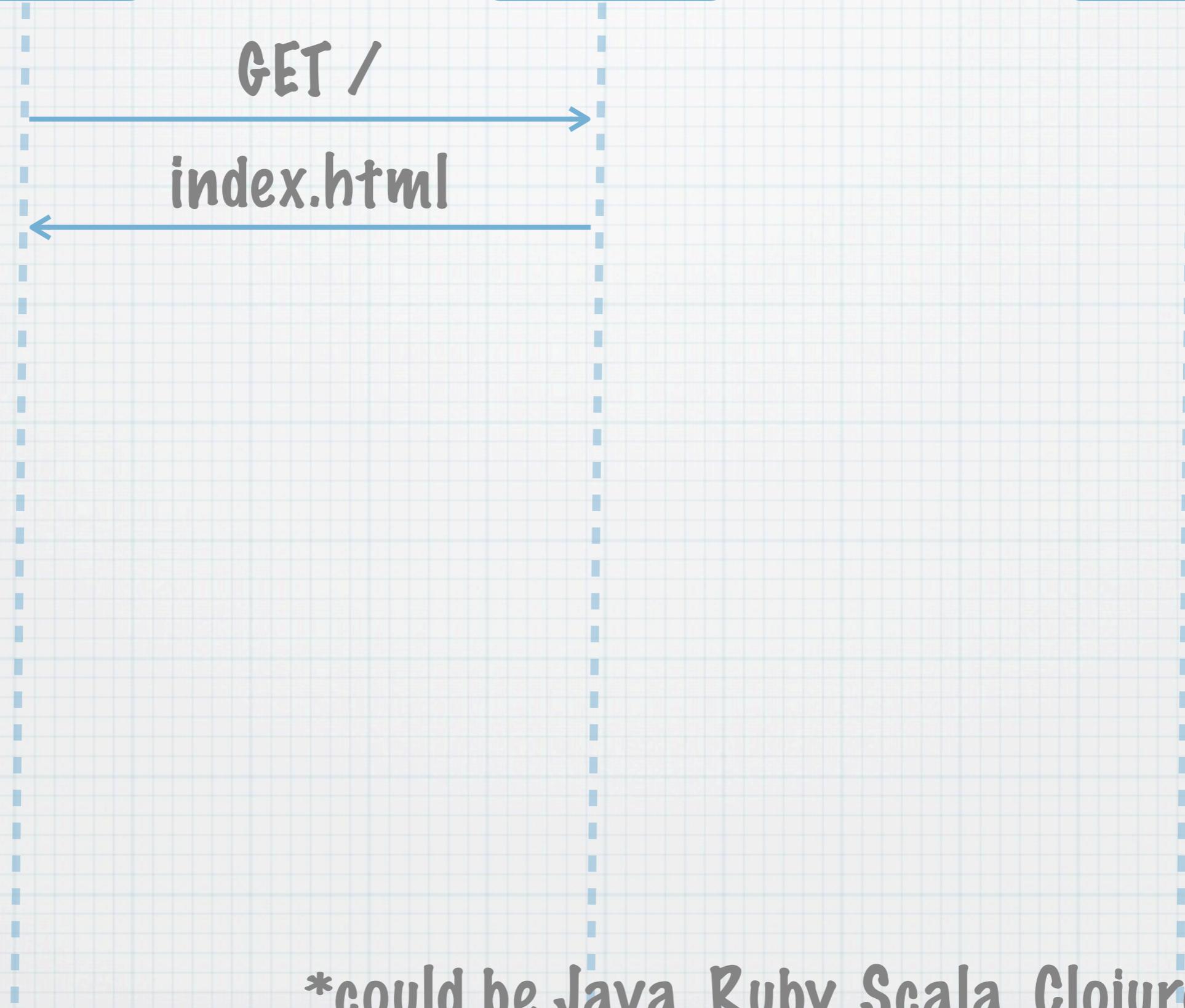
GET /

\*could be Java, Ruby, Scala, Clojure, Node...

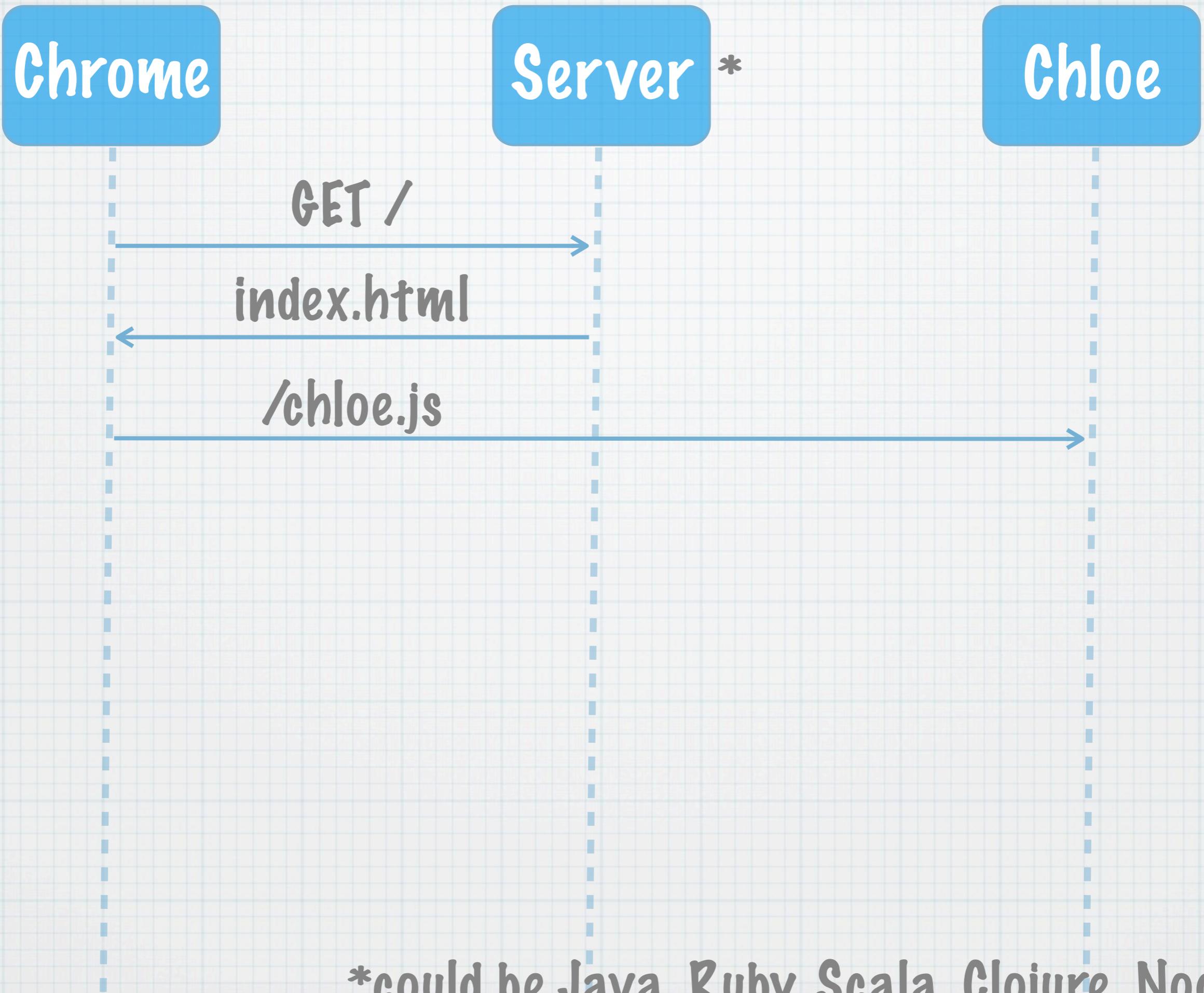
Chrome

Server \*

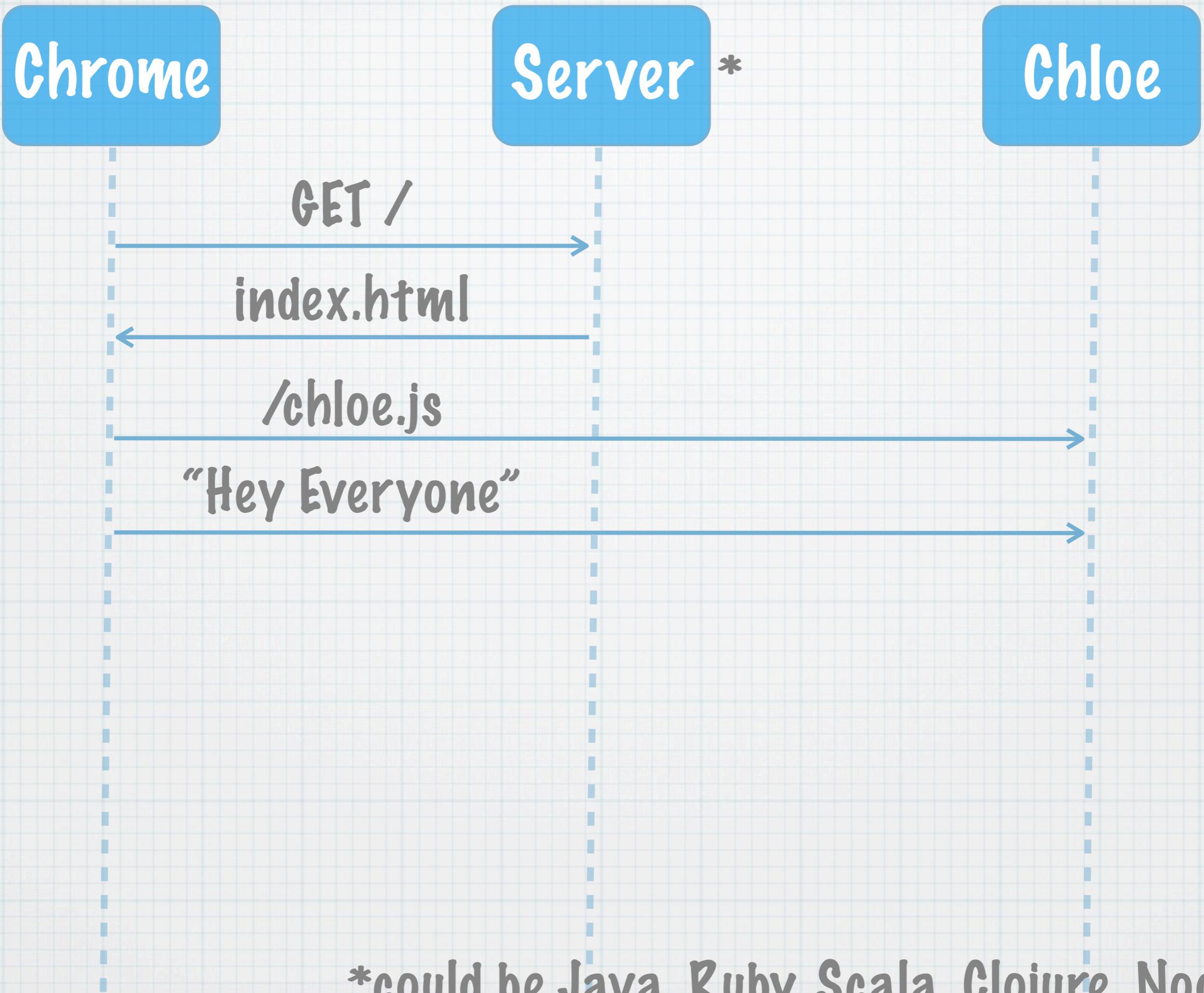
Chloe

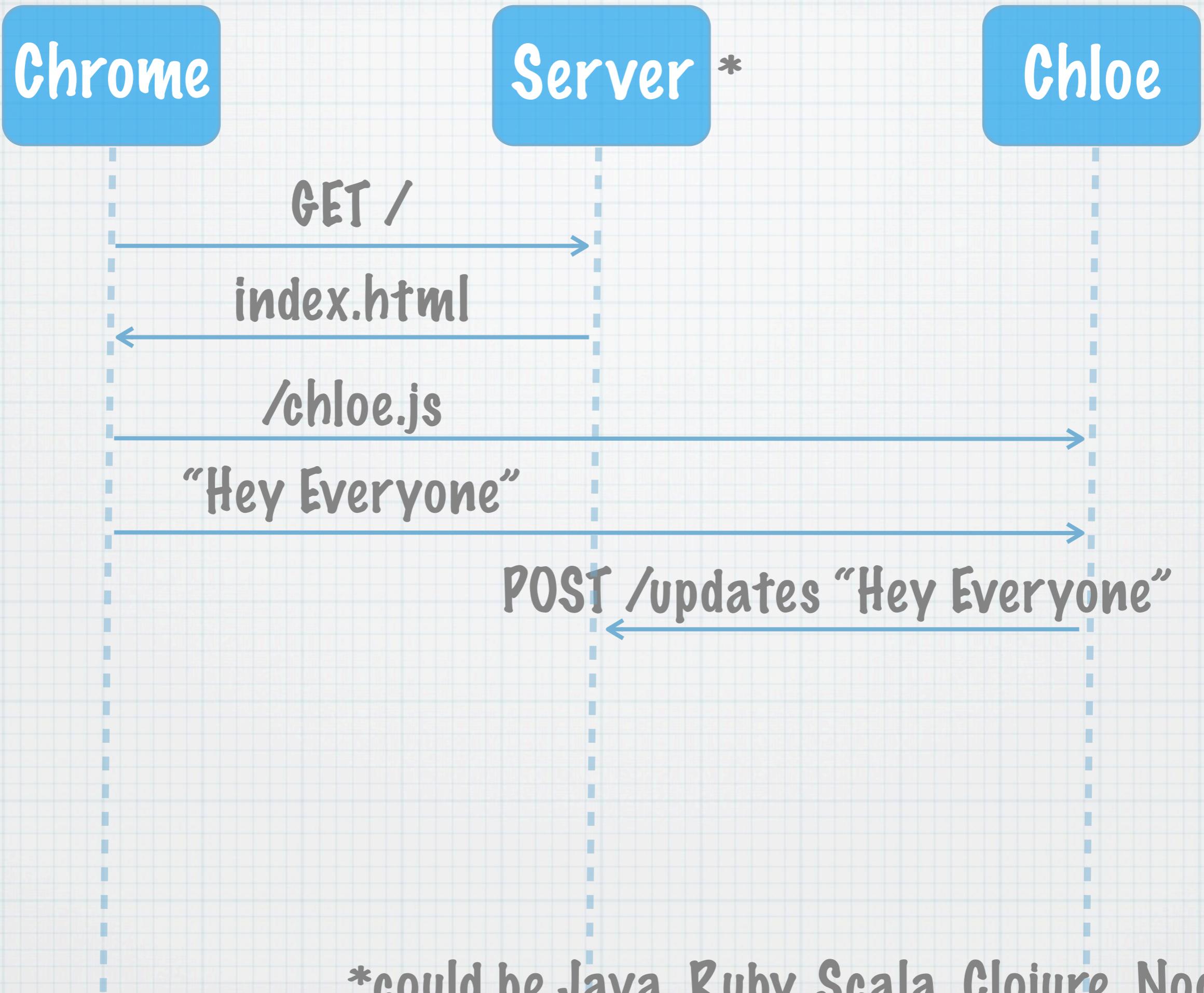


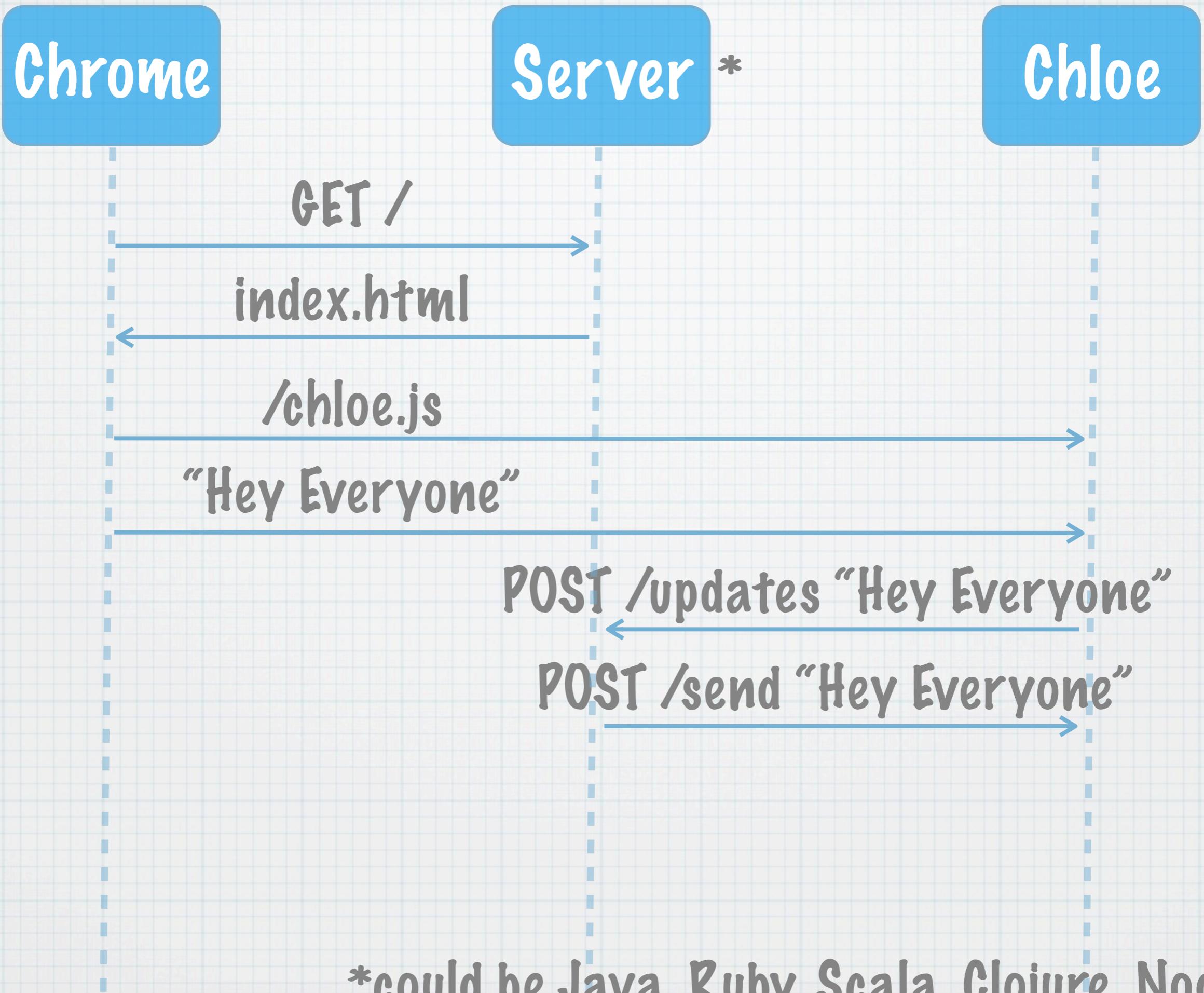
\*could be Java, Ruby, Scala, Clojure, Node...

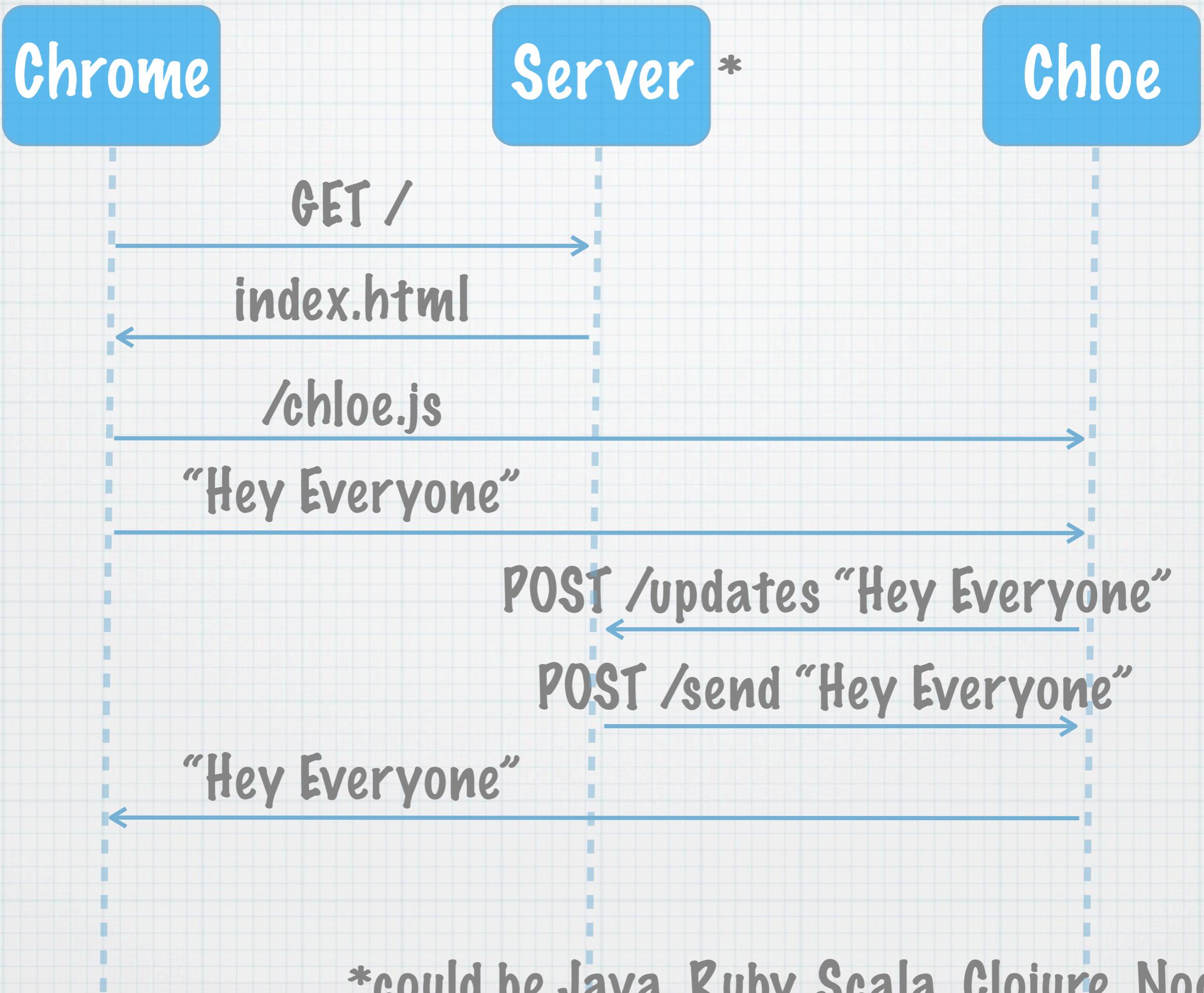


\*could be Java, Ruby, Scala, Clojure, Node...









# Fallbacks

- \* Websockets
- \* Long polling (jsonp)
- \* Cross domain xhr - SOON!
- \* This seems to be enough for now...

# Channels

```
chloe.connect(function () {
  chloe.subscribe('chat-room-5', function (message) {
    $('#messages').prepend("<li>" + message + "</li>")
  });
});
```

```
post '/updates' do
  uri = URI.parse("http://localhost:8901/send")
  Net::HTTP.post_form(uri,
    { "data" => "Hello room!",
      "channel" => "chat-room-5" })
  "ok"
end
```

Channel Name

```
chloe.connect(function () {
  chloe.subscribe('chat-room-5', function (message) {
    $('#messages').prepend("<li>" + message + "</li>")
  });
});
```

```
post '/updates' do
  uri = URI.parse("http://localhost:8901/send")
  Net::HTTP.post_form(uri,
    { "data" => "Hello room!",
      "channel" => "chat-room-5" })
  "ok"
end
```

```
chloe.connect(function () {  
  chloe.subscribe('chat-room-5', function (message) {  
    $('#messages').prepend("<li>" + message + "</li>")  
  });  
});
```



Called when message arrives

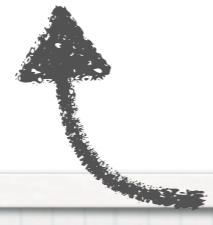
```
post '/updates' do  
  uri = URI.parse("http://localhost:8901/send")  
  Net::HTTP.post_form(uri,  
    { "data" => "Hello room!",  
      "channel" => "chat-room-5" })  
  "ok"  
end
```

```
chloe.connect(function () {
  chloe.subscribe('chat-room-5', function (message) {
    $('#messages').prepend("<li>" + message + "</li>")
  });
});
```

```
post '/updates' do
  uri = URI.parse("http://localhost:8901/send")
  Net::HTTP.post_form(uri,
    { "data" => "Hello room!",
      "channel" => "chat-room-5" })
  "ok"
end
```

```
chloe.connect(function () {  
  chloe.subscribe('chat-room-5', function (message) {  
    $('#messages').prepend("<li>" + message + "</li>")  
  });  
});
```

```
post '/updates' do  
  uri = URI.parse("http://localhost:8901/send")  
  Net::HTTP.post_form(uri,  
    { "data" => "Hello room!",  
      "channel" => "chat-room-5" })  
  "ok"  
end
```



Channel for broadcast

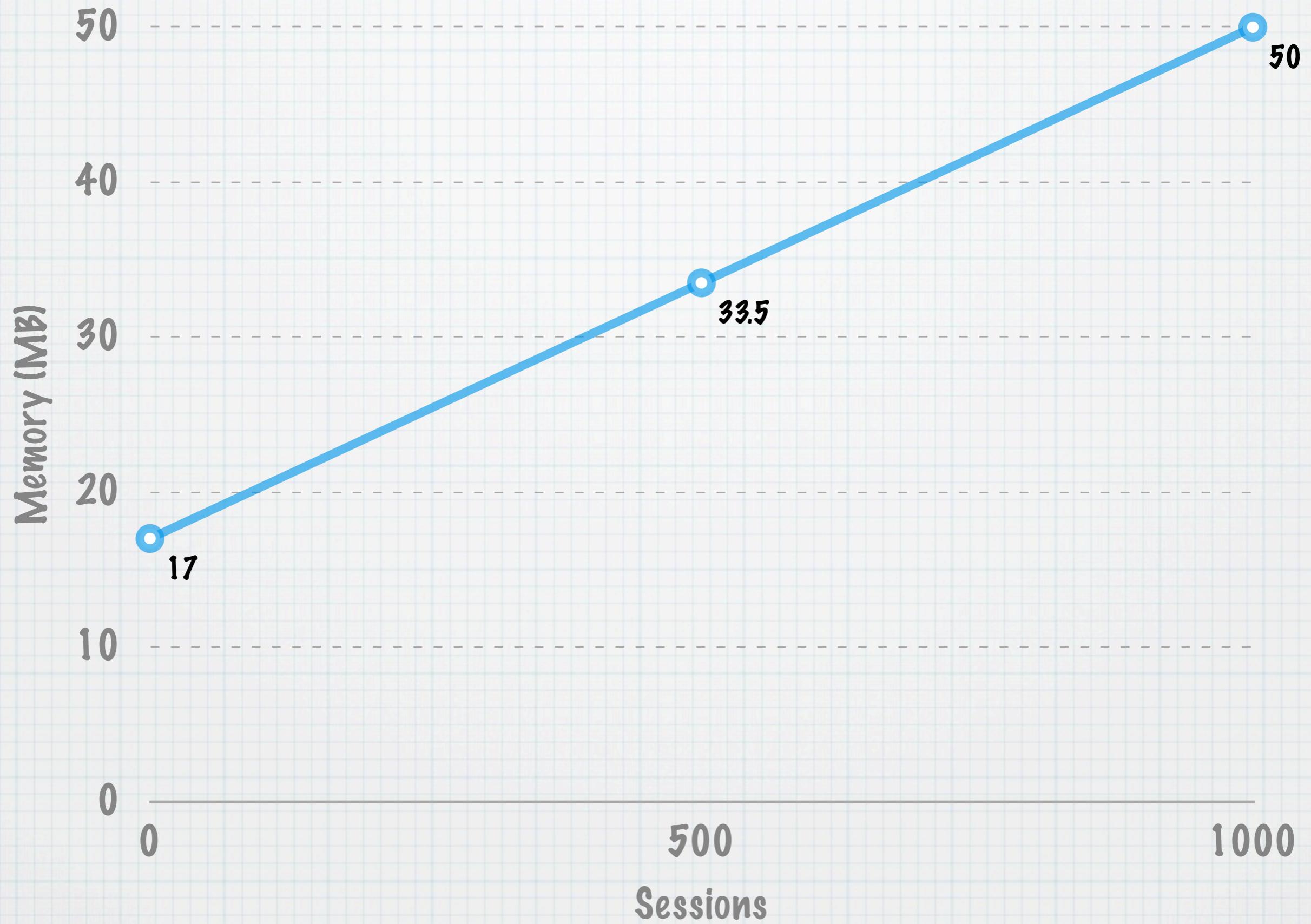
# What's left for Channels

- \* Bi-directionality
- \* Per Channel callbacks

# Performance

---

Difficult to determine...



# Tune Your TCP Stack

```
# ulimit -n 65536
# ifconfig eth0 txqueuelen 8192
# /sbin/sysctl -w net.core.somaxconn=4096
# /sbin/sysctl -w net.core.netdev_max_backlog=16384
# /sbin/sysctl -w net.core.rmem_max=16777216
# /sbin/sysctl -w net.core.wmem_max=16777216
# /sbin/sysctl -w net.ipv4.tcp_max_syn_backlog=8192
# /sbin/sysctl -w net.ipv4.tcp_syncookies=1
```

# Tune Your TCP Stack

```
# ulimit -n 65536
# ifconfig eth0 txqueuelen 8192
# /sbin/sysctl -w net.core.rmem_max=16777216
# /sbin/sysctl -w net.core.wmem_max=16777216
# /sbin/sysctl -w net.ipv4.tcp_max_syn_backlog=8192
# /sbin/sysctl -w net.ipv4.tcp_syncookies=1
```

**Increase number of file descriptors** = 16384

# Tune Your TCP Stack

```
# ulimit -n 65536
# ifconfig eth0 txqueuelen 8192
# /sbin/sysctl -w net.core.somaxconn=4096
# /sbin/sysctl -w net.core.rmem_max=16384
# /sbin/sysctl -w net.core.wmem_max=16777216
# /sbin/sysctl -w net.ipv4.tcp_max_syn_backlog=8192
# /sbin/sysctl -w net.ipv4.tcp_syncookies=1
```

# Tune Your TCP Stack

```
# ulimit -n 65536
# ifconfig eth0 txqueuelen 8192
# /sbin/sysctl -w net.core.somaxconn=4096
# /sbin/sysctl -w net.core.netdev_max_backlog=16384
# /sbin/sysctl -w net.core.backlog=16
# /sbin/sysctl -w net.core.somaxconn=16
# /sbin/sysctl -w net.ipv4.tcp_max_syn_backlog=8192
# /sbin/sysctl -w net.ipv4.tcp_syncookies=1
```

Increase size of listen queue

# Tune Your TCP Stack

```
# ulimit -n 65536
# ifconfig eth0 txqueuelen 8192
# /sbin/sysctl -w net.core.somaxconn=4096
# /sbin/sysctl -w net.core.netdev_max_backlog=16384
# /sbin/sysctl -w net.core.rmem_max=16777216
# /sbin/sysctl -w net.core.wmem_max=16777216
# /sbin/sysctl -w net.core.somaxconn=4096
# /sbin/sysctl -w net.core.netdev_max_backlog=8192
# /sbin/sysctl -w net.ipv4.tcp_syncookies=1
```

Increase incoming packet queue

# Tune Your TCP Stack

```
# ulimit -n 65536
# ifconfig eth0 txqueuelen 8192
# /sbin/sysctl -w net.core.somaxconn=4096
# /sbin/sysctl -w net.core.netdev_max_backlog=16384
# /sbin/sysctl -w net.core.rmem_max=16777216
# /sbin/sysctl -w net.core.wmem_max=16777216
# /sbin/sysctl -w net.core.rmem_min=192
# /sbin/sysctl -w net.core.wmem_min=192
```

Increase maximum receive window

# Tune Your TCP Stack

```
# ulimit -n 65536
# ifconfig eth0 txqueuelen 8192
# /sbin/sysctl -w net.core.somaxconn=4096
# /sbin/sysctl -w net.core.netdev_max_backlog=16384
# /sbin/sysctl -w net.core.rmem_max=16777216
# /sbin/sysctl -w net.core.wmem_max=16777216
# /sbin/sysctl -w net.ipv4.tcp_max_syn_backlog=8192
# /sbin/sysctl -w net.core.wmem_kbytes=kies=1
```

Increase send window

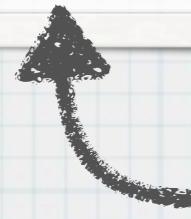
# Tune Your TCP Stack

```
# ulimit -n 65536
# ifconfig eth0 txqueuelen 8192
# /sbin/sysctl -w net.core.somaxconn=4096
# /sbin/sysctl -w net.core.netdev_max_backlog=16384
# /sbin/sysctl -w net.core.rmem_max=16777216
# /sbin/sysctl -w net.core.wmem_max=16777216
# /sbin/sysctl -w net.ipv4.tcp_max_syn_backlog=8192
# /sbin/sysctl -w net.ipv4.tcp_syncookies=1
```

How many SYN requests to keep in memory

# Tune Your TCP Stack

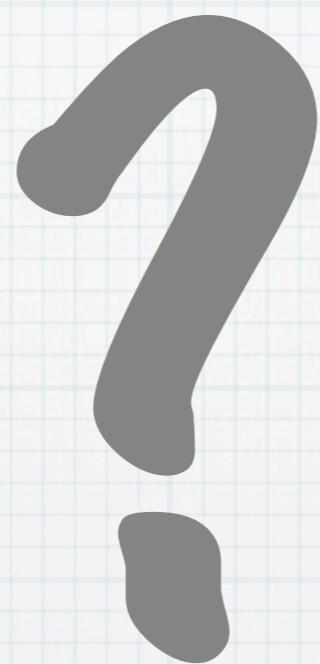
```
# ulimit -n 65536
# ifconfig eth0 txqueuelen 8192
# /sbin/sysctl -w net.core.somaxconn=4096
# /sbin/sysctl -w net.core.netdev_max_backlog=16384
# /sbin/sysctl -w net.core.rmem_max=16777216
# /sbin/sysctl -w net.core.wmem_max=16777216
# /sbin/sysctl -w net.ipv4.tcp_max_syn_backlog=8192
# /sbin/sysctl -w net.ipv4.tcp_syncookies=1
```



Makes `tcp_max_syn_backlog` work

# How much Overhead?

# How much Overhead?



Up and Running

# Binaries Available

- \* Mac, Ubuntu 32bit, Ubuntu 64bit
- \* Fully self-contained (Erlang included)
- \* .deb and .rpm will be coming soon...

# It's Easy!

```
# wget http://bit.ly/oqyQfL
# tar xzvf chloe-0.0.3-osx.tgz
# cd chloe-0.0.3
# ./bin/chloe start
# ./bin/chloe stop
```

# It's Easy!

```
# wget http://bit.ly/oqyQfL
# tar xzvf chloe-0.0.3-osx.tgz
# cd chloe-0.0.3
# ./bin/chloe start
# ./bin/chloe stop
```

# It's Easy!

```
# wget http://bit.ly/oqyQfL
# tar xzvf chloe-0.0.3-osx.tgz
# cd chloe-0.0.3
# ./bin/chloe start
# ./bin/chloe stop
```

# It's Easy!

```
# wget http://bit.ly/oqyQfL
# tar xzvf chloe-0.0.3-osx.tgz
# cd chloe-0.0.3
# ./bin/chloe start
# ./bin/chloe stop
```

# It's Easy!

```
# wget http://bit.ly/oqyQfL
# tar xzvf chloe-0.0.3-osx.tgz
# cd chloe-0.0.3
# ./bin/chloe start
# ./bin/chloe stop
```

# ./etc/app.config

```
{chloe,  
 [  
 {application_server, "http://localhost:4567"},  
 {application_server_url, "http://localhost:4567/updates"},  
 {port, 8901},  
 {doc_root, "./public"},  
 {log_dir, "/var/log/chloe"},  
 {secret, "SEKRET PASSFRASE"}  
]}
```

# ./etc/app.config

```
{chloe,  
 [  
  {application_server, "http://localhost:4567"},  
  {application_server_url, "http://localhost:4567/updates"},  
  {port, 8901},  
  {doc_root, "./public"},  
  {log_dir, "/var/log/chloe"},  
  {secret, "SEKRET PASSFRASE"}  
 ]}
```

**For XHR**



# ./etc/app.config

```
{chloe,  
 [  
   {application_server, "http://localhost:4567"},  
   {application_server_url, "http://localhost:4567/updates"},  
   {port, 8901},  
   {doc_root, "./public"},  
   {log_dir, "/var/log/chloe"},  
   {secret, "SEKRET PASSFRASE"}  
 ]}
```

**Callback url on your application**

# ./etc/app.config

```
{chloe,
[
  {application_server_url, "http://localhost:4567"}://localhost:4567"},  

  {application_server_url, "http://localhost:4567/updates"},  

  {port, 8901},  

  {doc_root, "./public"},  

  {log_dir, "/var/log/chloe"},  

  {secret, "SEKRET PASSFRASE"}  
]}
```

# ./etc/app.config

```
{chloe,  
 [  
  {application_server, },  
  {application_server_u  
   {port, 8901},  
   {doc_root, "./public"},  
   {log_dir, "/var/log/chloe"},  
   {secret, "SEKRET PASSFRASE"}  
 ] }
```

Where Chloe's JavaScript Lives

# ./etc/app.config

```
{chloe,  
 [  
 {application_server, "http://localhost:4567"},  
 {application_server_url, "http://localhost:4567/updates"},  
 {port, 8901},  
 {doc_root, "./public"},  
 {log_dir, "/var/log/chloe"},  
 {secret, "SEKRET PASSFRASE"}  
 ]}
```

Directory for Logs

# ./etc/app.config

```
{chloe,  
 [  
 {application_server, "http://localhost:4567"},  
 {application_server_url, "http://localhost:4567/updates"},  
 {port, 8901},  
 {doc_root, "./public"},  
 {log_dir, "/var/log/chloe"},  
 {secret, "SEKRET PASSFRASE"}  
]}
```



For security (not required)...

# Signing Requests

```
SECRET = "SEKRET PASSFRASE"
```

```
post '/updates' do
  data = URI.decode_www_form(request.body.read)
  message = Hash[data]["message"]
  sig = Digest::MD5.hexdigest(message + SECRET)
  uri = URI.parse("http://localhost:8901/send")
  Net::HTTP.post_form(uri,
    {"data" => message, "sig" => sig})
  'ok'
end
```

# Signing Requests

```
SECRET = "SEKRET PASSFRASE"
```

```
post '/updates' do
  data = URI.decode_www_form(request.body.read)
  message = Hash[data]["message"]
  sig = Digest::MD5.hexdigest(message + SECRET)
  uri = URI.parse("http://localhost:8901/send")
  Net::HTTP.post_form(uri,
    {"data" => message, "sig" => sig})
  'ok'
end
```

# Signing Requests

```
SECRET = "SEKRET PASSFRASE"
```

```
post '/update'
  data = URI.parse(From ./etc/app.config).quest.body.read)
  message = Hash[data]["message"]
  sig = Digest::MD5.hexdigest(message + SECRET)
  uri = URI.parse("http://localhost:8901/send")
  Net::HTTP.post_form(uri,
    {"data" => message, "sig" => sig})
  'ok'
end
```



# Signing Requests

```
SECRET = "SEKRET PASSFRASE"
```

```
post '/updates' do
  data = URI.decode(Net::HTTP.get_response(uri).body.read)
  message = Hash[data]["message"]
  sig = Digest::MD5.hexdigest(message + SECRET)
  uri = URI.parse("http://localhost:8901/send")
  Net::HTTP.post_form(uri,
    {"data" => message, "sig" => sig})
  'ok'
end
```

**Concat two strings**



# Signing Requests

```
SECRET = "SEKRET PASSFRASE"
```

```
post '/updates' do
  data = URI.decode www.uest.body.read)
  message = Hash[data] ["message"]
  sig = Digest::MD5.hexdigest(message + SECRET)
  uri = URI.parse("http://localhost:8901/send")
  Net::HTTP.post_form(uri,
    { "data" => message, "sig" => sig})
  'ok'
end
```

**Hash them**



# Signing Requests

```
SECRET = "SEKRET PASSFRASE"
```

```
post '/updates' do
  data = URI.decode_www_form(request.body.read)
  message = Hash[data]["message"]
  sig = Digest::MD5.hexdigest(message + SECRET)
  uri = URI.parse("http://localhost:8901/send")
  Net::HTTP.post_form(uri,
    { "data" => message, "sig" => sig})
  'ok'
end
```

Add signature to request



If `secret` is in your config, all requests to `/send` must be signed

# Coming Soon

- \* Improved Performance
- \* Bi-directional channels
- \* Per-channel callback urls
- \* Runtime configuration
- \* SSL Support

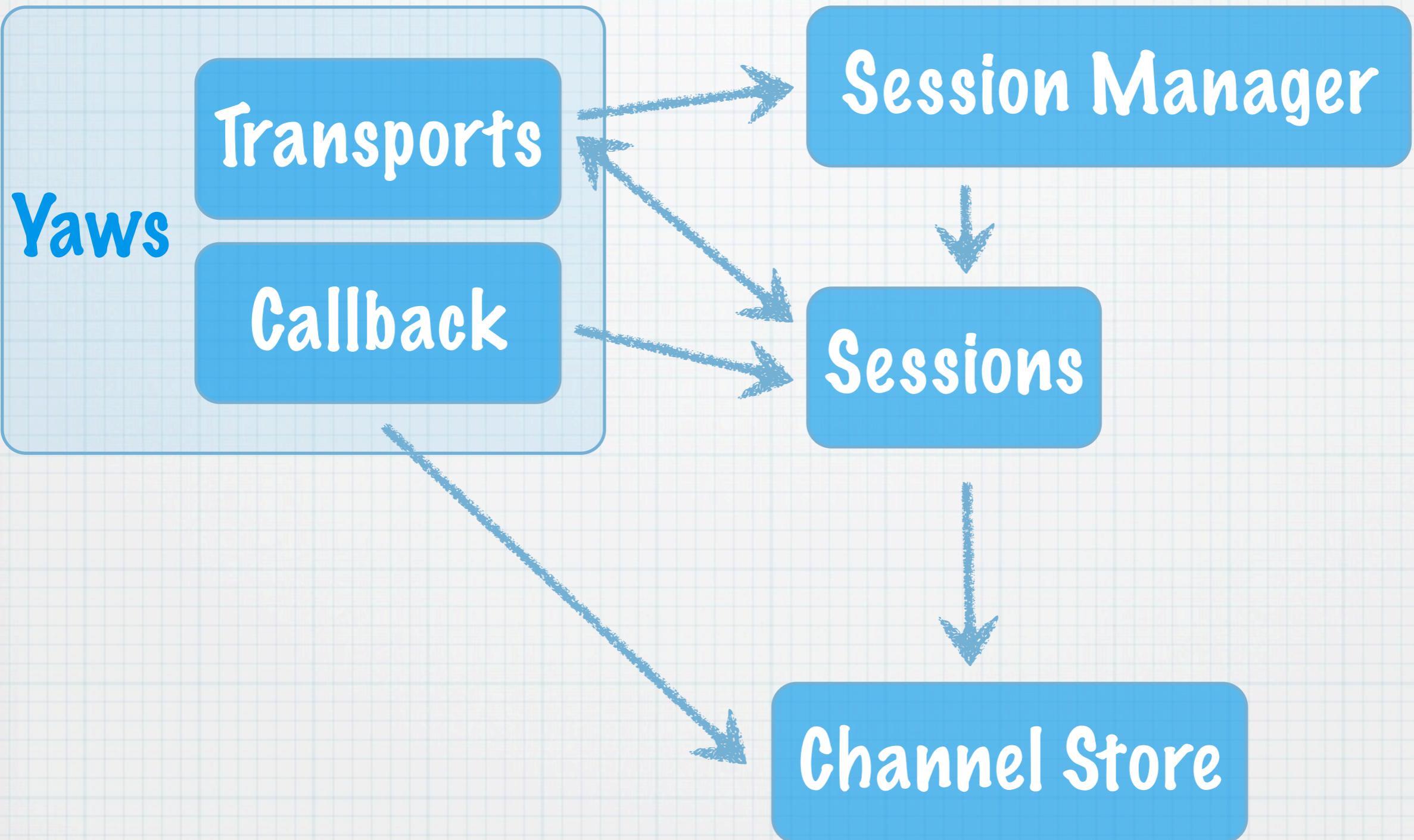
# Coming Later

- \* Improved monitoring
- \* Explore switch to mochiweb internally
- \* Client side authentication

# Internals

---

Erlang Nerdery Time!



**Yaws**

Transports

Callback

Session Manager

Sessions

Channel Store

# Session Manager

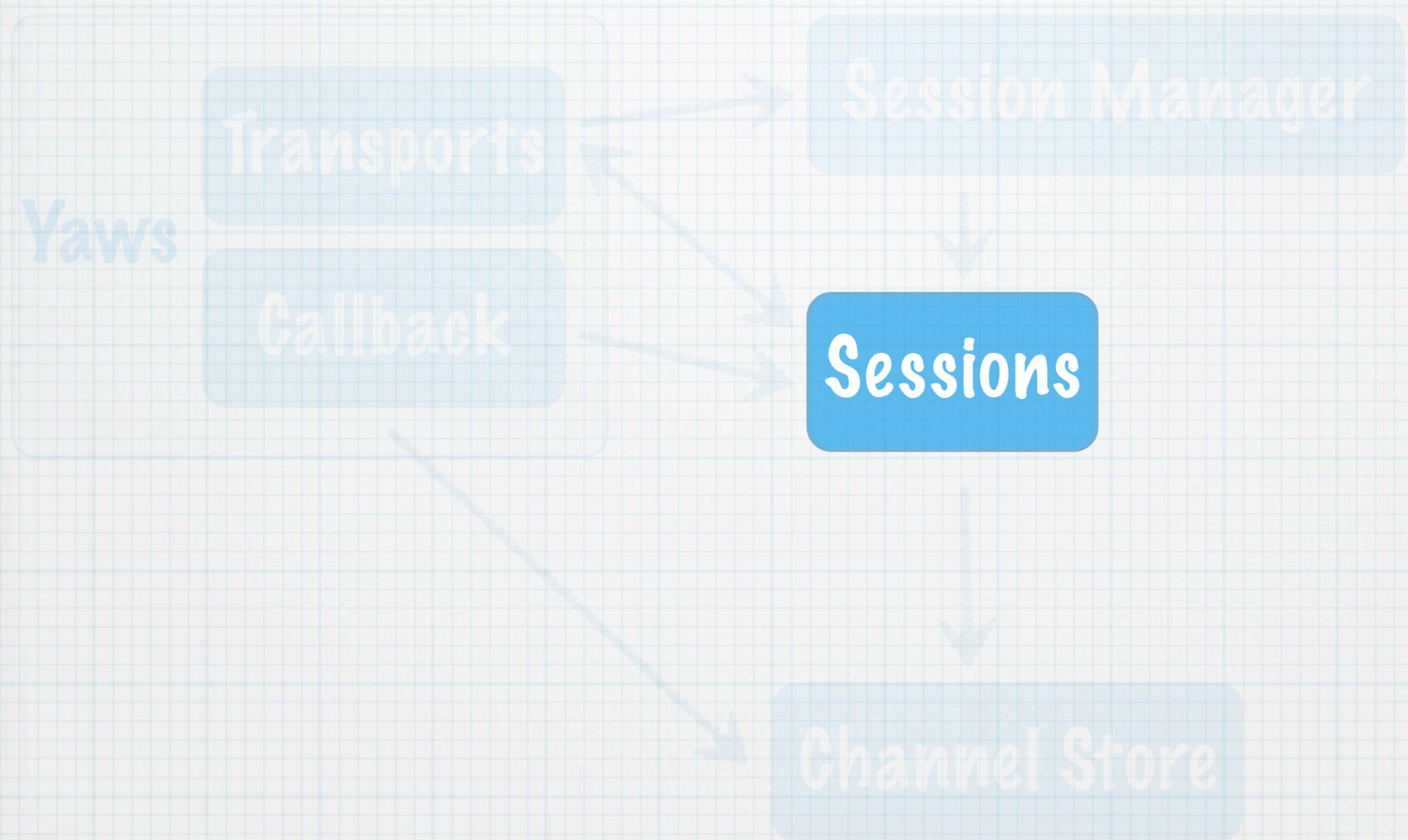
Transports

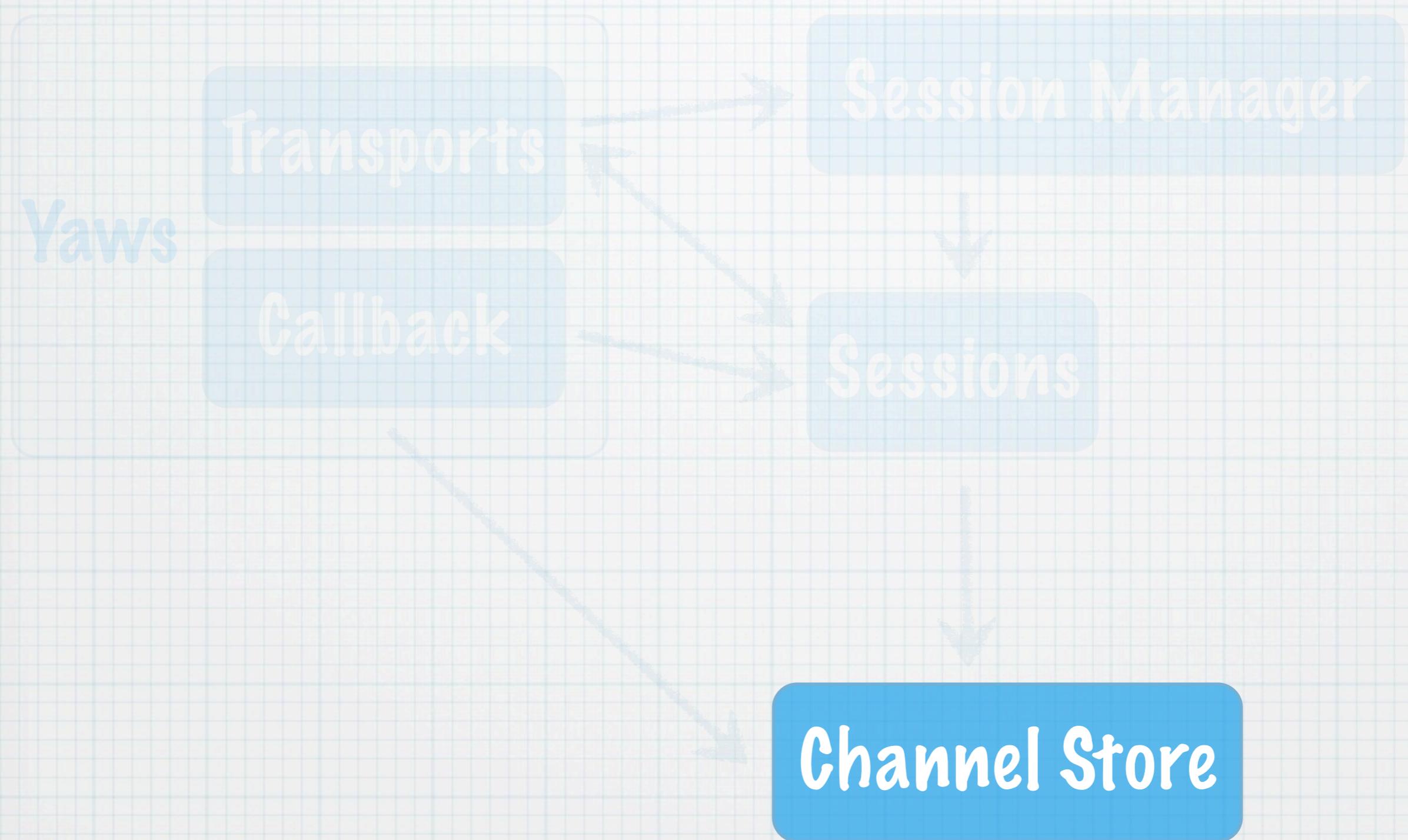
Yaws

Callback

Sessions

Channel Store





**Yaws**

**Websocket**

**Session  
Manager**

**Session**



**Yaws**

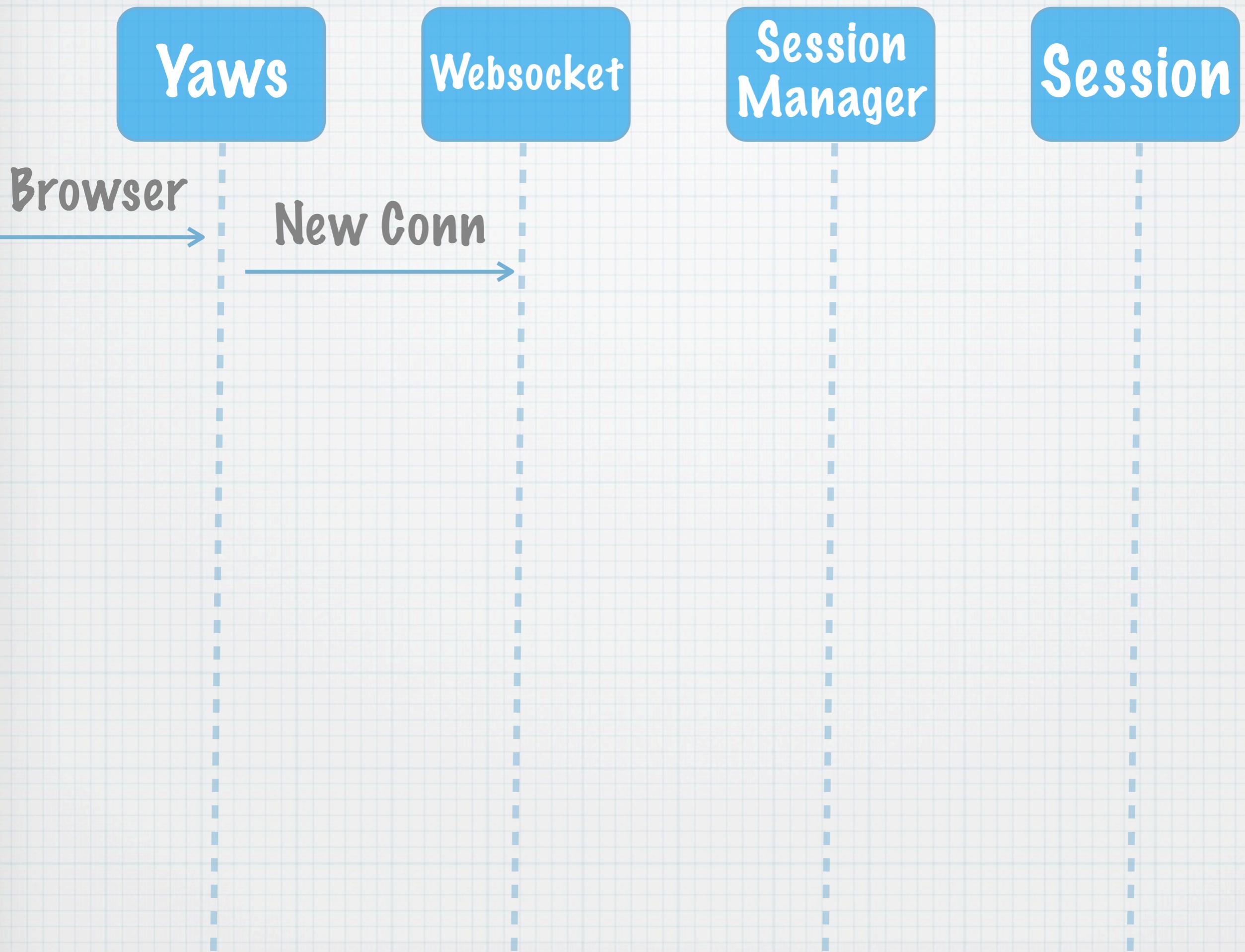
**Websocket**

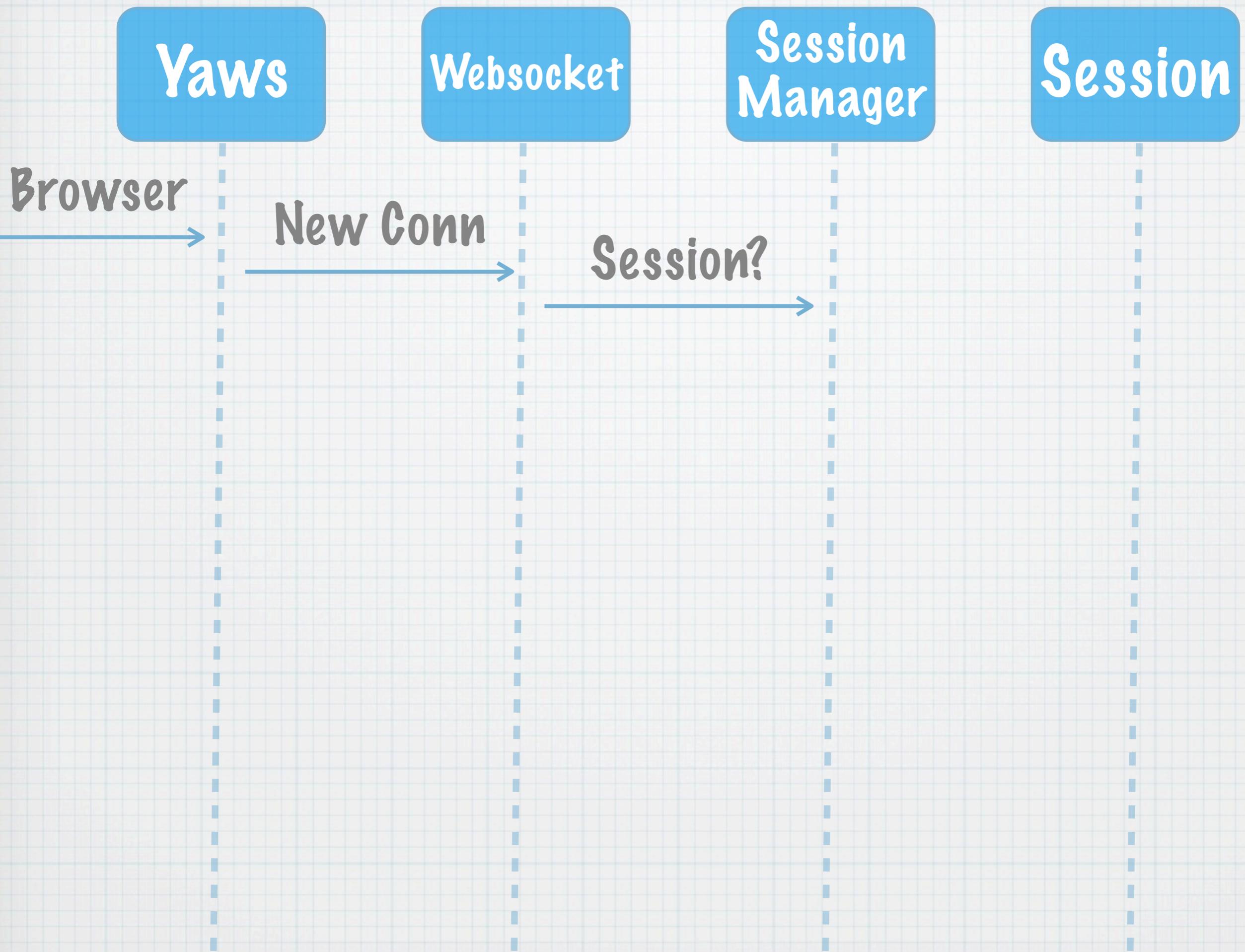
**Session  
Manager**

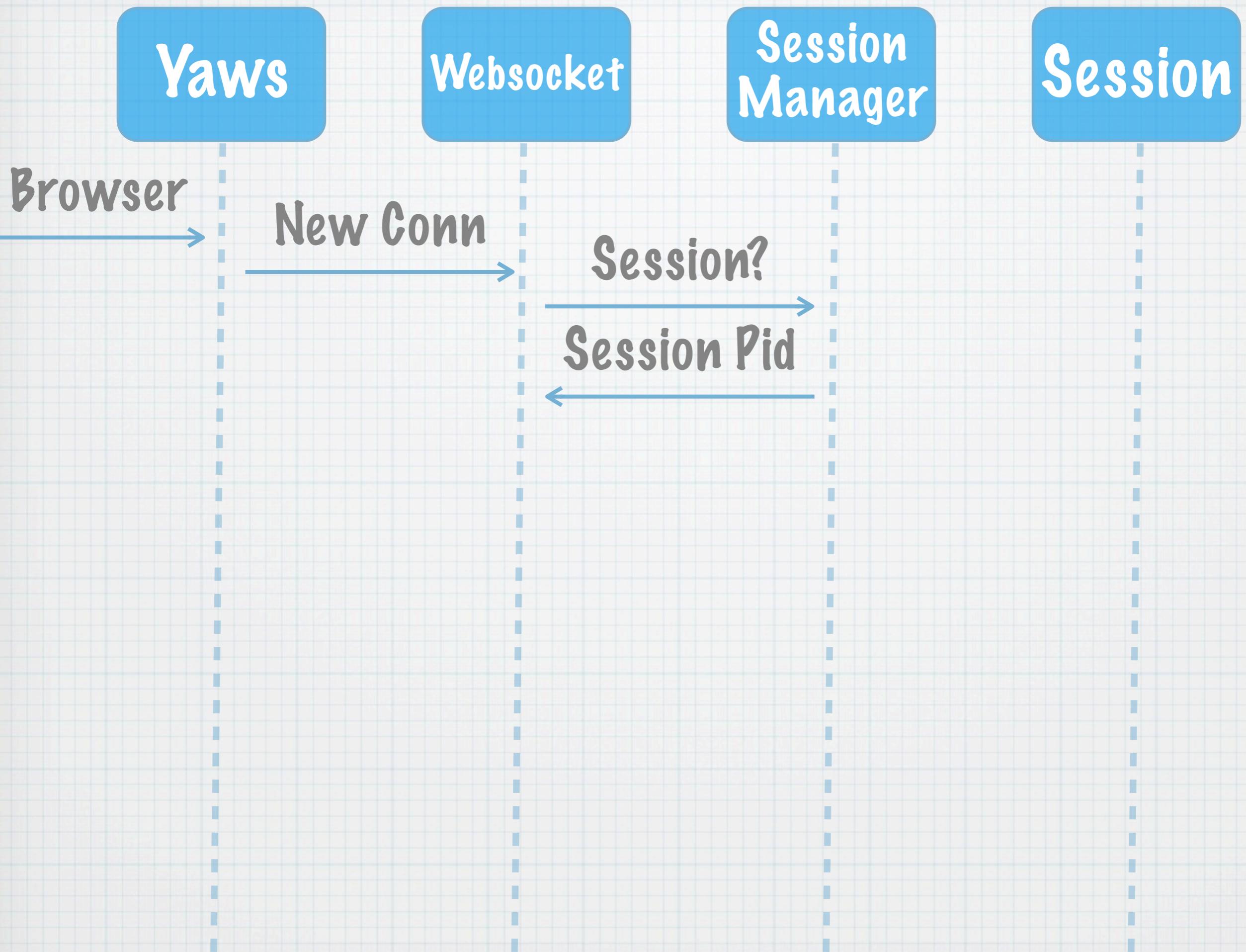
**Session**

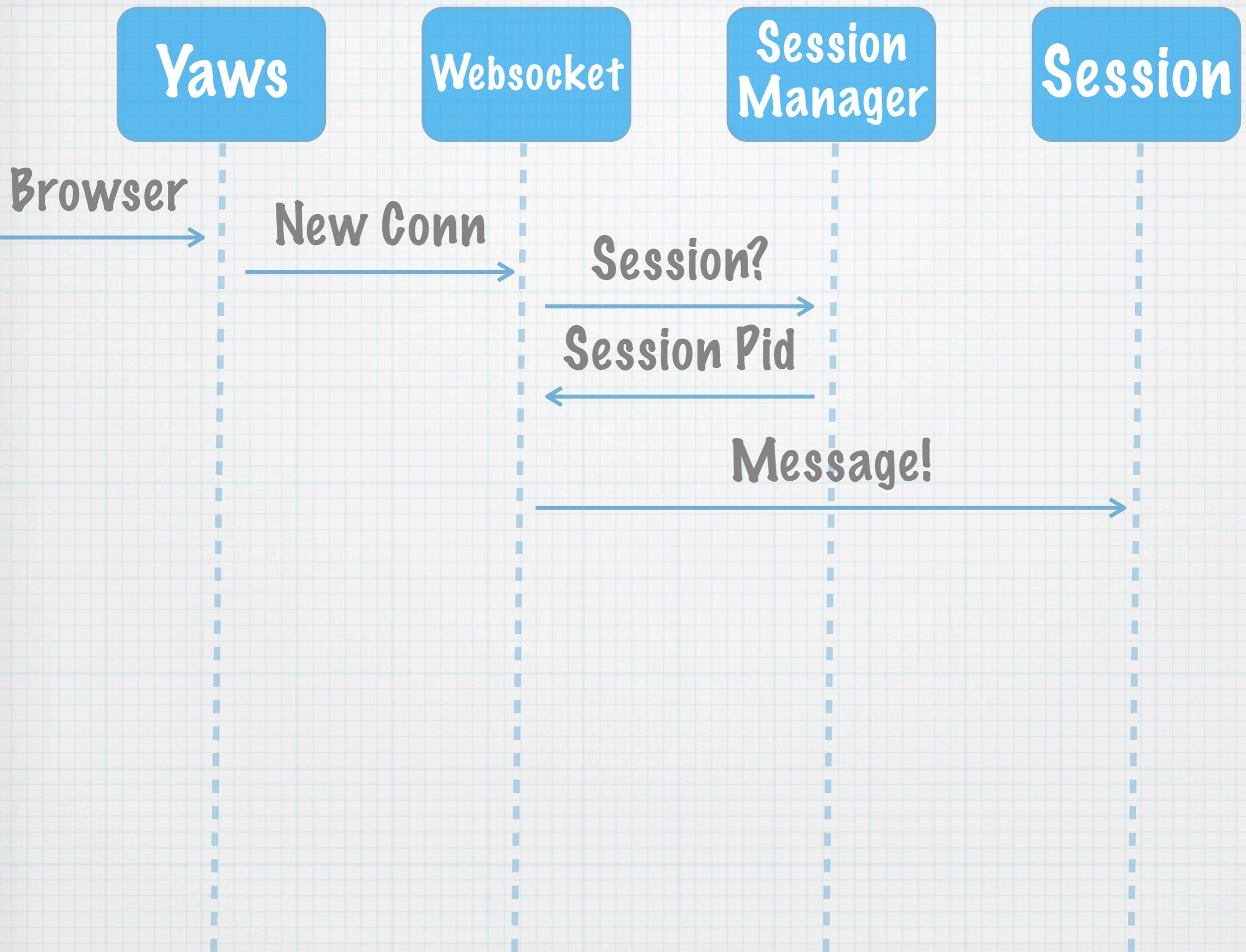
**Browser**

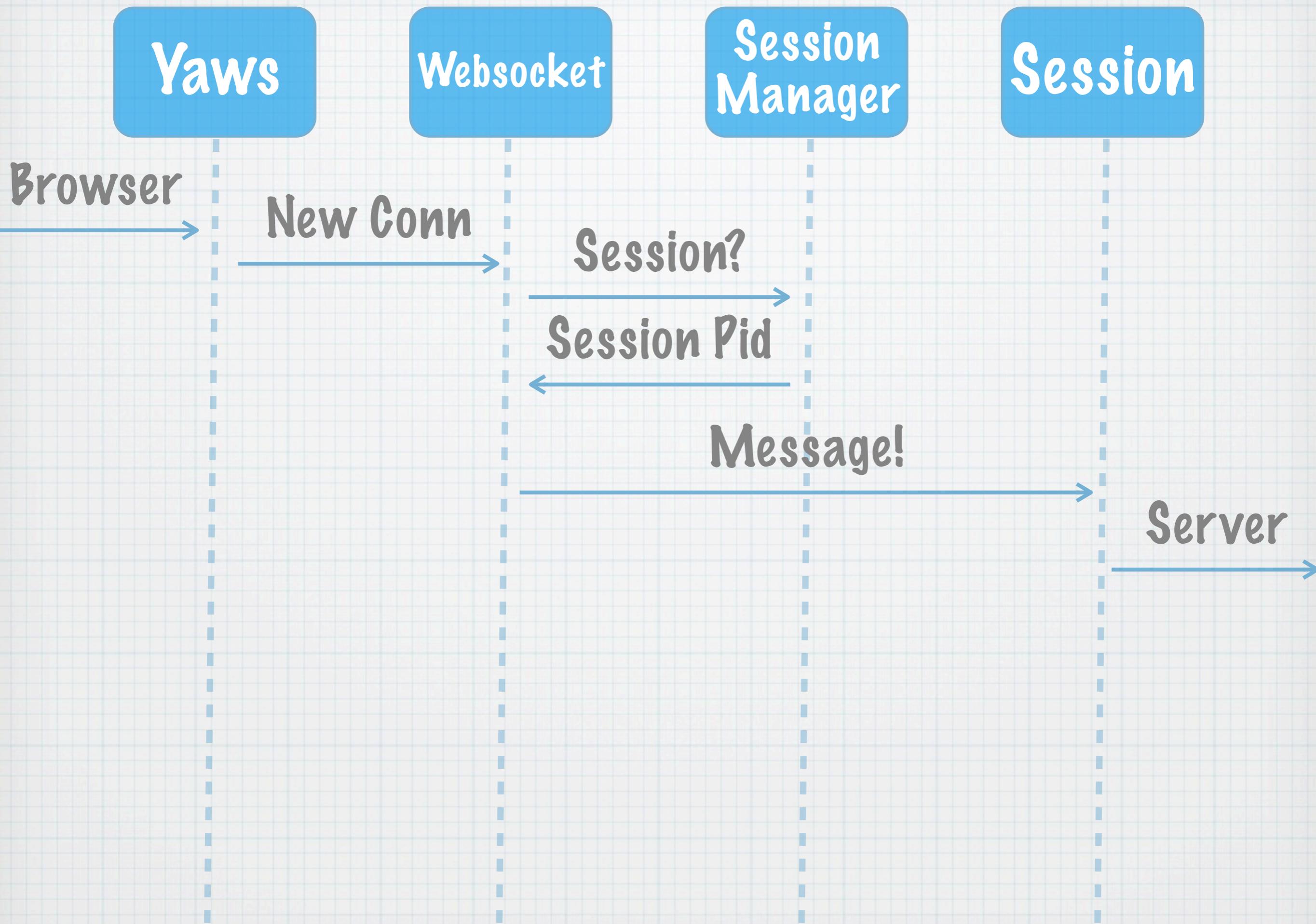












**Yaws**

**Channel  
Store**

**Session**

**Websocket**



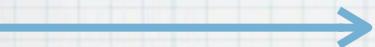
**Yaws**

**Channel  
Store**

**Session**

**Websocket**

**Server**



**Yaws**

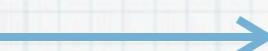
**Channel  
Store**

**Session**

**Websocket**

**Server**

**Sessions?**



**Yaws**

**Channel  
Store**

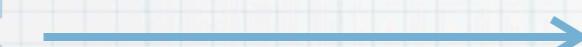
**Session**

**Websocket**

**Server**

**Sessions?**

**Session Pid**



**Yaws**

**Channel  
Store**

**Session**

**Websocket**

**Server**

**Sessions?**

**Session Pid**

**Message!**



**Yaws**

**Channel  
Store**

**Session**

**Websocket**

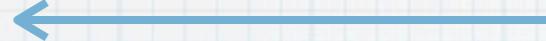
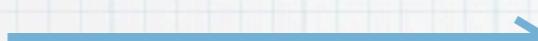
**Server**

**Sessions?**

**Session Pid**

**Message!**

**Message!**



**Yaws**

**Channel  
Store**

**Session**

**Websocket**

**Server**

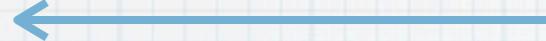
**Sessions?**

**Session Pid**

**Message!**

**Message!**

**Browser**



# Message Structure

```
{ data,  
  version,  
  type,  
  channel,  
  id,  
  sessionId  
}
```

# Message Structure

```
{ data,  
  version,  
  type, The message itself  
  channel,  
  id,  
  sessionId  
}
```

# Message Structure

```
{ data,  
  version,  
  type,  
  channel, Version of the message protocol  
  id,  
  sessionId  
}
```

# Message Structure

```
{ data,  
  version,  
  type,  
  channel,  
  id,  
  sessionId  
}
```

Four types: connect, message,  
channel-subscribe, poll

# Message Structure

```
{ data,  
  version,  
  type,  
  channel,  
  id,  
  sessi  
}  
Where the message is going
```

# Message Structure

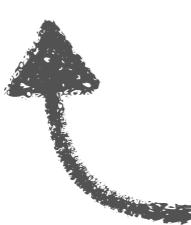
```
{ data,  
  version,  
  type,  
  channel,  
  id,  
  sessionId  
}
```



No Idea...

# Message Structure

```
{ data,  
  version,  
  type,  
  channel,  
  id,  
  sessionId  
}
```



Session id for connection

# Please Go Play With It

- \* <http://github.com/mashion/chloe>
- \* <http://mashion.net>
- \* <http://trottercashion.com>
- \* [@cashion](#)

# Moved to Mountain View

---

Be my friend?

# Thank You!

---

Good luck w/ Chloe