LEARN TO CODE FOR THE WEB

SIMPLE CHAT

SETUP

- Code editor : atom.io
- Node.js : nodejs.org
  - Test for node & npm : node –version
- Create working directory: e.g. chatApp
- Change to chatApp directory
- Open editor to chatApp directory (Atom : open project)

TEXT

CONFIGURE NODE SERVER

- npm init -y
- Creates package.json
- Open package.json in editor
- In terminal: npm install express - - save
- Inspect package.json

TEXT

CONFIGURE EXPRESS SERVER

- var app = require('express')();
- var http = require('http').Server(app);
- app.get('/', function(req, res){
- res.send('<h1>Hello world</h1>');
- });

- http.listen(4000, function(){
- console.log('listening on *:4000');
- });

TEST SERVER

- Open terminal and navigate to working directory (simpleChat)
- Check directory contents: ls -al
- Enter: “node server.js” : message “listening on”
- In browser type: localhost:4000
- ??

WEBPAGE

- In editor create new file: index.html
- Create basic web page:
- Headers
- Imports from CDN
- JQuery, Bootstrap CSS, Bootstrap.js

TEXT

MODIFY SERVER.JS

- var app = require('express')();
- var http = require('http').Server(app);
- var io = require('socket.io')(http);
- app.get('/', function(req, res){
- res.sendFile(__dirname + '/index.html');
- });

SERVE WEBPAGE

- Go to browser
- Enter: “localhost:4000”

HTTP

- HTTP is stateless protocol
  - After send or response, it does not remember transaction
- Yes, the webpages may know who you are but this is because of other code such as cookies
- High overhead for fast data transmission (opening and closing connections)

ADDING WEBSOCKET

- Connection initialized over http
- Request to upgrade to websocket if possible
- If upgraded, a duplex (2 way) communication channel is opened
- Both ends of the channel have “listeners” and “emitters”
- Enter message on webpage (index.html)

WEBSOCKETS IN ACTION

- User enters message on webpage (index.html) creating (“message”, package)
- Sent to server (server listening for “message”)
- Reads the received object and then emits the package to all listeners
- All listeners (users) see the messages appear in browser.

ADDING WS CODE

- In terminal: npm install socketio - -save
- In browser: go to socket.io
- Paste cdn code to index.html

MODIFY INDEX.HTML

```
<script>
  $(function() {
    var socket = io();


    $("#send").click(function() {
      var content = {
        name: $("#txtName").val(),
        chat: $("#messageTxt").val()
      };
      socket.emit('chatMessage', content);
    })
    socket.on('chatMessage', function(msg) {


      $('#messages').prepend($('<li>').text(msg.name + " : " + msg.chat));


    });
```

TEXT

MODIFY SERVER IMPORTS

- var app = require('express')();
- var http = require('http').Server(app);
- var io = require('socket.io')(http);
- app.get('/', function(req, res){
- res.sendFile(__dirname + '/index.html');
- });

MODIFY SERVER.JS

```
io.on('connection', function(socket){
  console.log('a user connected on :', socket.id);
});


io.on('connection', function(socket){
  socket.on('chatMessage', function(msg){
    console.log("chatMessage :", msg.name,msg.chat);
    io.emit('chatMessage', msg);
  });
});
```

PUTTING IT IN PLAY

- Check that server.js and index.html are saved
- Go to browser and refresh (localhost:4000)
- Open second page with the same address
- Chat between two users
- Check url of computer's WiFi connection (192.168.0.132)
- Change address bar in computer (192.168.0.XXX:4000)
- Class chat