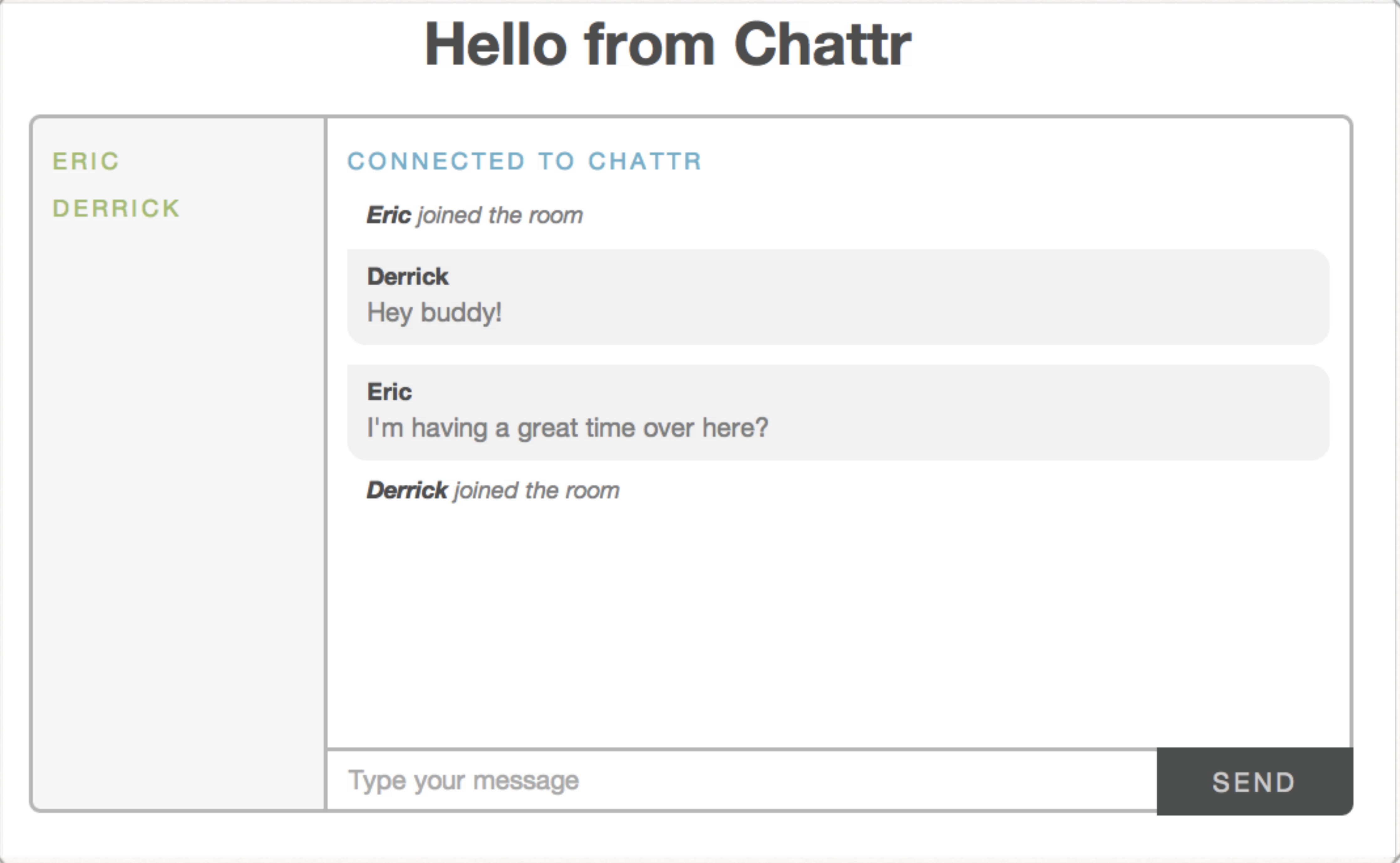
SOCKET.IO

- LEVEL SIX -

CHATTR



Hello from Chattr

ERIC

DERRICK

CONNECTED TO CHATTR

Eric *joined the room*

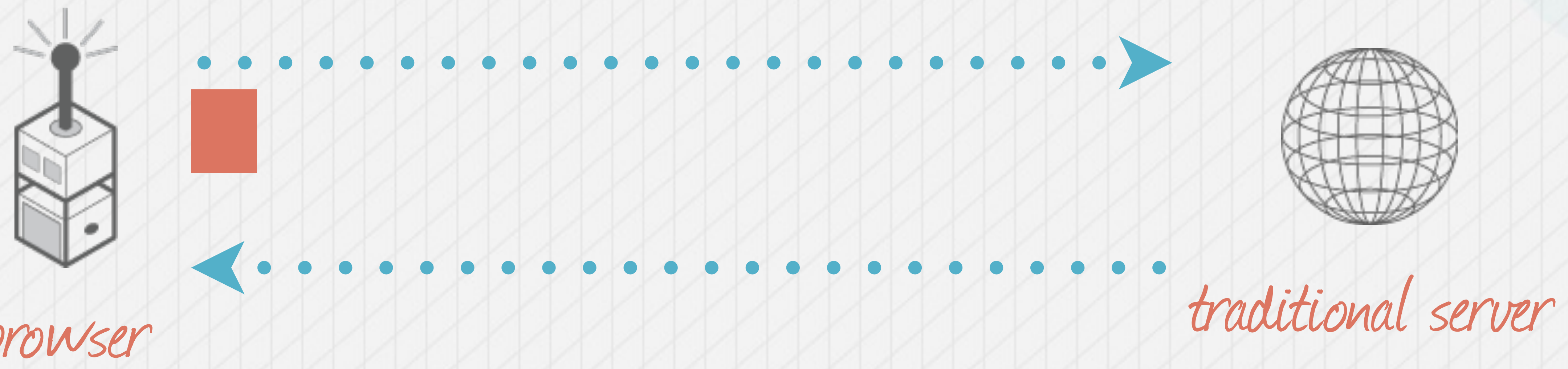
Derrick
Hey buddy!

Eric
I'm having a great time over here?

Derrick *joined the room*

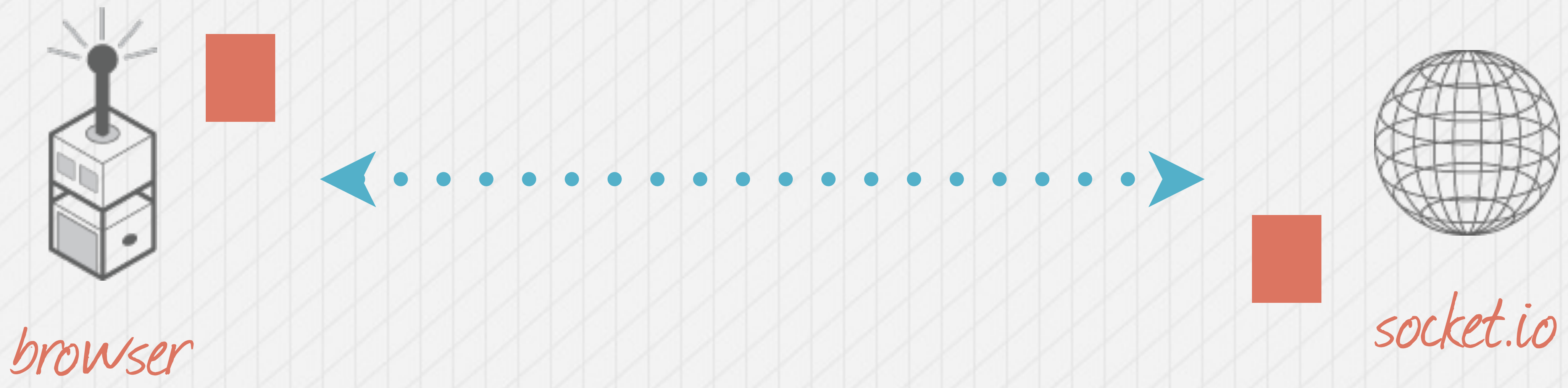
SOCKET.IO

WEBSOCKETS



Traditional request/response cycle

WEBSOCKETS



Using duplexed websocket connection

SOCKET.IO FOR WEBSOCKETS

Abstracts websockets with fallbacks

```
$ npm install --save socket.io
```

app.js

```
var express = require('express');
var app = express();
var server = require('http').createServer(app);
var io = require('socket.io')(server);

io.on('connection', function(client) {
  console.log('Client connected...');
});

app.get('/', function (req, res) {
  res.sendFile(__dirname + '/index.html');
});

server.listen(8080);
```

SOCKET.IO FOR WEBSOCKETS

socket.io client connects to the server

index.html

```
<script src="/socket.io/socket.io.js"></script>

<script>
  var socket = io.connect('http://localhost:8080');
</script>
```

SENDING MESSAGES TO CLIENT

app.js

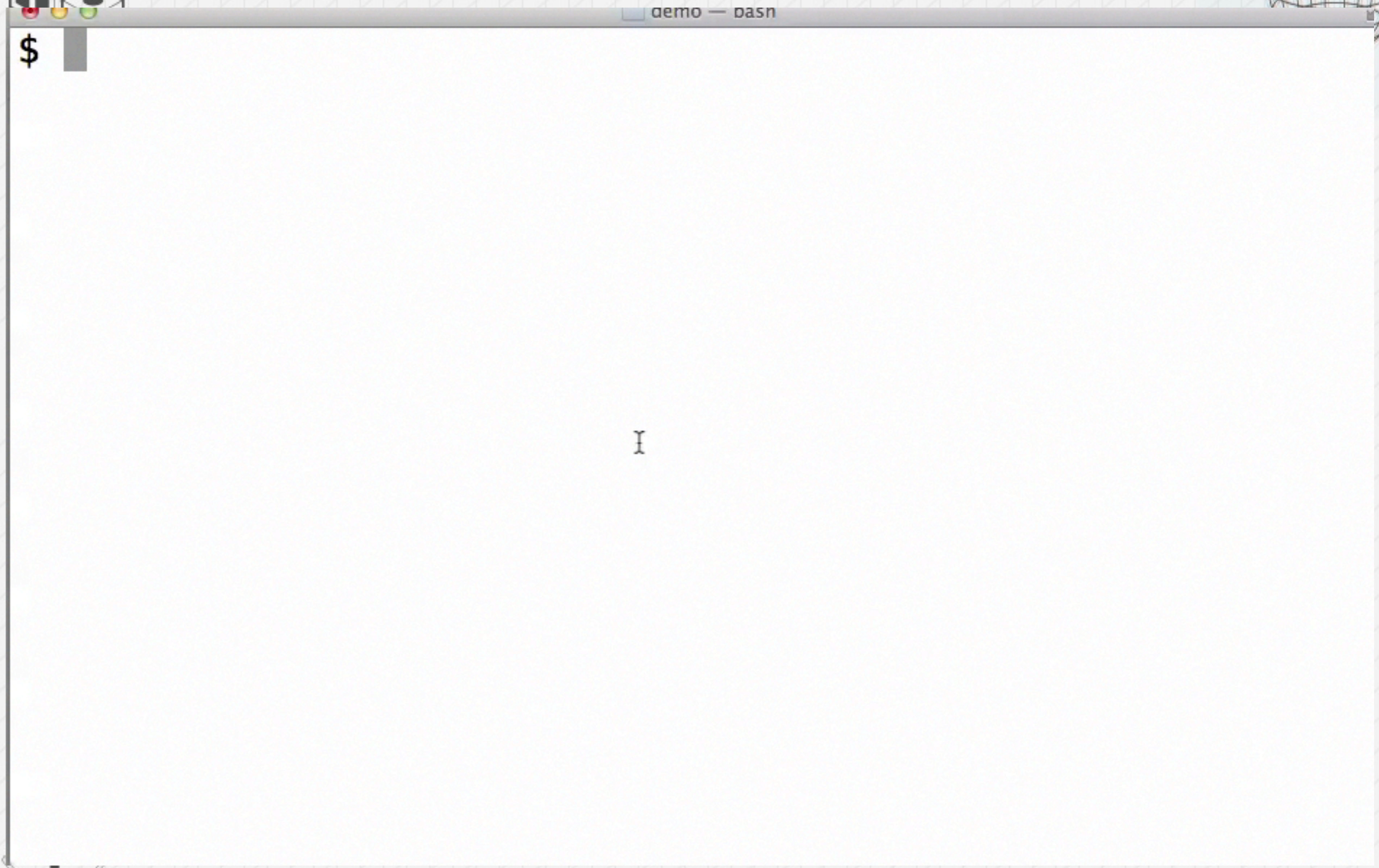
```
io.on('connection', function(client) {
  console.log('Client connected...');

    emit the 'messages' event on the client
  client.emit('messages', { hello: 'world' });
});
```

index.html

```
<script src="/socket.io/socket.io.js"></script>
<script>
  var socket = io.connect('http://localhost:8080');
  socket.on('messages', function (data) {
    alert(data.hello);
  });            listen for 'messages' events
</script>
```

CHATTR HELLO WORLD



```
$ 
```

SENDING MESSAGES TO SERVER

app.js

```
io.on('connection', function(client) {
  client.on('messages', function (data) {
    console.log(data);
  });
});
```

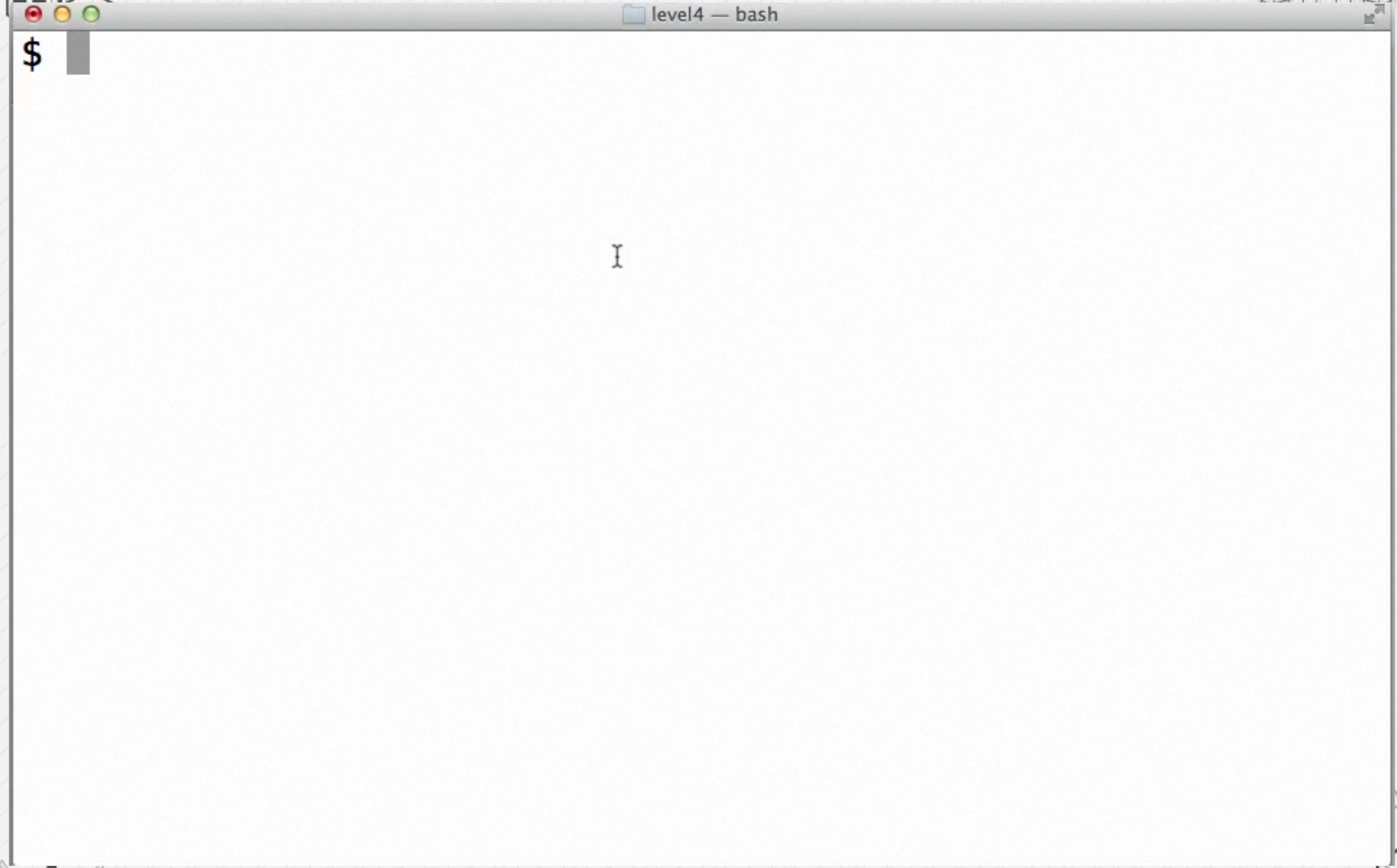
listen for 'messages' events

index.html

```
<script>
  var socket = io.connect('http://localhost:8080');
  $('#chat_form').submit(function(e){
    var message = $('#chat_input').val();
    socket.emit('messages', message);
  });
</script>
```

emit the 'messages' event on the server

CHATTR HELLO WORLD

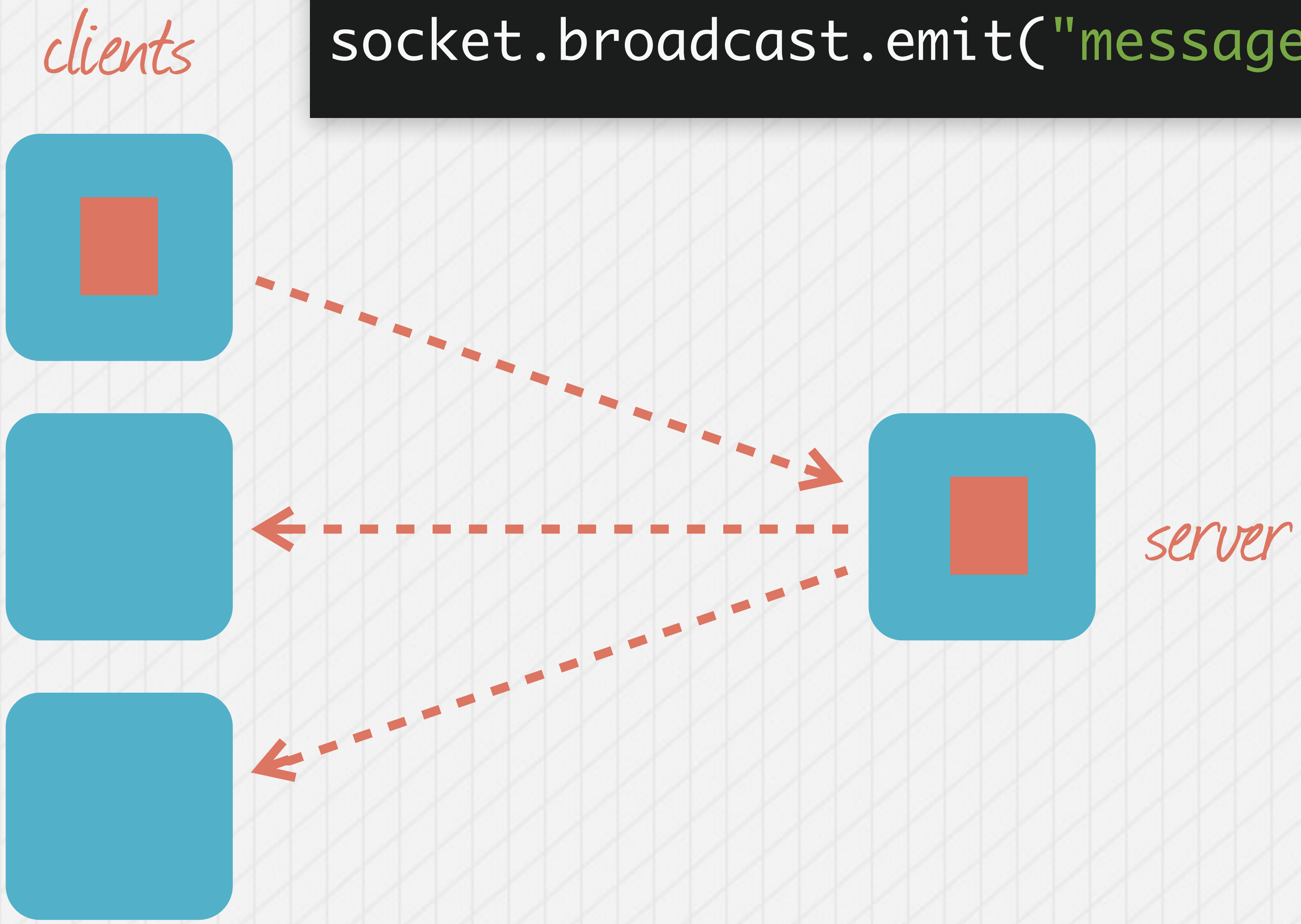


```
$
```

BROADCASTING MESSAGES

app.js

```
socket.broadcast.emit("message", 'Hello');
```



SOCKET.IO

BROADCASTING MESSAGES

app.js

```
io.on('connection', function(client) {
  client.on('messages', function (data) {
    client.broadcast.emit("messages", data);
  });
});
```

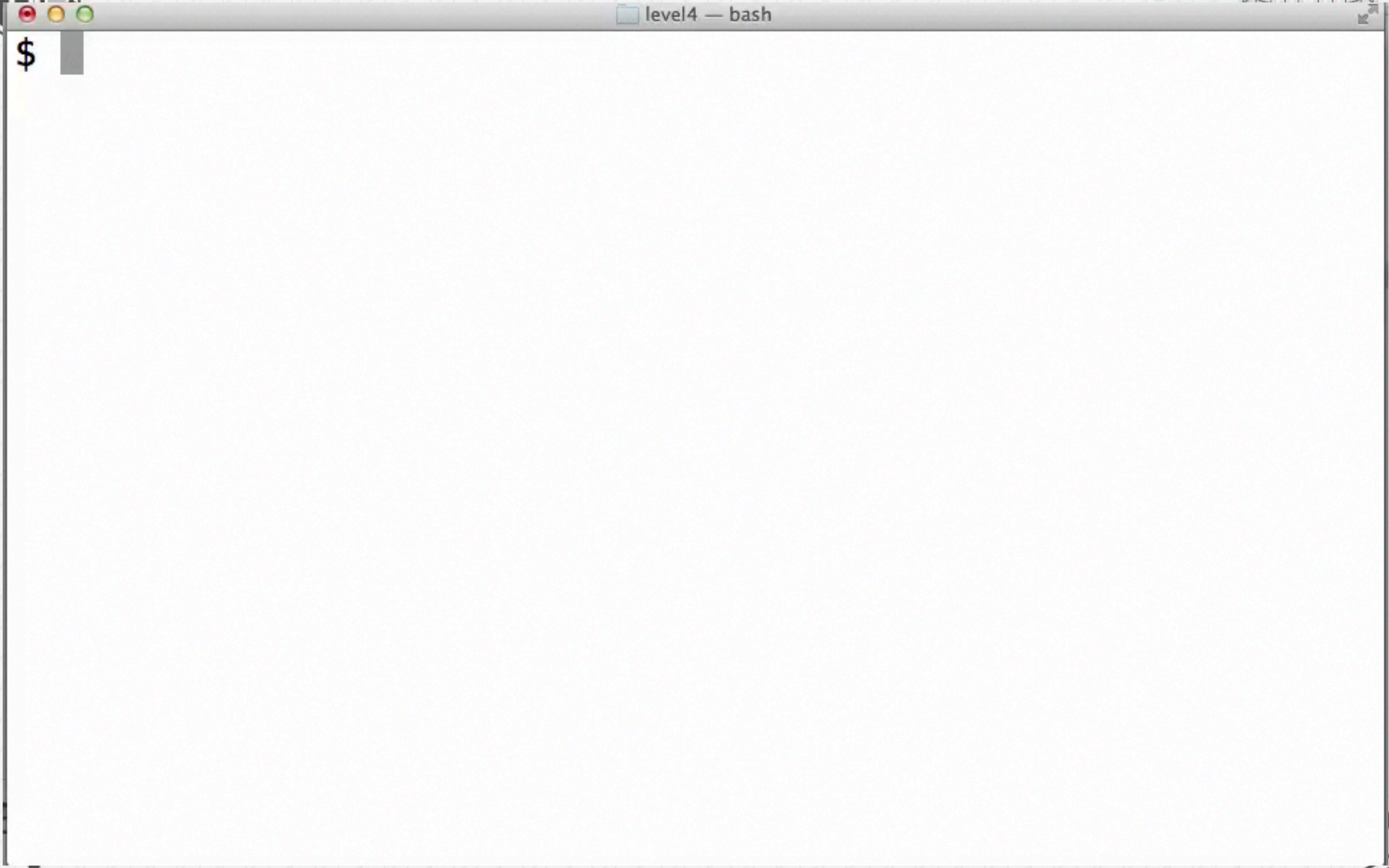
broadcast message to all other clients connected

index.html

```
<script>
  ...
  socket.on('messages', function(data) { insertMessage(data) });
</script>
```

insert message into the chat

BROADCASTING MESSAGES



```
$
```

SAVING DATA ON THE SOCKET

app.js

```
io.on('connection', function(client) {
  client.on('join', function(name) {
    client.nickname = name;
  });
});
```

set the nickname associated with this client

index.html

```
<script>
  var server = io.connect('http://localhost:8080');
  server.on('connect', function(data) {
    $('#status').html('Connected to chattr');
    nickname = prompt("What is your nickname?");


    server.emit('join', nickname);
  });
</script>
```

notify the server of the users nickname

SAVING DATA ON THE CLIENT

app.js

```
io.on('connection', function(client) {
  client.on('join', function(name) {
    client.nickname = name;
  });

  client.on('messages', function(data){

    var nickname = client.nickname;

    client.broadcast.emit("message", nickname + ": " + message);

    client.emit("messages", nickname +  ": " + message);

  });
});
```

set the nickname associated with this client

get the nickname of this client before broadcasting message

broadcast with the name and message

send the same message back to our client