La cosa es asi. Las dos funciones mas importantes de socket.io (que si van a las paginas no es websockets idrectamente sino que utiliza websockets para hacer todo) es emit y on. En resumidas palabras:

emit: Envio un evento

on: Reacciono a un evento determinado

¿Un evento que seria? Cualquier cosa, ¿bien? Por ejemplo, tocar un boton, subir una imagen, etc, que los distinguis con un string como puede ser "table update" o "button clicked". Socket.io funciona en base a eventos, es decir, se mandan mensajes o se interactua con un servidor solamente por eventos. Si no hay eventos, no hay interaccion.

Primero hablaré un poco del servidor y luego un poco del cliente y luego voy a hacer como una "unión" entre ambos.

En el lado servidor hay que tener las siguientes lineas (como minimo, osea depende de lo que quieran hacer pero para hacerlo andar necesitas):

```
var express = require('express')
var socketIO = require('socket.io')
var portNumber = 4000
var socketSrv;
var application = express();
var server = application.listen(portNumber, function(){ console.log('Server is listening!!!') });
application.use(express.static('public'));
socketSrv = socketIO(server);
socketSrv.on('connection', (socket) => { } );
```

Explico pedacito por pedacito, cacho por cacho ahre:

var express = require('express')

var socketIO = require('socket.io')

Aca le decimos que modulos van a ser necesarios para este codigo y luego con las variables express y socketIO accedemos a los metodos (funciones mas que nada) de estos modulos.

var portNumber = 4000

var socketSrv;

El primero es evidente osea tomamos un numero de puerto que es donde el servidor va a estar escuchando. Tomen cualquier numero desde 1024 para arriba y luego en socketSvr guardaremos el objetos socket del servidor.

var application = express();

var server = application.listen(portNumber, function(){ COSAS PARA HACER });

Express por lo que lei es una libreria o modulo que te sirve para poder simplificar un monton de cosas en el diseño de una pagina web que interactua mucho con clientes. express te devuelve una 'aplicacion' que hace referencia al servidor y luego pones application.listen pasando como parametro una funcion (que la pueden definir como hice yo ahi) que se ejecuta al poner a la aplicacion en estado de listening (osea solo quiero imprimir un 'hola' al ponerlo en listening escribo adentro de los { } console.log('hola')) y tambien pasas como parametro el puerto en el que el servidor va a estar escuchando solicitudes.

application.use(express.static('public'));

socketSrv = socketIO(server);

Para no complicarla, basicamente, lo que lei es que use determina como un ruta de direccion para donde buscar los archivos para hacer funcionar la pagina web y basicamente le pasas el nombre de la carpeta en odnde se va a encontrar las fotos, el index.html etc. En el ejemplo que vimos con Nacho, el programa que corres en el node seria este que esta en la misma carpeta en que se encuenta la carpeta 'public'. Por ultimo, con socketIO conseguimos el socket del servidor.

socketSrv.on('connection', (socket) => { }); ->EL IMPORTANTISIMO

Con esto el servidor estaria funcionando. Ahora volvamos a las funcion **on** y **emit**. Como yo dije que on hace referencia a **como reaccionar a un estimulo, osea que hacer ante un evento determinado**, la conexion a un cliente hace referencia a un evento determinado que seria justamente el evento "connection". Luego de las lineas de "configuración", pasamos a esta linea de codigo (***socketSrv.on('connection', (socket) => { });***). Lo podemos leer asi:

"Ante la conexión de un cliente (evento connection) realizo una accion determinada ((socket) => { })"

Observar que como parametro se le pasa un socket, ¿ de quien ? El que está asignado al cliente (eso inteprete al leerlo). De este objeto socket podemos obtener ID (que seria lo que tenemos que obtener para el tp), etc. Nos da mucha informacion acerca del cliente.

Dentro de esta función, lo que habria que hacer es agregar mas "on"s y "emit". El cliente va a estar constantemente mandando eventos y el servidor va a tener que ser capaz de tomar estos eventos y viceversa. Como ejemplo...

socketSrv.on('connection', (socket) => {

socket.on("Boton Clickeado", *() => { console.log("El cliente toco un boton") }*)

socket.on("Archivo enviado", *(archivo) => { console.log("Se recibió archivo, devolverselo");*

socket.emit("Archivo devuelto", archivo) });

POR SUPUESTO QUE SE PUEDE PONER MAS

});

Puse dos "on" para ejemplificar con el emit. EL primero lo que hace simplemente es reaccionar ante un clic sobre un boton del cliente y imprime EN EL SERVIDOR "El cliente toco un boton". Veamos el segundo "on". El cliente manda un archivo (es lo que se recibe por parametro que ahora hablo con respecto a eso), el servidor imprime EN EL SERVIDOR *"Se recibió archivo, devolverselo"* y luego hace un "emit" al cliente. Este "emit" manda el evento "Archivo Devuelto" y le tira el archivo.

Aca esta lo importante. Como parametro, podemos mandar objetos, estrucuturas y tipo de datos "basicos". Si yo por ejemplo, tanto en el servidor como en el cliente tengo

on("Evento A", PARAMETRO1, PARAMETRO2, ... PARAMETRON)

tengo que tener

emit("Evento A", PARAMETRO1, PARAMETRO2, ... PARAMETRON)

Se respetan los parametros. NO ESTOY SEGURO SI PODEMOS PASAR MAS DE UN PARAMETRO, PERO LA IDEA ESTA EN ENTEDER QUE LO QUE YO MANDO POR EL EMIT, EN EL ON TIENE QUE HABER MANERA DE MANEJARLO.

Al hacerlo vi que a veces para mandar datos hay que crear objetos pero para lo que tenemos que hacer string o no pasar nada esta mas que bien.

Tomando el ejemplo que mostré, en el cliente deberia aparecer esto:

socket.on("Archivo enviado", *(archivo) => { console.log("Se recibió archivo, devolverselo");*

socket.emit"Archivo devuelto", archivo) });

Asi, el cliente puede reaccionar ante este evento.

Con respecto a lo del boradcast, podemos usar socket.broadcast.emit('new client', client); que es uan que use en el TP. Le mandamos a todos excepto a mi mismo ese evento. Si quiero que ciertos tipos de usuarios no intepreten ese evento se lo sacas del codigo y listo. hice eso y funca

perfecto.

Una cosa, vos determinas los eventos como quieras. Si quiere crear un evento "Hola como te va" hacelo pero no es muy claro. Te deja esa libertad socket.io. Los unicos que hay que respetar son 'connection' y 'disconnection'. Una desconexión la provocas desde el servidor o desde el cliente con el metodo close().

¿Donde invocar on y emit en el cliente? SI este es tu codigo

function hola(){ ... }

function pepito(){ ... }

function hola(){ ... }

function don_jose(){ ... }

Bueno pones los pones en donde indico a continuación:

function hola(){ ... }

function pepito(){ ... }

function hola(){ ... }

function don_jose(){ ... }

io.on(...)

io.on(...)

Los emit siempre los generas en algunas de las funciones de arriba, osea desde hola hasta don_jose. Es ismplemente mandar un evento y el servidor se fija como reaccionar.

Espero que haya sido ayuda. :)