SOCKET.IO

Sockets work based on events. There are some reserved events, which can be accessed using the socket object on the server side.

These are −

- Connect
- Message
- Disconnect
- Reconnect
- Ping
- Join and
- Leave

The client-side socket object also provides us with some reserved events, which are −

- Connect
- Connect_error
- Connect_timeout
- Reconnect, etc

In the Hello World example, we used the connection and disconnection events to log when a user connected and left. Now we will be using the message event to pass message from the server to the client. To do this, modify the **io.on** ('connection', function(socket)) call to include the following −

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

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

io.on('connection', function(socket) {
   console.log('A user connected');

   //Send a message after a timeout of 4seconds
   setTimeout(function() {
      socket.send('Sent a message 4seconds after connection!');
   }, 4000);

   socket.on('disconnect', function () {
      console.log('A user disconnected');
   });
});
```

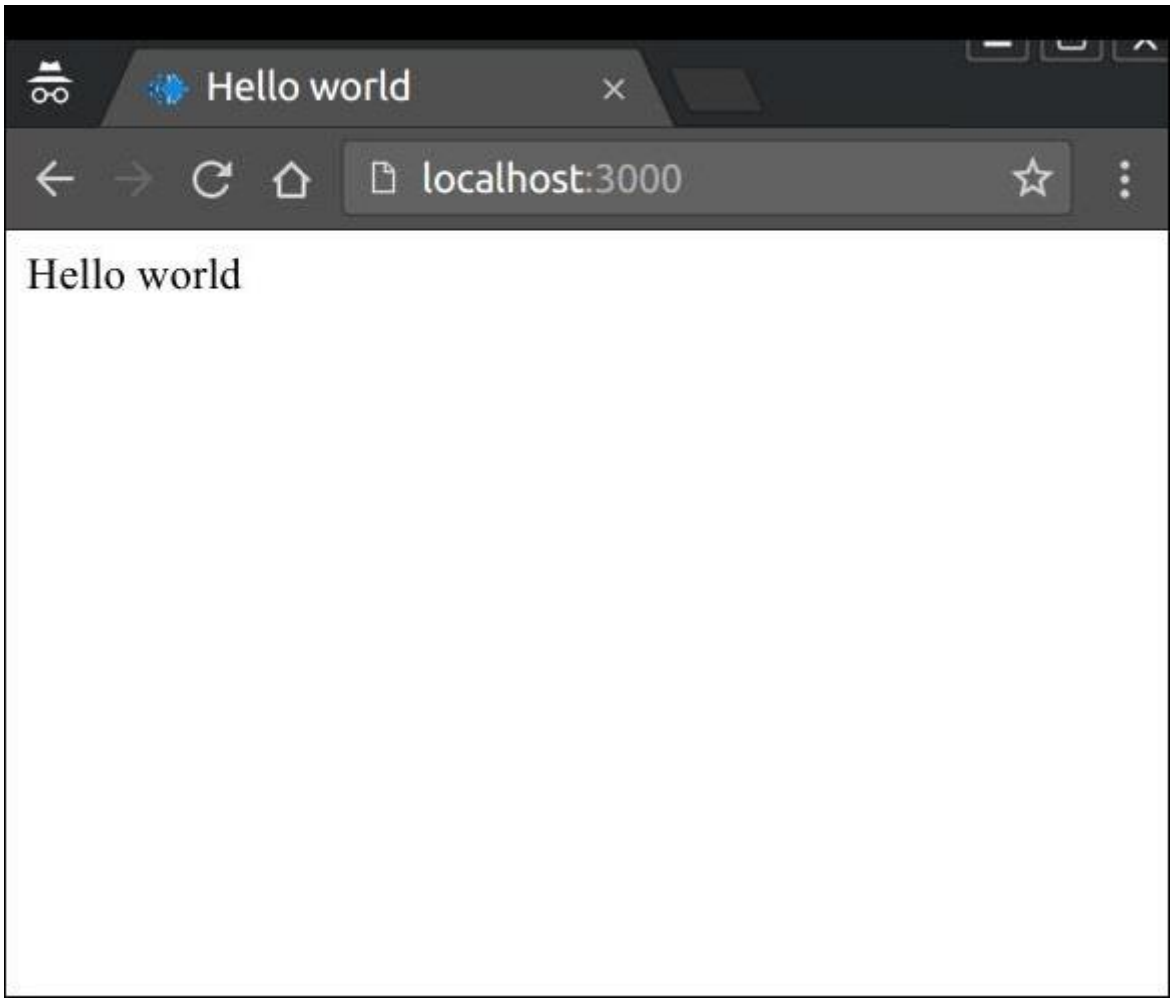
```
http.listen(3000, function() {
   console.log('listening on *:3000');
});
```

This will send an event called **message(built in)** to our client, four seconds after the client connects. The send function on socket object associates the 'message' event.

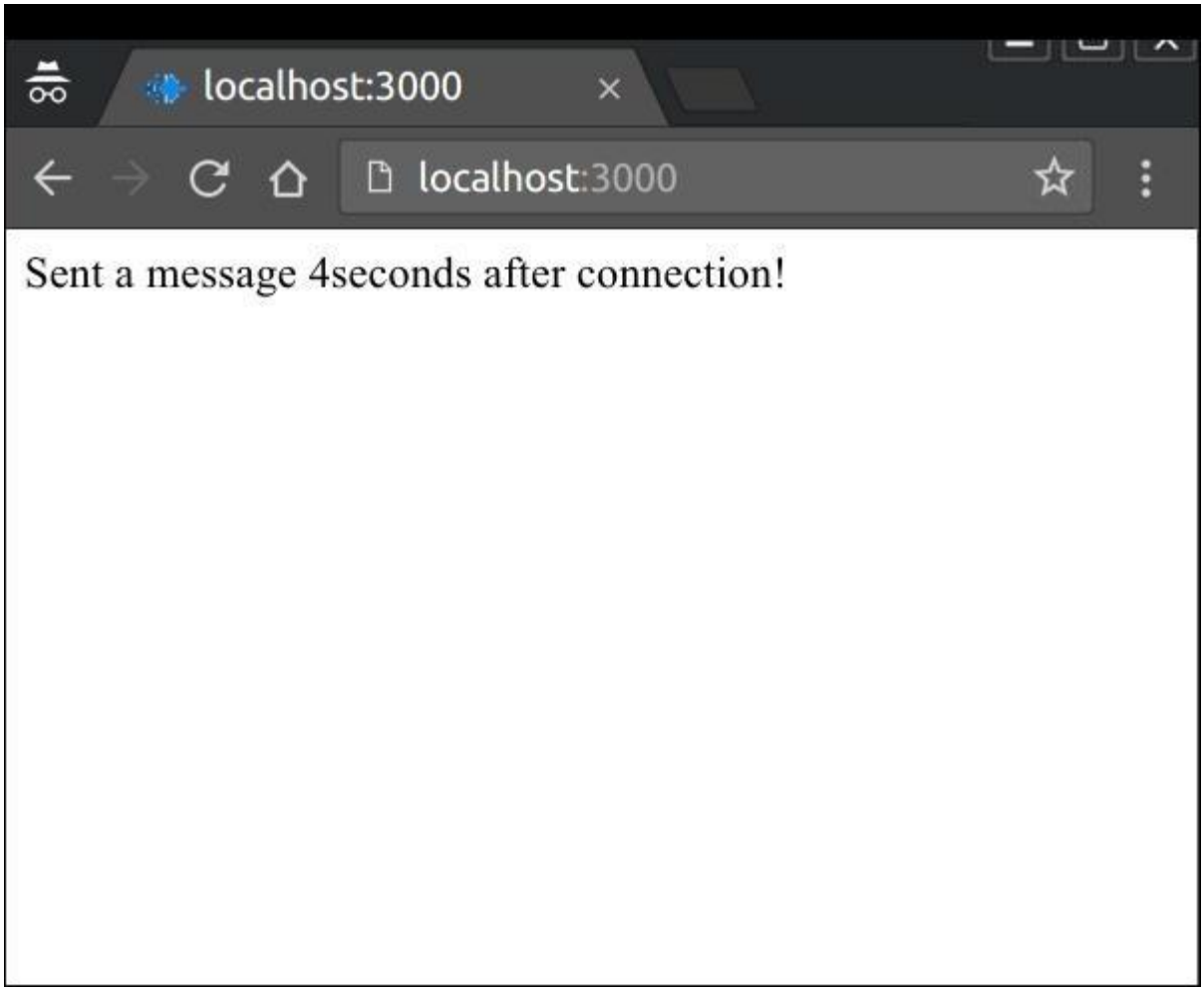
Now, we need to handle this event on our client side. So, edit your index.html script tag to include the following code −

```
<script>
   var socket = io();
   socket.on('message', function(data){document.write(data)});
</script>
```

We are now handling the 'message' event on the client. When you go to the page in your browser now, you will be presented with the following screenshot.



After 4 seconds pass and the server sends the message event, our client will handle it and produce the following output −



Note − We sent a string of text here; we can also send an object in any event.

Message was a built-in event provided by the API, but is of not much use in a real application, as we need to be able to differentiate between events.

To allow this, Socket.IO provides us the ability to create **custom events**. You can create and fire custom events using the **socket.emit** function.

For example, the following code emits an event called **testerEvent** −

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

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

io.on('connection', function(socket) {
   console.log('A user connected');

   //Send a message when
   setTimeout(function() {
      //Sending an object when emmiting an event
```

```
      socket.emit('testerEvent', { description: 'A custom event
named testerEvent!'});
   }, 4000);

   socket.on('disconnect', function () {
      console.log('A user disconnected');
   });
});

http.listen(3000, function() {
   console.log('listening on localhost:3000');
});
```

To handle this custom event on client we need a listener that listens for the event testerEvent. The following code handles this event on the client −

```
<!DOCTYPE html>
<html>
   <head>
      <title>Hello world</title>
   </head>
   <script src = "/socket.io/socket.io.js"></script>

   <script>
      var socket = io();
      socket.on('testerEvent',
function(data){document.write(data.description)});
   </script>
   <body>Hello world</body>
</html>
```

This will work in the same way as our previous example, with the event being testerEvent in this case. When you open your browser and go to localhost:3000, you'll be greeted with −

```
Hello world
```

After four seconds, this event will be fired and the browser will have the text changed to −

```
A custom event named testerEvent!
```

We can also emit events from the client. To emit an event from your client, use the emit function on the socket object.

```
<!DOCTYPE html>
<html>
   <head>
      <title>Hello world</title>
   </head>
   <script src = "/socket.io/socket.io.js"></script>

   <script>
      var socket = io();
```

```
      socket.emit('clientEvent', 'Sent an event from the
client!');
   </script>
   <body>Hello world</body>
</html>
```

To handle these events, use the **on function** on the socket object on your server.

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

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

io.on('connection', function(socket) {
   socket.on('clientEvent', function(data) {
      console.log(data);
   });
});

http.listen(3000, function() {
   console.log('listening on localhost:3000');
});
```

So, now if we go to localhost:3000, we will get a custom event called **clientEvent** fired. This event will be handled on the server by logging −

```
Sent an event from the client!
```