Websockets

Building a chat application

Lesson Plan

- what are websockets
- how connection is made
- Websocket server
- websocket connection
- client side connection
- send message client to server and server to client
- keep alive
- security concerns

Lesson summary

https://www.nerdeez.com/articles/node/express-websockets

What are websockets

- full duplex communication protocol
- depends on TCP
- used for real time data transfer between client to server and server to client
- connection is left open
- Websocket define the following URI schemas: ws:// or wss://

How connection is made

- first phase - opening handshake
  - based on http
  - I want to open a websocket
  - I want this protocol version
  - I don’t want any caching so here is some random encoded string
  - I want this subprotocol
- second phase - exchange data
  - message based
  - message can be text or binary
  - long messages are splitted across multiple frames
- third phase - close connection
  - gracefully close by sending a close message
  - the connection will also be closed when exiting the session
  - there could be an error when connection is open in the server and not responding - keepalive is necessary

websocket server

- install a package called **ws**
- create an instance of **Websocket.Server** which will need to port to listen
- this class inherits from EventEmitter
- has the following events
  - **listening -** server starts to listen
  - **headers -** on handshake
  - **connection** - when we have a new connection the event will get a **WebSocket** object
  - **error**

WebSocket object

- describes a single connection
- we will get it on the Websocket.Server connection event
- extends EventEmitter
  - **message -** when a message is recieved
  - **open/close**
  - **error**
  - **pong**
- method **send** will send a message to the client

Client side websocket

- **WebSocket** class
- to create a websocket we create instance of that class with the url of the websocket
- the instance also emits events
  - **onerror**
  - **onopen**
  - **onmessage - event.data** will hold the message

Create a chat application

- lets build a group chat application
- when a single user posts a message we will push that message to all the people in the chat

keepalive

- there are cases where the link between client and server is interrupted but the server and the client are unaware of the broken state of the connection
- it’s best practice to periodically send a ping message from the server to make sure the connection is still active
- there is a special message type for that case called ping which the client will automatically respond with a pong message
- on every connection we will maintain a **isAlive** flag and every minute we will send a ping message if pong is not accepted until the next minute we will close the connection.
- not maintaining a keep alive might cause a memory leakage
- Lets add a keep alive to our chat application.

Security

- make sure that you use TLS encryption and send the message on **wss**
- make sure you validate the data on the client side and the server side
- make sure you maintain a token authentication system

Summary

when we are interested in real time communication for games or chat we will use websockets.

We saw in this lesson how extremely easy it is to set up websockets with node.