EECS 4413 BUILDING e-COMMERCE SYSTEMS

SERVER-SIDE NODE

PROF H ROUMANI
Dept. of Electrical Engineering & Computer Science

1

NodeJS

- JavaScript Runtime
  The V8 engine is like the JVM
- Execution Model
  Single-Threaded but Asynchronous
- Architecture
  C/C++ packages managed by npm
- App Server
  We will use the Express package

2

2

Server

```
// imports:
const net = require('net');
const https = require('https');
const express = require('express');
const session = require('express-session');

// objects:
var app = express();
var sqlite3 = require('sqlite3').verbose();
var db = new sqlite3.Database(DB_PATH);

// SERVER
var server = app.listen(port, () =>
   { console.log('Listening...'); });
```

3

3

Service

```
// route and middleware:

app.use('/Test', function(req, res)
{
  res.writeHead(200, {'Content-Type': ...});
  res.write("Hello!");
  res.end("You sent me: " + req.query.x);
});


// can also do callback with arrow:
app.use('/Test', (req, res) =>
```

4

4

Persistence Layers

```
// The counterpart of Tomcat's session
app.use(session(
{
  secret: "mine",
  proxy: true,
  resave: true,
  saveUninitialized: true
}));

// Tomcat's get/set attribute
req.session.<attribute>

// Tomcat's context get/setAttribute
Use global var or db (JS is single-threaded)
```

5

5

TCP Client

```
// Create a tcp socket
var opt = {host:'...', port: ...};
var client = net.createConnection(opt, () =>
  { console.log('connected to TCP :)'); });

// Send a request
client.write('...');

// Listen for response or error
client.on('data', (data) => { use data });
client.on('error', (err) => { use err  });

// close the socket
client.end();
```

6

6

HTTP Client

```
https.get(url, (resp) =>
{
   let data = '';
   resp.on('data', (x) => {data += x;} );
   resp.on('end', () => { use data })
}).on("error", (err) => { use err });
```

7

7

Database Access

```
// given a query
db.all(query, function(err, rows)
{
   if (err == null)
   {
      use rows
   }
   else
   {
      use err
   }
});
```

See https://github.com/mapbox/node-sqlite3/wiki/API

8

8

JS and ECMAScript

- **Data Types**
  Number, String, Boolean, Null, Undefined, and Object
- **Dynamic Typing**
  Value-based coercion is done via toString, valueOf, and a number of *"intuitive"* conventions
- **Operators**
  Same as Java plus === except for coercion (you can, for example, use > on strings or dates)

9

9