Buena pregunta 👌 Si querés **notificar a todos los usuarios conectados cuando un nuevo usuario entra**, podés hacerlo así:

**🔧 1. En el servidor (socket.on('sesionIniciada', ...))**

Cuando un usuario inicia sesión, hacés un broadcast para avisar al resto:

js

CopiarEditar

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

socket.on('sesionIniciada', (userInfo) => {

console.log(`${userInfo.username} se ha conectado`);

// Avisar a todos los demás (excepto al que se conectó)

socket.broadcast.emit('usuarioConectado', {

username: userInfo.username,

userId: userInfo.id,

socketId: socket.id

});

});

});

🔁 socket.broadcast.emit(...) emite a **todos excepto** al socket que originó el evento.

**🧠 2. En el cliente (socket.on('usuarioConectado'))**

Escuchás el evento y mostrás una notificación (puede ser un toast, una alerta, o lo que prefieras):

js

CopiarEditar

useEffect(() => {

socket.on('usuarioConectado', (data) => {

console.log(`${data.username} está ahora en línea`);

// Ejemplo usando un toast

toast.info(`${data.username} se conectó`);

});

return () => {

socket.off('usuarioConectado');

};

}, []);

**⚠️ Si querés notificar también al usuario que él mismo se conectó, podés usar io.emit(...) en vez de socket.broadcast.emit(...).**

### 🔹 io → en el ****servidor****

Es la **instancia principal del servidor de Socket.io**.  
Con io podés emitir eventos a **todos los clientes conectados**, o a un grupo/sala.

js

CopiarEditar

// Emitir a todos los sockets conectados (incluye el emisor)

io.emit('evento', data);

// Emitir solo a todos MENOS el emisor actual

socket.broadcast.emit('evento', data);

// Emitir a un socket específico (por id)

io.to(socketId).emit('evento', data);

### 🔸 socket → tanto en ****cliente como en servidor****

Representa **una conexión individual**.  
Es con lo que se maneja la **comunicación con un solo cliente**.

* En el **servidor**, lo usás dentro del io.on('connection', socket => {...}) para interactuar con ese socket puntual.
* En el **cliente**, socket es la conexión entre **ese cliente y el servidor**.

js

CopiarEditar

// Cliente: escuchar evento

socket.on('evento', (data) => {...});

// Cliente: emitir al servidor

socket.emit('evento', payload);

// Servidor: emitir solo al que se acaba de conectar

socket.emit('evento', payload);

### Ejemplo resumido:

**Servidor:**

js

CopiarEditar

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

// Solo a este cliente

socket.emit('soloVos', "Hola!");

// A todos menos este cliente

socket.broadcast.emit('nuevoUsuario', "Alguien se conectó!");

// A todos, incluyendo este cliente

io.emit('notificacionGlobal', "Nuevo usuario online");

});

**Cliente:**

js

CopiarEditar

socket.on('notificacionGlobal', (msg) => {

console.log(msg);

});

**Contexto general:**

Cuando hacés esto en el **servidor**:

js

CopiarEditar

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

...

})

Estás diciendo:  
➡️ "Cada vez que **alguien se conecte**, ejecutá esta función y dame acceso al socket de ese usuario en particular."

**Entonces, ¿por qué usamos socket y luego io?**

| **Código** | **¿A quién afecta?** | **¿Para qué sirve?** |
| --- | --- | --- |
| socket.emit(...) | 🔸 **Solo al usuario que se acaba de conectar** | Le mandás un mensaje exclusivo |
| socket.broadcast.emit(...) | 🔹 **A todos los demás conectados, excepto al nuevo** | Para avisar que alguien se unió, pero sin avisarle al que recién llegó |
| io.emit(...) | 🌐 **A todos, incluyendo al nuevo** | Para mandar una notificación global |

**✅ Ejemplo realista:**

js

CopiarEditar

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

// A este nuevo usuario le damos la bienvenida

socket.emit('soloVos', '¡Bienvenido a la app!');

// A todos los demás les avisamos que alguien más se conectó

socket.broadcast.emit('nuevoUsuario', 'Un nuevo usuario se conectó');

// A todos, incluido el nuevo, les decimos que hay un nuevo usuario online

io.emit('notificacionGlobal', 'Actualización: alguien más está online');

});

**🧠 En resumen:**

* socket.emit → **a uno solo** (el que se conectó)
* socket.broadcast.emit → **a todos menos ese**
* io.emit → **a todos**