**WebSocket io**

* WebSocket is a computer communication protocol, which provides communication channels over a single TCP (Transmission Control Protocol) connection
* Real Time Communication from server to browser and reverse
* Bidirectional communication

**Socket io Segment**

* There are two segment of socket io   
  -> Server   
  -> client



**Socket in Express**

* We can setup socket in many ways , here we are now going to setup socket in express
* First create a complete express server
* **Then install socket io**  
  npm install socket.io
* **Now convert express server to a socket server**  
  const { createServer } = require(‘http’);   
  const { Server } = require(‘socket.io’);   
    
  // now catch express server in http   
  const httpServer = createServer(app);  
    
  // now init socket server   
  const io = new Server(httpServer);

**Socket in Client**

* Now we are going to setup socket in client
* To setup socket in client, we can use cdn or npm
* **Now by using cdn just use this link**   
  <script src="https://cdn.socket.io/4.5.3/socket.io.min.js" ></script>
* **Socket include from stand alone process froma server**   
  <script src="/socket.io/socket.io.js"></script>

<script>

const socket = io();

</script>

**Connection Setup**

* Now we need to setup client connection from socket server
* For that now first create a connection from socket instanc   
  io.on( ‘connection’ , (socket) => {  
   console.log(‘client is connected’);  
   socket.on( ‘disconnect’ , () => {  
   console.log(‘client is disconnected’);  
   }):  
  });

**Server to Client Data**

* For apps requirements we need to send data from server to client and client to server
* For that we can send it from server like this   
  io.on( ‘connection’ , (socket) => {  
   console.log(‘client is connected’);  
   socket.send( ‘I love Socket );  
   socket.emit( ‘emitName’, ‘I love Socket’ );  
  });
* Now receive data from client   
  <script src="/socket.io/socket.io.js"></script>

<script>

const socket = io();  
 // from send event

socket.on(‘message’, (data) => {  
 console.log(data);  
});  
  
// from emit event

socket.on(emitName, (data) => {  
 console.log(data);  
});

* </script>

**Emitting multiple**

* We can also send multiple data in emitting   
  io.on( ‘connection’ , (socket) => {  
   console.log(‘client is connected’);  
   socket.emit( ‘emitName’, ‘agr1’, ‘agr2’, ‘agr3’ );  
  });
* Receive multiple data from client   
  socket.on(emitName, (data1, data2, data3) => {  
   console.log(data);  
  });

**Socket events**

* Connect
* Disconnect
* Reconnect
* Message
* Ping
* Join
* Leave

**Brodcusting**

* If we want to update all clients data after changing one client user interact then we need to use socket brodcusting
* Now create a brodcusting   
  io.on( ‘connection’ , (socket) => {  
   console.log(‘client is connected’);  
   io.sockets.emit( ‘emitName’, ‘agr1’, ‘agr2’, ‘agr3’ );  
  });

**Namechace**

* We can create many connection under namespace
* Namespace differ the socket endpoint
* To create a namespace   
  const io = new Server(httpServer);  
  const personalChat = io.of(‘/personal’);  
  const groupChat = io.of(‘/group);

**Rooms**

* A room is a arbitrary channel that can join and leave. It can be used to broadcast events to a subset of clients
* Room is used to broadcast an events with many clients
* To join a room   
  io.on( ‘connection’ , (socket) => {  
   console.log(‘client is connected’);  
   socket.join(‘room-name-1’):  
   io.sockets.in(‘room-name).emit(‘eventName’,’data’);  
    
   socket.join(‘room-name-2’):  
   io.sockets.in(‘room-name).emit(‘eventName’,’data’);  
    
  });
* Get a room size ( joined member numbers )  
  io.on( ‘connection’ , (socket) => {  
   console.log(‘client is connected’);  
   socket.join(‘room-name-1’):  
   let memberCount = io.sockets.adapter.rooms.get(‘room-name’).size;  
   io.sockets.in(‘room-name).emit(‘eventName’,’data’);  
    
   socket.join(‘room-name-2’):  
   io.sockets.in(‘room-name).emit(‘eventName’,’data’);  
    
  });