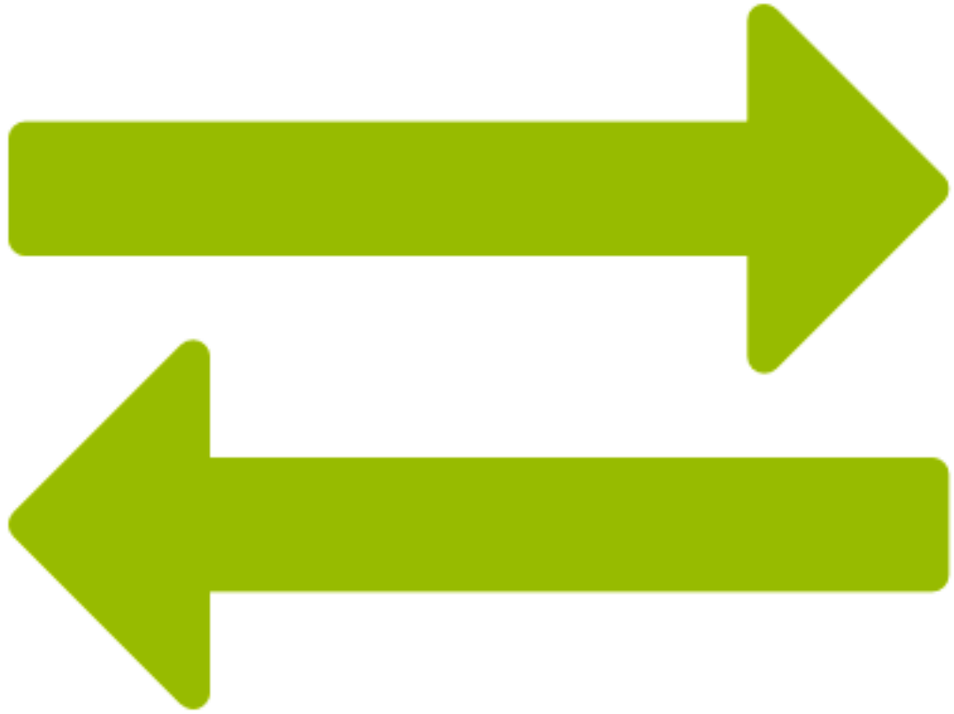
Tema 7

Socket.io



¿Qué es Socket.io?

- El protocolo HTTP fue concebido desde sus orígenes para ofrecer comunicaciones en un sólo sentido, desde el servidor hacia el cliente. Sin embargo las aplicaciones web de hoy en día demandan más que eso para poder ofrecer una experiencia de usuario más rica, necesitan flujo de información en ambos sentidos en el mismo instante en el que ocurren los eventos.

- **Socket.io es** una librería en JavaScript para Node.js que permite una **comunicación bidireccional en tiempo real entre cliente y servidor.** Para ello se basa principalmente en Websocket aunque ofrece otras posibilidades.

Socket.io

- Socket.io está formado por dos partes:
  - Un servidor que se integra en el servidor http de node.js (“socket.io”)
  - Una librería que se carga en el navegador del cliente (“socket.io-client”)
- Durante el desarrollo la librería del cliente se sirve automáticamente así que sólo nos tenemos que preocupar por instalar la parte del servidor.

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

Socket.io

- Una vez instalada la dependencia, tenemos que hacer un require en nuestro código para empezar a utilizar socket.io

```
var app = require('../app');
var debug = require('debug')('test:server');
var http = require('http');

var port = normalizePort(process.env.PORT || '3000');
app.set('port', port);

/**
 * Create HTTP server.
 */

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

server.listen(port);
```

Socket.io

- Para que nuestro servidor escuche un determinado evento, empleamos el método **on** pasando como argumento el identificador del mismo:

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

- Para conectar el cliente con el servidor, necesitamos añadir un script en la página desde la que se realizará la conexión:

```
<!– en html  –>
<script src="/socket.io/socket.io.js"></script>
<script>
  var socket = io();
</script>
```

```
// en jade
script(src="/socket.io/socket.io.js")
 script.
   var socket = io();
```

Escuchar eventos

- Los sockets tienen dos eventos especiales **‘connect’** y **‘disconnect’** que se gestionan automáticamente desde el cliente. Para escuchar el evento disconnect en el servidor procedemos de la misma forma:

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

Emitir eventos

- La idea principal de Socket.IO es que puedas recibir y enviar los eventos que quieras con los datos que desees. Socket.io es capaz de transmitir en objetos JSON y datos binarios.

- Para lanzar un evento empleamos el método **emit** de los sockets que necesita un identificador para el evento y los datos a transmitir.

```
socket.emit('chat message', "mymessage");
```

- Es posible lanzar y escuchar eventos de forma bidireccional entre cliente y servidor.

Broadcasting

- Desde el servidor tenemos la posibilidad de lanzar eventos a todos los sockets conectados con el método **emit.**

- Si queremos enviar un mensaje a todos los sockets excepto al que ha lanzado el evento, emplearemos el método **broadcast.emit**

```
io.on('connection', function(socket){
  socket.broadcast.emit('hi');
});
```

Namespaces

- Socket.io permite organizar los sockets en diferentes espacios de nombres de forma que sea posible crear varios canales de comunicación.
- Por defecto, el espacio de nombres de todos los sockets es /

- Para configurar un nuevo espacio de nombres empleamos la función of en el servidor. En el cliente especificamos el namespace como argumento.

```
//Conexión servidor
var nsp = io.of('/my-namespace');
nsp.on('connection', function(socket){
  console.log('someone connected'):
});
nsp.emit('hi', 'everyone!');
```

```
//Conexión cliente
var socket = io('/my-namespace');
```

Rooms

- Permiten definir canales dentro de un espacio de nombres .
- Los sockets pueden unirse (join) y salir(leave) de cada canal.
- El servidor es el único que puede asociar los sockets a las rooms.

```
//Asociar un socket a un canal/room
io.on('connection', function(socket){
  socket.join('some room');
});

//Mandar un mensaje a un canal/room
io.to('some room').emit('some event'):
```

Room por defecto

- Cada socket recibe un id aleatorio y único.
- Cada socket se une a un canal/room que recibe el nombre de su identificador.
- Sabiendo esto, resulta muy fácil enviar un mensaje a un determinado socket:

```
io.on('connection', function(socket){
  socket.on('say to someone', function(id, msg){
    socket.broadcast.to(id).emit('my message', msg);
  });
});
```

Mensajes del mundo exterior

- Si queremos enviar mensajes a sockets alojados en otros procesos necesitamos añadir dos módulos: socket.io-redis y socket.io-emitter

```
//Adaptador redis en el servidor
var io = require('socket.io')(3000);
var redis = require('socket.io-redis');
io.adapter(redis({ host: 'localhost', port: 6379 }));

//Mensaje desde otro proceso
var io = require('socket.io-emitter')();
setInterval(function(){
  io.emit('time', new Date);
}, 5000);
```