Mojolicious and Websockets

Erik Johansen
DK Hostmaster A/S
2013-06-25

Mojolicious and websockets

presentation

1) Basics
   a) Mojolicious - a real-time web application framework.
   b) Websockets

2) Two examples
   a) Send message, every 5 sec.
   b) Chat script – multiple clients talking together.

1) Basics

Mojolicious

- Written by Sebastian Riedel.
- Written in Perl.
- Runs on all operating systems supported by Perl.
- Designed for use in both simple (Mojolicious::Lite) and complex (Mojolicious) web applications. (Also note even shorter: perl -Mojo -E '...')
- Mojolicious::Lite – everything in one file (app, templates, content)

Mojolicious

- Install from CPAN.
- Prebuilt packages for NetBSD or Windows (ActiveState ppm).
- No package dependencies.
  (Added functionality when packages are installed)
- Scalability: Event loop got better at managing 10k concurrent connections.

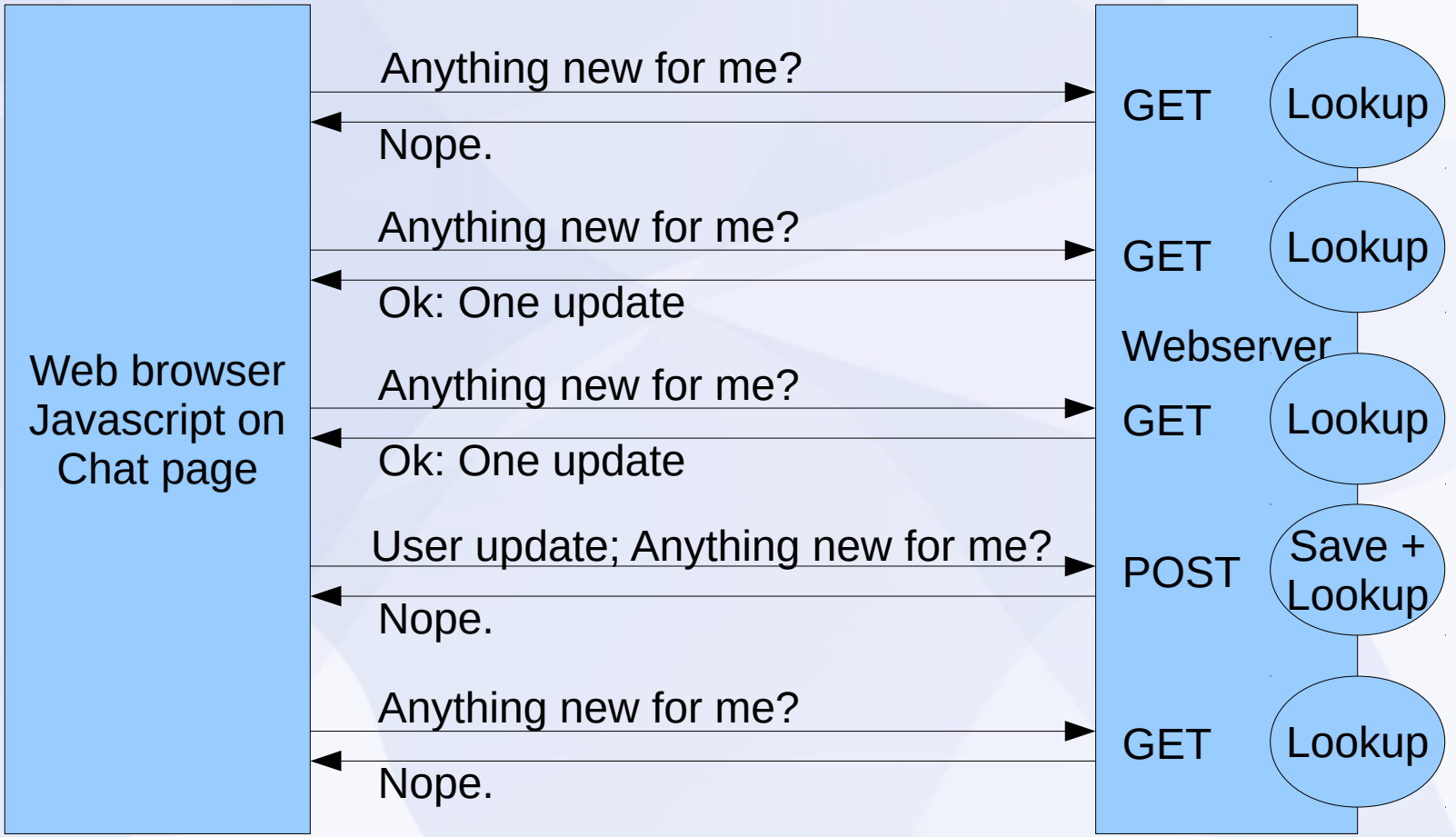
Websockets

`ws://www.server.dk/`
`wss://www.server.dk/`

- Part of HTML5
- Full-duplex communication over single TCP connection.
- Supported by all latest browsers, except Android.
- Idle when no data. Eliminates polling.
- Less overhead (cookies, authentication is sent only once)

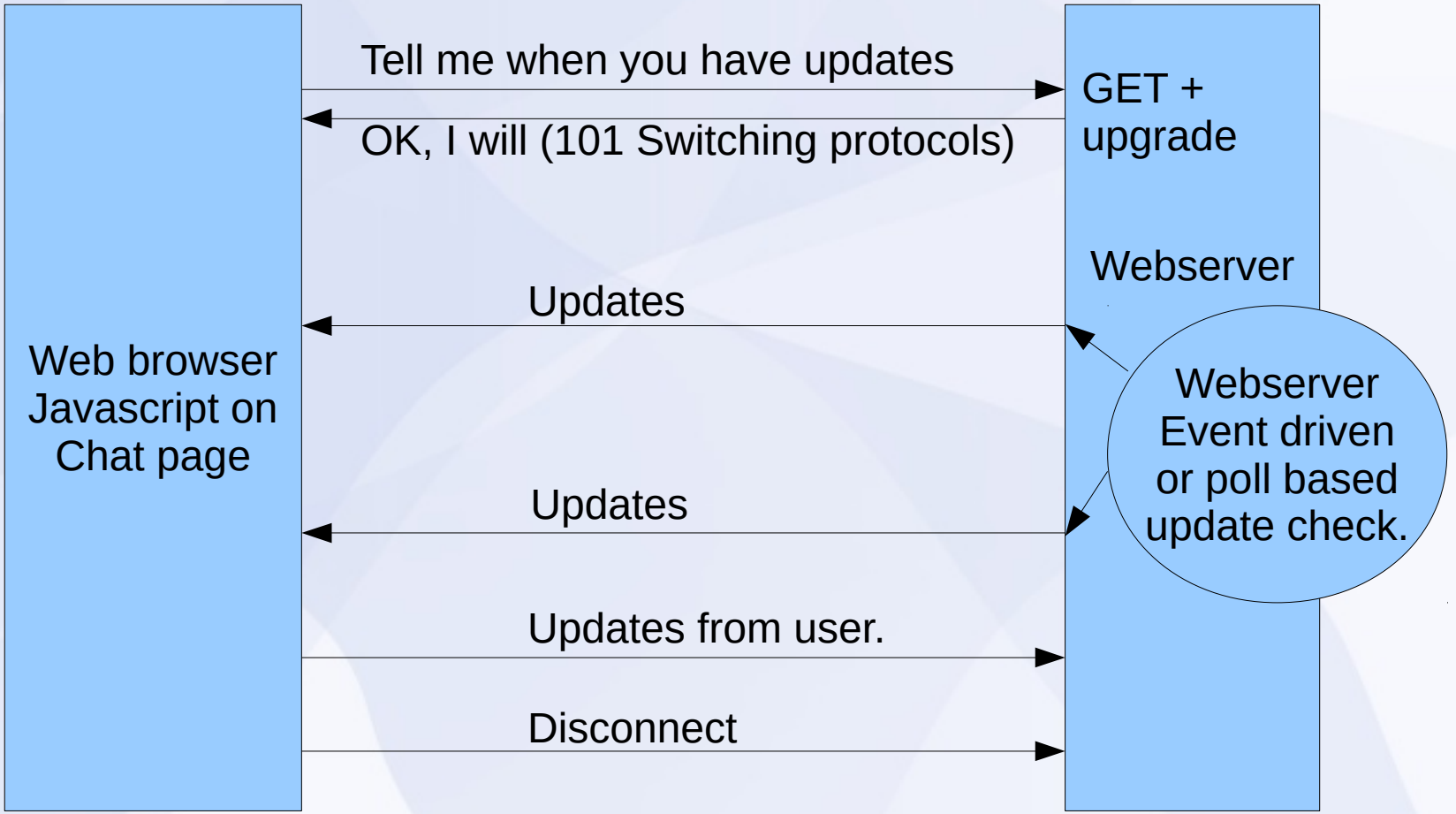
Poll

Before websockets



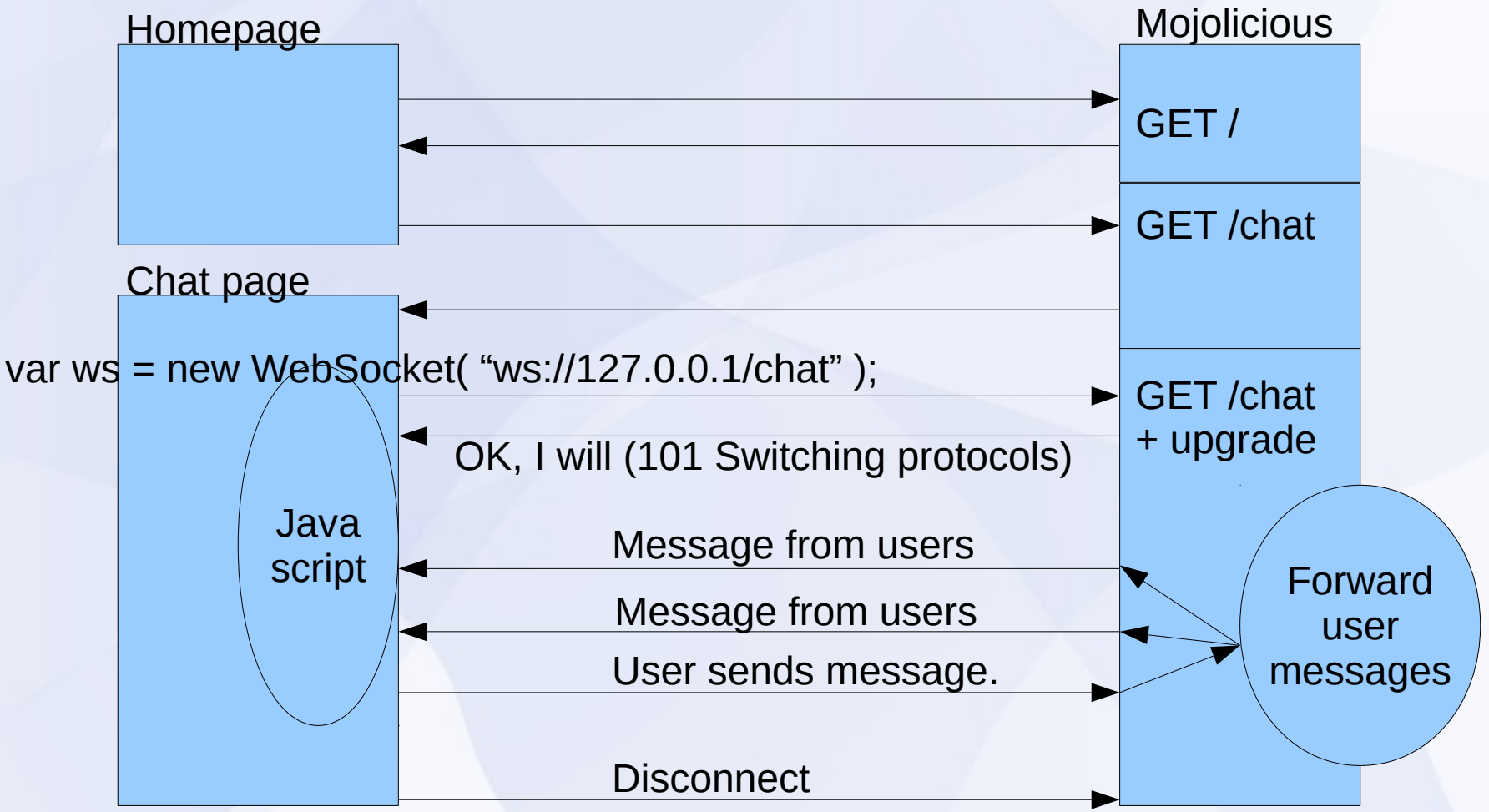
5 requests/responses (tcp+http+cookies etc)

Full duplex
With websockets

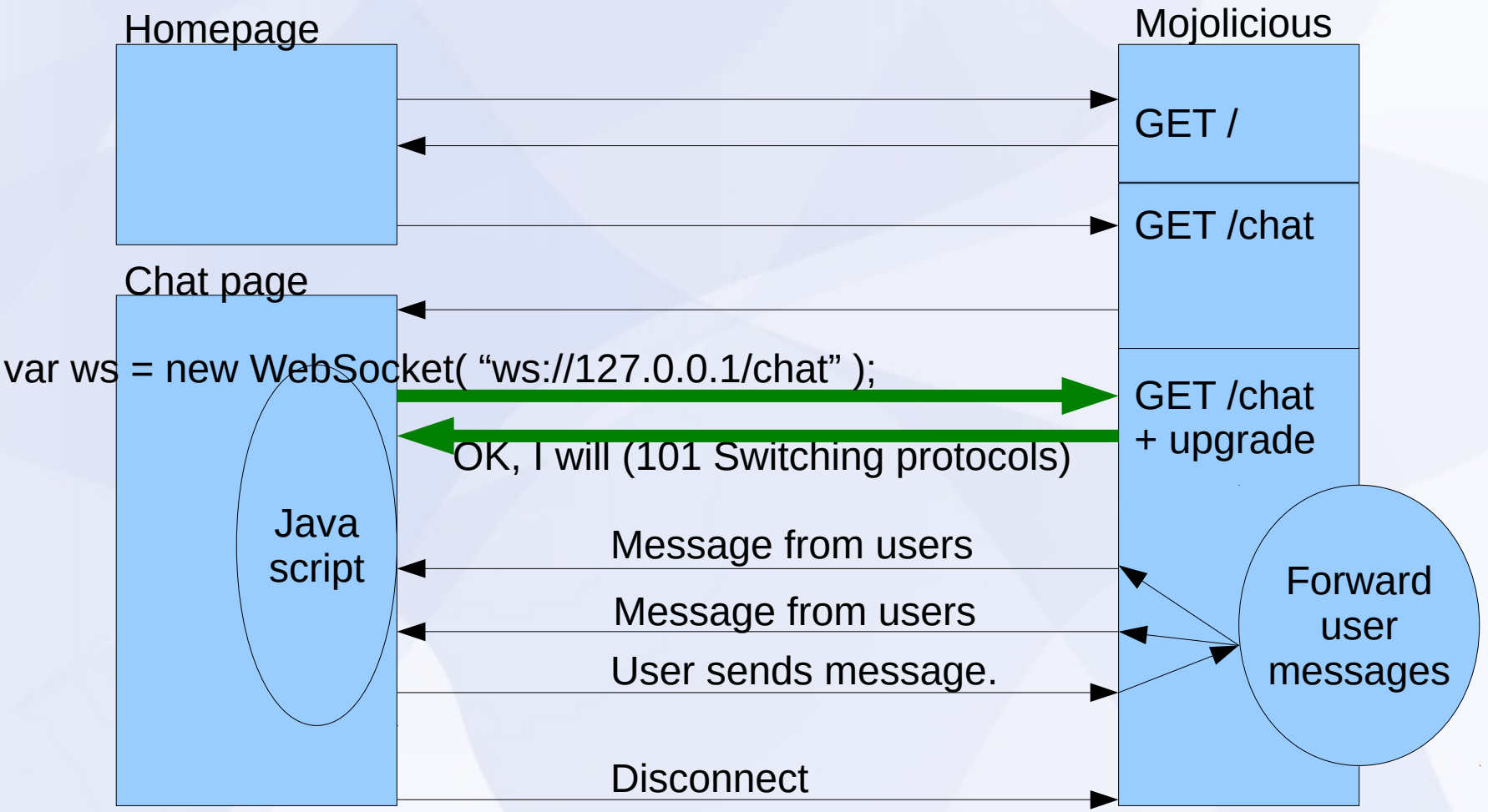


1 request/response (tcp+http+cookies etc) + 4 short messages

Our setup
With websockets

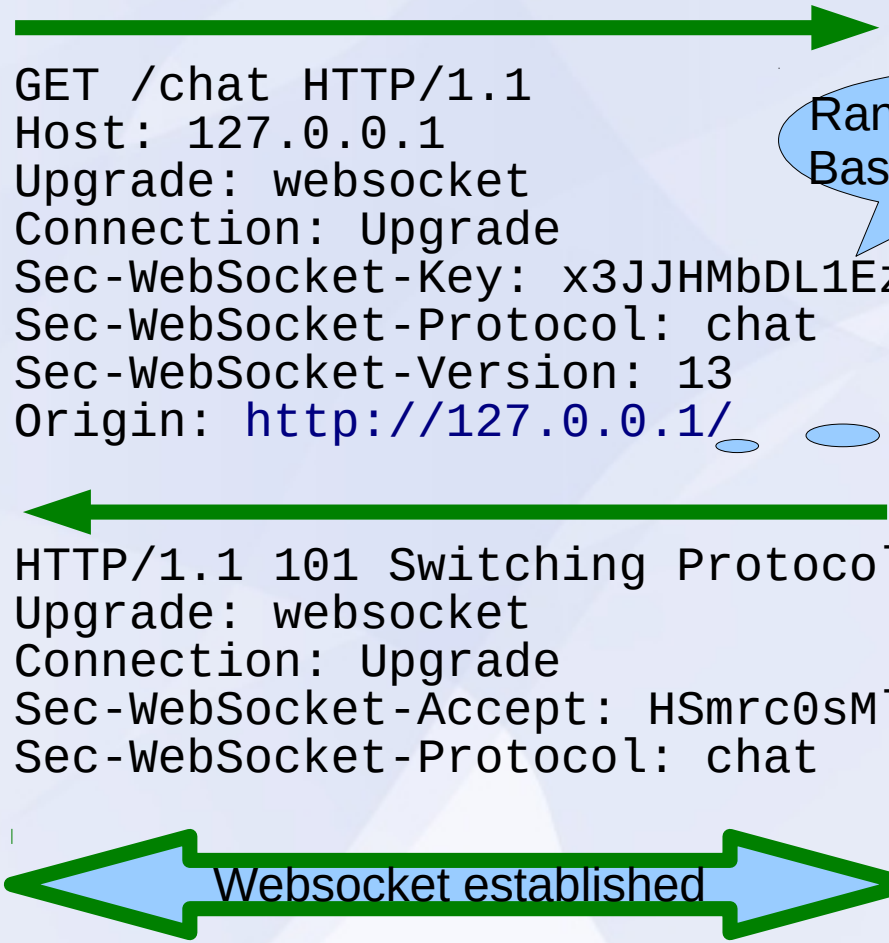


Our setup
With websockets

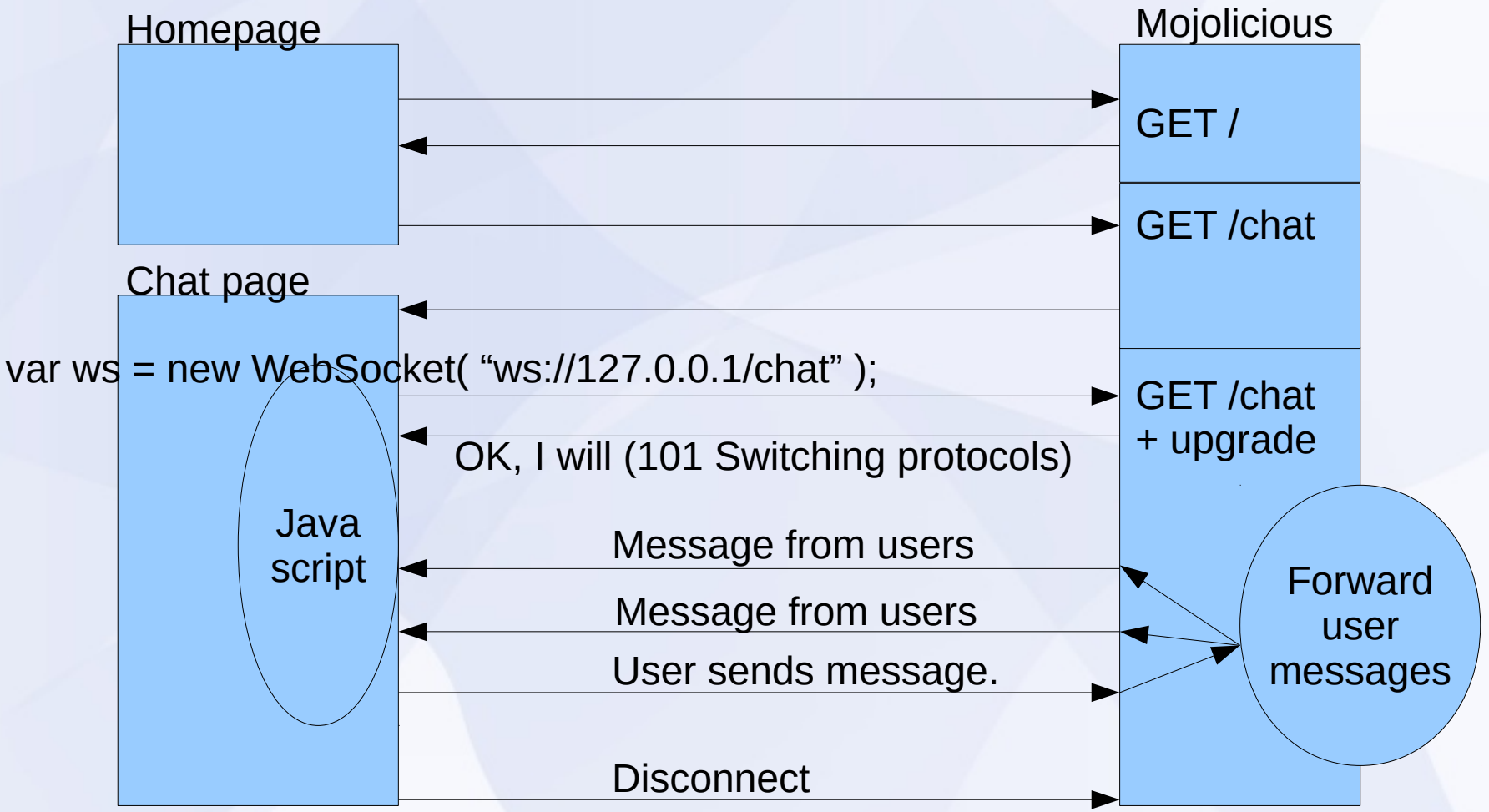


Websockets

Connection setup



Our setup
With websockets



Examples

Examples

- Create new Mojolicious app.
  http://127.0.0.1/
  Home page, Wswork.pm, template

a) Send message, every 5 sec.
  http://127.0.0.1/time
  Base page, Wstime.pm, template, websocket, javascript

b) Chat script – multiple clients talking together.
  http://127.0.0.1/chat
  Base page, Wschat.pm, template, websocket, javascript

- Application testing.

Create new Mojolicious app

```
$ sudo cpan Mojolicious

$ mojo generate app Wstest


$ cd wstest

$ touch log/development.log
$ tail -f log/development.log &

$ morbo script/wstest
```

lib/Wstest.pm

```
package Wstest;
use Mojo::Base 'Mojolicious';

# This method will run once at server start
sub startup {
  my $self = shift;

  # Router
  my $r = $self->routes;

  #
  # Top index page
  #
  $r->get('/')->to('wswork#welcome');

  #
  # /time  -- Websocket and GET welcome page
  #
  $r->websocket('/time')->to('wstime#accept');
  $r->get(      '/time')->to('wstime#welcome');

}

1;
```

lib/Wstest/Wswork.pm
for http://127.0.0.1/

```
package Wstest::Wswork;

use Mojo::Base 'Mojolicious::Controller';

sub welcome {
}

1;
```

templates/wswork/welcome.html.ep
for http://127.0.0.1/

```
<!doctype html>
<html>
<head>
    <title>Mojo Websocket Demo pages</title>

    <link rel="stylesheet" type="text/css"
href="/css/mystyle.css" />
</head>
<body>
    <h2>Mojo WebSocket Demo</h2>

    <ul>
     <li> <a href="/time">Time</a> </li>
     <li> <a href="/chat">Chat</a> </li>
    </ul>
</body>
</html>
```

templates/wstime/welcome.html.ep
for http://127.0.0.1/time/

```
<!doctype html>
<html>
<head>
    <title>Mojo Websocket Time Demo</title>

    <link rel="stylesheet" type="text/css"
href="/css/mystyle.css" />

    <script type="text/javascript"
src="/js/jquery.js"></script>
    <script type="text/javascript"
src="/js/wstime.js"></script>

</head>
<body>
    <h2>Mojo WebSocket Demo</h2>
    <div>
      <div class="timearea">
        ## Data should appear here
      </div>
    </div>
</body>
</html>
```

public/js/wstime.js
for http://127.0.0.1/js/wstime.js

```
// Take the current document location href
// and replace http or https with ws or wss

console.log("url  :  " + document.location.href );

var wsurl = document.location.href.replace(/^http/i,"ws");

console.log("wsurl:  " + wsurl );

...
```

public/js/wstime.js
for http://127.0.0.1/js/wstime.js

```
...

console.log("About to connect to websocket on: " + wsurl);

var ws = new WebSocket( wsurl );

ws.onopen = function() {
    console.log("Websocket open");
};

ws.onmessage = function(msg) {
    console.log("Got message: "+ msg);

    // var res = JSON.parse(msg.data);

    // Lets do something with the message
    $('.timearea').append( "<br>" +
        $("<div/>").text(msg.data).html()
    );
}
```

lib/Wstest/Wstime.pm

```
package Wstest::Wstime;
use Mojo::Base 'Mojolicious::Controller';

sub welcome {
    # default renders templates/wstime/welcome.html.ep
}

sub accept {
    # ... see next slide ...
}

1;
```

lib/Wstest/Wstime.pm

```
sub accept {
    my $self = shift;

    my $interval = 5;

    # How long to go without events before closing connection.
Logs timeouts as: "Inactivity timeout"
    Mojo::IOLoop->stream($self->tx->connection)->timeout(300);

    my $scheduler = Mojo::IOLoop->recurring( $interval => sub {

        my $data = scalar localtime();

        $self->app->log->debug("Sending data: ".$data);
        $self->send( $data );  # Possibly add JSON encoding

    });

    $self->on( finish => sub {
        $self->app->log->debug("Finished websocket");
        Mojo::IOLoop->remove( $scheduler );
    });

}
```

Demo

`http://127.0.0.1:3000/time/`

lib/Wstest.pm

```
package Wstest;
use Mojo::Base 'Mojolicious';

# This method will run once at server start
sub startup {

  …

  #
  # /time  -- Websocket and GET welcome page
  #
  $r->websocket('/time')->to('wstime#accept');
  $r->get(      '/time')->to('wstime#welcome');

  #
  # /chat  -- Websocket and GET welcome page
  #
  $r->websocket('/chat')->to('wschat#accept');
  $r->get(      '/chat')->to('wschat#welcome');

}

1;
```

templates/wschat/welcome.html.ep for http://127.0.0.1/chat/

```
<!doctype html>
<html>
<head>
   <title>Mojo Websocket Demo pages</title>

   <link rel="stylesheet" type="text/css"
href="/css/mystyle.css" />
   <script type="text/javascript" src="/js/jquery.js"></script>

</head>
<body>
   <h2>Mojo WebSocket Demo</h2>
   <div>
     <form>
     <div class="chatarea">
       ## Data here
     </div>
     <input class="chatinput" type="text" value="">
     </form>
   </div>
</body>
   <script type="text/javascript" src="/js/wschat.js"></script>
</html>
```

public/js/wschat.js

```
var wsurl = document.location.href.replace(/^http/i,"ws");
console.log("About to connect to websocket on: " + wsurl);

var ws = new WebSocket( wsurl );

ws.onopen = function() {
    console.log("Websocket open");
};

ws.onmessage = function(msg) {
    console.log("Got message: ", msg);

    // var res = JSON.parse(msg.data);

    // Lets do something with the message
    $('.chatarea').append("<br>" +
        $("<div/>").text(msg.data).html()
    );
}

...
```

public/js/wschat.js

```
...

$('.chatinput').keydown( function(e) {
  if ( e.which != 13 ) return true;

  var inp = $('.chatinput');
  var msg = inp.val();
  console.log("Send message: "+msg);
  ws.send(msg);
  inp.val('');
  return false; // Do not use keypress for anything
});
```

lib/Wstest/Wschat.pm

```
package Wstest::Wschat;
use Mojo::Base 'Mojolicious::Controller';

sub welcome {
}

our %clients;

sub accept {
   ...
}

sub broadcast {
    my $self = shift;
    my $from = shift;
    my $msg = shift;

    while ( my($id, $ws) = each %clients ) {
        $ws->send($msg) unless $id eq $from;
    }
}

1;
```

lib/Wstest/Wschat.pm

```
sub accept {
    my $self = shift;

    Mojo::IOLoop->stream($self->tx->connection)->timeout(300);

    my $id = sprintf "%s", $self->tx;
    $self->app->log->debug("Joined by: ".$id);

    $self->send("Talking to: ".$_) foreach keys %clients;

    $clients{ $id } = $self;
    $self->broadcast($id, "Joined by ".$id);

    $self->app->log->debug(
        scalar(keys %clients)." Active clients");

    $self->on( message => sub {
        my($ws, $msg) = @_;
        $self->app->log->debug("Got message: ".$msg);
        $self->broadcast($id, $msg);
    });

    $self->on( finish => sub {
        $self->app->log->debug("Finished websocket");
        $self->broadcast($id, "Bye bye from ".$id);
        delete $clients{ $id };
    });
}
```

Demo

http://127.0.0.1:3000/chat/

Test

Test

t/basic.t

```
#!/usr/bin/perl

use Test::More tests => 22;
use Test::Mojo;
use FindBin qw($Bin);
use lib "$Bin/../lib";

my $t = Test::Mojo->new('Wstest');

$t ->get_ok('/')
   ->status_is(200)
   ->content_like(qr/Demo Page/i);

my $tt = $t->websocket_ok('/time')
   ->send_ok('hello', 'Sent messages are ignored');
diag("Expexct to wait 5 secs for timed message");
$tt->message_ok
   ->message_like(qr/\b\d{4}\b/)  # Match year of timestamp.
   ->finish_ok;

...
```

Test

t/basic.t

```
…

my $c1 = $t->websocket_ok('/chat');
my $c2 = $t->websocket_ok('/chat');

$c1->send_ok('hello from c1', 'Sent message from c1');
$c2->send_ok('hello from c2', 'Sent message from c2');

$c1->message_ok
   ->message_like(qr/Joined by/);

$c2->message_ok
   ->message_like(qr/Talking to/);

$c1->message_ok
   ->message_like(qr/hello from c1/)
   ->finish_ok;

$c2->message_ok
   ->message_like(qr/hello from c2/)
   ->finish_ok;
```

Test
t/basic.t

```
$ xhmod +x t/basic.t

$ t/basic.t
1..22
ok 1 - GET /
ok 2 - 200 OK
ok 3 - content is similar
ok 4 - WebSocket /time
ok 5 - Sent messages are ignored
# Expexct to wait 5 secs for timed message
ok 6 - message received
ok 7 - message is similar
ok 8 - closed WebSocket
ok 9 - WebSocket /chat
ok 10 - WebSocket /chat
ok 11 - Sent message from c1
ok 12 - Sent message from c2
ok 13 - message received
ok 14 - message is similar
ok 15 - message received
ok 16 - message is similar
ok 17 - message received
ok 18 - message is similar
ok 19 - closed WebSocket
ok 20 - message received
ok 21 - message is similar
ok 22 - closed WebSocket
```

Files

```
$ find . -type f
./lib/Wstest.pm
./lib/Wstest/Wschat.pm
./lib/Wstest/Wstime.pm
./lib/Wstest/Wswork.pm
./log/development.log
./public/css/mystyle.css
./public/js/jquery.js
./public/js/wschat.js
./public/js/wstime.js
./script/wstest
./t/basic.t
./templates/wschat/welcome.html.ep
./templates/wstime/welcome.html.ep
./templates/wswork/welcome.html.ep
./wstest.conf
```

Downloads

`http://www.uniejo.dk/presentations/2013-06-25.pdf`
This presentation

`http://www.uniejo.dk/presentations/2013-06-25.tgz`
Gzipped tar archive of files used in this presentation.

End of presentation...