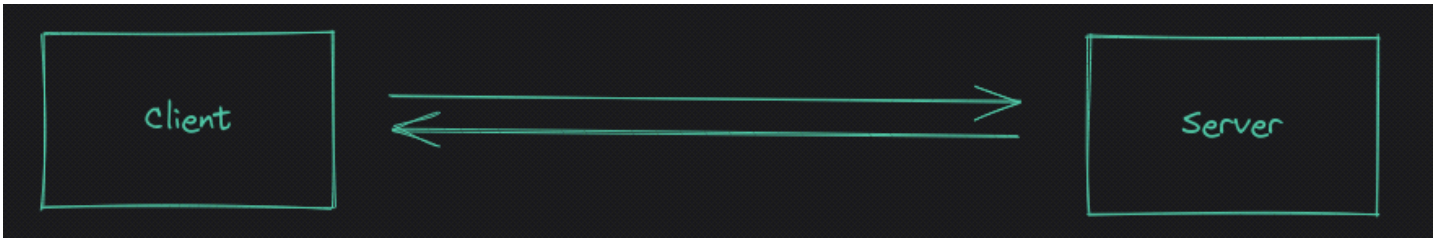
Socket.io

21 May 2025 08:02



Request-respond cycle.

Client request to some data and server response to it accordingly (it gives data if it has or empty response).

Long polling: client send request to server , takes time to search that data and if it has requirements server will send data to client(frontend) .
Long polling is like having a sustained conversation where the server holds the line until it has something to share, avoiding constant check-ins.

When to Use Long Polling? Long polling shines when real-time updates matter, and you want to minimize unnecessary requests. Imagine a messaging app where you don't want to miss a message but also don't want to keep asking, "Any new messages?"

The server holds the connection open until new data arrives.

Short polling:
Short polling is ideal for scenarios where real-time updates aren't critical. Think of a weather app refreshing every few minutes; short polling suits these cases well.

Web Socket: establish a persistent, two-way connection between the client and server.
its like a real-time ,bidirectional conversation.
Both the client and server can send data whenever they want, without the need for constant polling.

Upgrade header

The HTTP upgrade request and response header can be used to upgrade an already-established client/server connection to a different protocol (over the same transport protocol).For example,

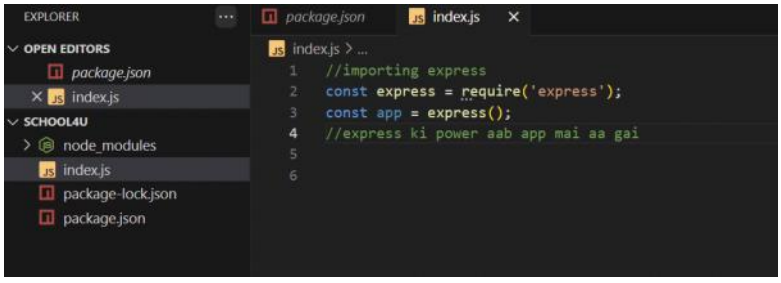
it can be used by a client to upgrade a connection from HTTP/1.1 to HTTP/2 or an HTTP(S) connection to WebSocket connection.(telling that now server should not break the connection after sending the response.

Understanding via Project(School4U)

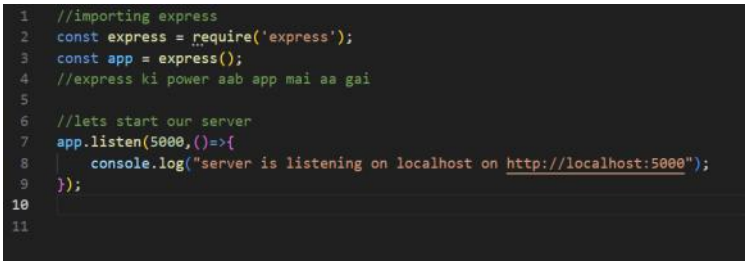
30 May 2025 23:53

-first I have created a folder name:School4u and put command in terminal as:
npm i express
Meaning creating a server with express

Now creating a index.js and importing express inside it! This is our simple server.

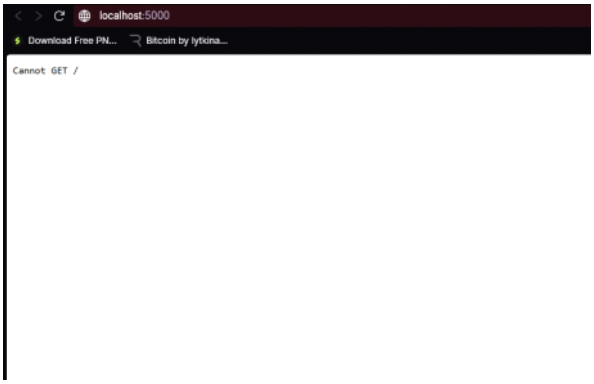


Now with listen command we will start our server!



Lets run this with command :
node index.js

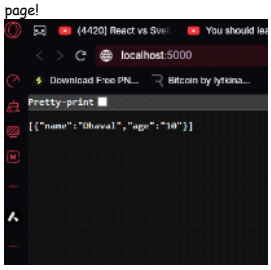
Now when we will visit this web port we will get this statement because we haven't yet made any APIs to call
Meaning GET request abhi banai hi nai hai



So from 7th line you can see we have created get request.

```
//importing express
const express = require('express');
const app = express();
//express ki power aab app mai aa gai


app.get('/',(req,res,next)=>{
    res.json([
        {
            name:"Dhaval",
            age:"10",
        }
    ])
})


//lets start our server
app.listen(5000,()=>{
    console.log("server is listening on localhost on http://localhost:5000");
});
```

So is the output after restarting node index.js and reloading localhost page!



Now we have installed dev dependency(-D) meaning production ke time kuch asar nai padega! here nodemon will help us to auto restart server so we don't have to!

-write this in command ternimal

```
E:\Website\WebSocket\socketTutorials\School4u>npm i -D nodemon
```

And after that type this in package.json file:

```
{
    Debug
  "scripts": {
    "start": "nodemon index.js"
  },
  "dependencies": {
    "express": "^5.1.0"
  },
  "devDependencies": {
    "nodemon": "^3.1.10"
  }
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

_837Z-debug-0.log

E:\Website\WebSocket\socketTutorials\School4u>npm start

> start
> nodemon index.js

[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node index.js`
server is listening on localhost on http://localhost:5000
```

So now we can start our server with npm start

Now lets make realtime web !

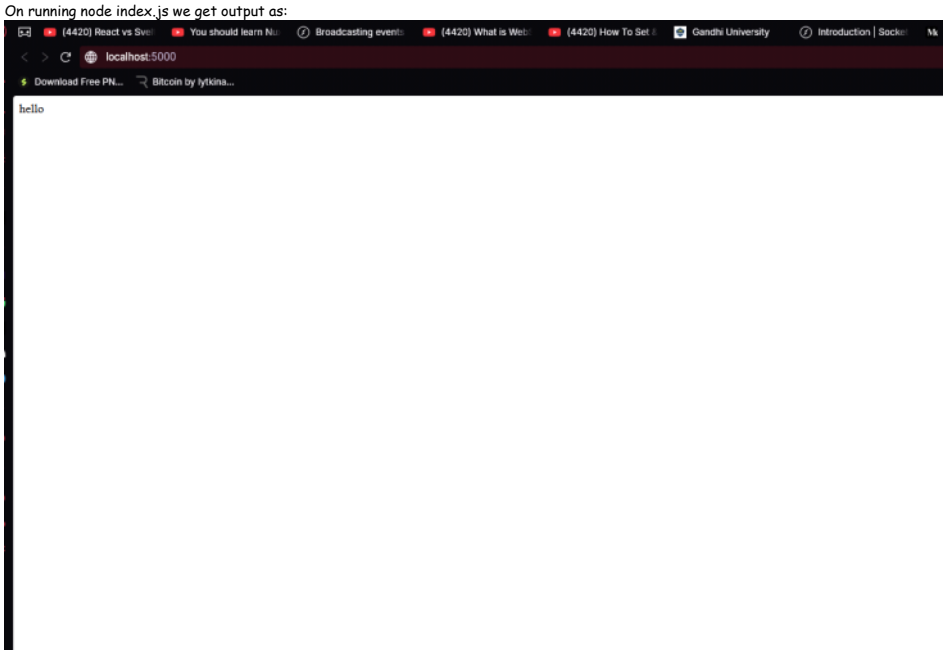
```
index.js
index.html
index.js > app.get('/') callback
1   const express = require('express'); //importing express
2   const http = require('http');//built-in module / library to initialize http
3
4   const app = express();//express ki power aab app mai aa gai
5   const server = http.createServer(app);
6
7
8   app.get('/',(req,res,next)=>{
9     res.send("hello")
10  })
11
12
13  //lets start our server
14  server.listen(5000,()=>{
15      console.log("server is listening on localhost on http://localhost:5000");
16  });
17
18
```

Here we have modified our previous code and added server to create Server so that we can upgrade our web socket in it!

Now we have created a simple front end to check our real time app:

```
EXPLORER
> OPEN EDITORS
v SCHOOL4U
  > node_modules
  v public
    index.html
    index.js
    package-lock.json
    package.json

index.js
index.html
public > index.html > html > body > div
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Chat app</title>
7   </head>
8   <body>
9       <div>Hello Real World!</div>
10  </body>
11  </html>
```

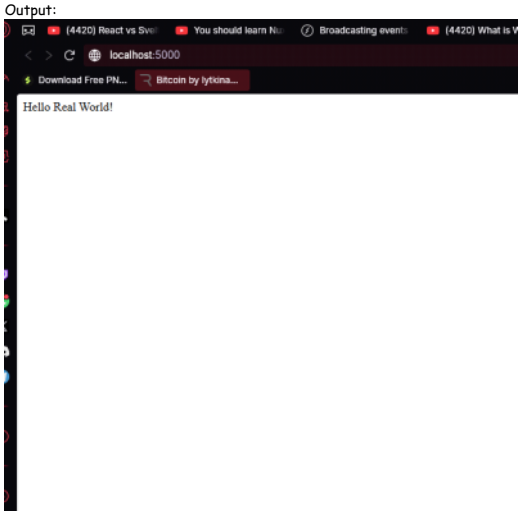
On running node index.js we get output as:



so we have to add our static index.html file!
now we can't just write relative path!
we need absolute path!

```
index.js > ...
const express = require('express'); //importing express
const http = require('http');//built-in module / library to initialize http
const path = require('path');//initiaziling for absolute path

const app = express();//express ki power aab app mai aa gai
const server = http.createServer(app);

console.log();

app.get('/',(req,res,next)=>{
//   res.send("hello");
   //syntax to add static index.html
  res.sendFile(path.join(__dirname,'public/index.html'));// __dirname is current file and second one is relative path of index.html
})


//lets start our server
server.listen(5000,()=>{
    console.log("server is listening on localhost on http://localhost:5000");
});
```

Output:



We can do it this way to:

```
index.js > ...
const express = require('express'); //importing express
const http = require('http');//built-in module / library to initialize http
const path = require('path');//initiaziling for absolute path

const app = express();//express ki power aab app mai aa gai
const server = http.createServer(app);
app.use(express.static(path.resolve('./public')))
app.get('/',(req,res,next)=>{
//   res.send("hello");
   //syntax to add static index.html
  //res.sendFile(path.join(__dirname,'public/index.html'));// __dirname is current file and second one is relative path of index.html

    res.sendFile('index.html');
})


//lets start our server
server.listen(5000,()=>{
    console.log("server is listening on localhost on http://localhost:5000");
});
```

Now lets install socket.io module:

```
E:\Website\WebSocket\socketTutorials\School4u>npm install socket.io
```

```
const express = require('express'); //importing express
const http = require('http');//built-in module / library to initialize http
const path = require('path');//initiaziling for absolute path
const { Server } = require("socket.io");//this is Socket.io server


const app = express();//express ki power aab app mai aa gai
const server = http.createServer(app);//This is our created server

const io = new Server(server);// putting our created server inside socket.io 's Server


app.use(express.static(path.resolve('./public')))
app.get('/',(req,res,next)=>{
//   res.send("hello");
   //syntax to add static index.html
  //res.sendFile(path.join(__dirname,'public/index.html'));// __dirname is current file and second one is relative path of index.html

    res.sendFile('index.html');
})


//lets start our server
server.listen(5000,()=>{
    console.log("server is listening on localhost on http://localhost:5000");
```

Now we have initialized socket.io Server(line 3) and put our own server inside it(line 10)
Meaning aab ho gaya aapna server socket.io se attached!

Now,
- with app(line 7)we will create api and modify server
- with io(line 10) we will work on real time connection

Now with the help of io we will turn on connection with socket.io

```
22
23  io.on("connection", (socket) => {
24    console.log('a user connected');
25  });
26
```

We have to also add script file inside our frontend like:

```
public > index.html > html > body > script
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Chat app</title>
7   </head>
8   <body>
9       <div>Hello Real World!</div>
10
11      <script src="/socket.io/socket.io.js"></script>
12      <script>
13          const socket = io();
14      </script>
15
16  </body>
17  </html>
```

Code:

```
<script src="/socket.io/socket.io.js"></script>
<script>
  const socket = io()
</script>
```

Now when I reload my web page I get connected message in our terminal:

```
E:\Website\WebSocket\socketTutorials\School4u>npm start

> start
> nodemon index.js

[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node index.js`
server is listening on localhost on http://localhost:5000
a user connected
```

now here is how we will send data to the backend(line 16);

```
index.js   index.html   package.json
public > index.html > html > body > script
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Chat app</title>
7   </head>
8   <body>
9       <div>Hello Real World!</div>
10
11      <script src="/socket.io/socket.io.js"></script>
12      <script>
13          const socket = io();
14
15          //we use emit method which sends data to our backend
16          socket.emit("msgFromFrontend","Hello I am from frontend");
17      </script>
18
19  </body>
20  </html>
```

Now to receive data we will use on function(line 27) :

```
index.js > io.on("connection") callback
13  app.use(express.static(path.resolve('./public')))
14  app.get('/',(req,res,next)=>{
15  //   res.send("hello");
16     //syntax to add static index.html
17    //res.sendFile(path.join(__dirname,'public/index.html'));// __dirname is current file and second one is relative path of index
18
19      res.sendFile('index.html');
20  })
21
22
23  io.on("connection", (socket) => {
24    console.log('a user connected',socket.id);//every user has unique socket id
25
26  //we use on function to recieve data from frontend!
27    socket.on('msgFromFrontend',(msg)=>{
28      console.log(msg);
29    }
30  });
31
32  //lets start our server
33  server.listen(5000,()=>{
34      console.log("server is listening on localhost on http://localhost:5000");
35  });
36
37
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
server is listening on localhost on http://localhost:5000
a user connected gFadWOrbML6yi7RFAAAB
a user connected zcboWt7zpWwL5RNTAAAD
Hello I am from frontend
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
server is listening on localhost on http://localhost:5000
a user connected cVWcMg49SCmyEDwhAAAB
a user connected 8wn5IbP9vbwDWAaiAAAD
Hello I am from frontend
```

Same thing If we want to send msg from backend:

```
index.html > html > body > script
2   <html lang="en">
7   </head>
8   <body>
9       <div>Hello Real World!</div>
10
11      <script src="/socket.io/socket.io.js"></script>
12      <script>
13          const socket = io();
14
15          //we use emit method which sends data to our backend
16          socket.emit("msgFromFrontend","Hello I am from frontend");
17
18
19          socket.on("msgFromBackend",(msg)=>{
20              console.log(msg);
21          })
22      </script>
23
24  </body>
25  </html>
```

```
index.js > io.on("connection") callback > socket.on("msgFromFrontend") callback
13  app.use(express.static(path.resolve('./public')))
14  app.get('/',(req,res,next)=>{
15  //   res.send("hello");
16     //syntax to add static index.html
17    //res.sendFile(path.join(__dirname,'public/index.html'));// __dir
18
19      res.sendFile('index.html');
20  })
21
22
23  io.on("connection", (socket) => {
24    console.log('a user connected',socket.id);//every user has unique
25
26  //we use on function to recieve data from frontend!
27    socket.on('msgFromFrontend',(msg)=>{
28      console.log(msg);
29      io.emit("msgFromBackend","Hello i am msg from backend");
30    }
31  });
32
33  //lets start our server
34  server.listen(5000,()=>{
35      console.log("server is listening on localhost on http://localho
36  });
37
```

here is our final output: