Javascript, The Swiss Army Knife of Programming Languages

David Morcillo

30-11-2013

Introduction

We are going to do our first steps on a multiplayer game using Websockets. First, we need a web server to serve our web application (our game) and handle communications between all clients connected to the server.

Introduction

We are going to do our first steps on a multiplayer game using Websockets. First, we need a web server to serve our web application (our game) and handle communications between all clients connected to the server.

Express.js

Express is a minimal and flexible node.js web application framework, providing a robust set of features for building single and multi-page, and hybrid web applications.

Express.js: Hello World!

Installation

```
npm install express --save-dev
```

Express.js: Hello World!

Installation

```
npm install express --save-dev
```

Hello World

```
var express = require('express'),
    app     = express();

app.get('/hello.txt', function(req, res){
    var body = 'Hello World';
    res.setHeader('Content-Type', 'text/plain');
    res.setHeader('Content-Length', body.length);
    res.end(body);
});

app.listen(9000);
console.log('Listening on port 9000');
```

Express.js: Hello World!

Installation

```
npm install express --save-dev
```

Hello World

```
var express = require('express'),
    app     = express();

app.get('/hello.txt', function(req, res){
    var body = 'Hello World';
    res.setHeader('Content-Type', 'text/plain');
    res.setHeader('Content-Length', body.length);
    res.end(body);
});

app.listen(9000);
console.log('Listening on port 9000');
```

Run server

```
$ node index.js // Open browser and visit http://localhost:9000/hello.txt
```

Express.js: API

Serving static files with app.use

```
app.use(express.static(__dirname + '/public'));
```

Express.js: API

Serving static files with `app.use`

```
app.use(express.static(__dirname + '/public'));
```

Using a template engine system

Install a template engine system. For example `jade`:

```
$ npm install jade --save-dev
```

Use it on our Express application:

```
app.set("view engine", "jade");
app.set("views", __dirname + "/views");
```

Express.js: API

Defining routes and rendering views

```
app.get('/about', function (req, res) {
    res.render('about');
});

app.get('/credits', function (req, res) {
  res.render('credits', { name: 'test' }); // Pass parameters to the view
});

app.post('/players', function (req, res) {
    res.render('players/show');
});

app.put('/players', function (req, res) {
    res.render('players/show');
});
```

Express.js: API

Parse query parameters

```
// GET /search?q=nintendo
app.get('/search', function (req, res) {
    var q = req.query.q;
});
```

Express.js: API

Parse query parameters

```
// GET /search?q=nintendo
app.get('/search', function (req, res) {
    var q = req.query.q;
});
```

Parse body

First, we need to use `bodyParser` middleware.

```
app.use(express.bodyParser());
```

Then, we can parse body directly:

```
// POST /players player[name]=David
app.post('/players', function (req, res) {
    var playerData = req.body.player,
        playerName = playerData.name;
});
```

Express.js: Lab

- `git checkout stage_8_1`
- Install your back-end dependencies with `npm install`
- Install your front-end dependencies with `bower install`
- Start `grunt watch` for auto linting
- Find TODOs and complete the exercise.

Websockets

Without Websockets we have the limitation of unidirectional communication between server and client. We can emulate some kind of bidirectional communication using AJAX and polling but it's a poor option in real-time applications.

Websockets

Without Websockets we have the limitation of unidirectional communication between server and client. We can emulate some kind of bidirectional communication using AJAX and polling but it's a poor option in real-time applications.

Socket.io

Socket.IO aims to make realtime apps possible in every browser and mobile device, blurring the differences between the different transport mechanisms. It's care-free realtime 100% in JavaScript.

Socket.io: Hello World!

Installation

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

Socket.io: Hello World!

Installation

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

Server-side

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

  // code omitted

  io.sockets.on('connection', function (socket) {
      socket.emit('news', { hello: 'world' });
      socket.on('my other event', function (data) {
          console.log(data);
      });
  });

  // Replace app.listen with this
  server.listen(9000);
```

Socket.io: Hello World!

Require.js configuration

```
requirejs.config({
    // code omitted
    paths: {
        'io': '/socket.io/socket.io'
    }
});
```

Socket.io: Hello World!

Require.js configuration

```
requirejs.config({
    // code omitted
    paths: {
        'io': '/socket.io/socket.io'
    }
});
```

Client-side

```
var io     = require('io'),
    socket = io.connect();

socket.on('news', function (data) {
    console.log(data);
    socket.emit('my other event', { my: 'data' });
});
```

Socket.io: API

Send and receive messages

```
socket.on('message', function (data) {
    var player = data.player;
});

socket.emit('message', { player: 'player1' });
```

Socket.io: API

Send and receive messages

```
socket.on('message', function (data) {
    var player = data.player;
});

socket.emit('message', { player: 'player1' });
```

Broadcast messages

```
// On the server side
socket.broadcast.emit('player logout', { player: 'player1' });
```

Socket.io: API

Store information associated to a client

```
// On the server side
socket.set('playerId', 'player', function () {
    socket.emit('player saved');
});
```

Socket.io: API

Store information associated to a client

```
// On the server side
socket.set('playerId', 'player', function () {
    socket.emit('player saved');
});
```

Retrieve information associated to a client

```
// On the server side
socket.get('playerId', function (err, player) {
    socket.emit('player loaded', { player: player });
});
```

Socket.io: Lab

- `git checkout stage_8_2`
- Install your back-end dependencies with `npm install`
- Install your front-end dependencies with `bower install`
- Start `grunt watch` for auto linting
- Find TODOs and complete the exercise.