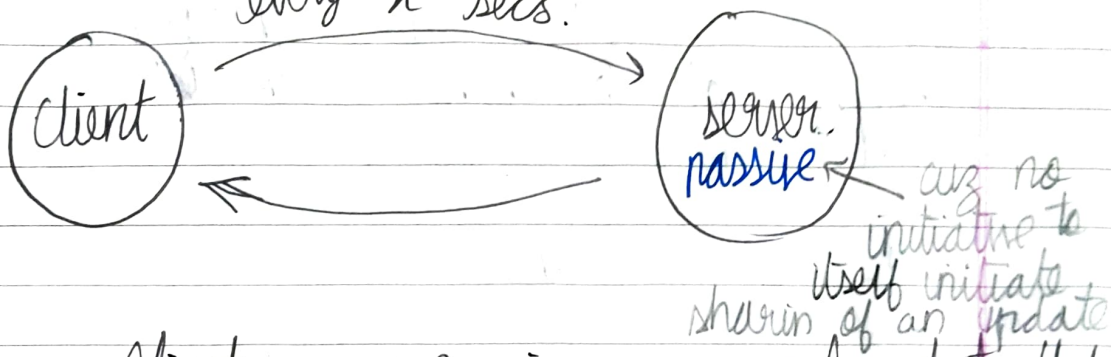
Polling & Streaming

Both deal with client getting some live updated data from server (eg:- temp, texts)

POLLING (Client talks (issues req) - Server listens (fulfils req))



Polling : Client is gonna issue a req. for data that it requires on a recurring basis following a set interval.

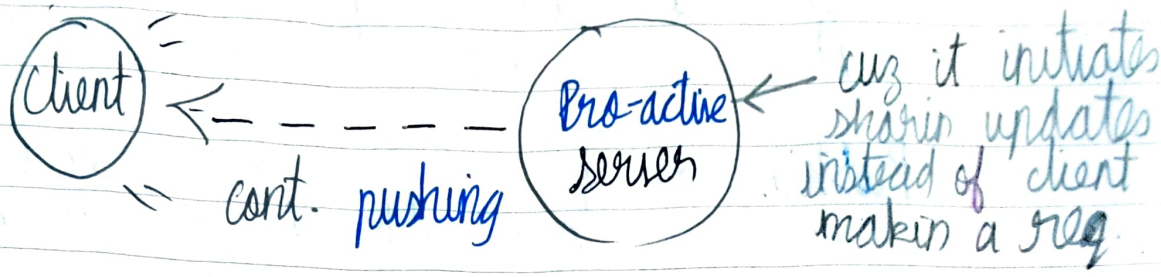
eg:- temp. → enables clients to monitor outside temp. and this temp.'s details are stored in our servers but it changes freq. ly, then the clients poll for temp. every, say, 30 sec, 1 min etc.

Limitations → No instantaneous user experience as updates are visib only after a fixed interval

eg:- can't use pollin for a chat app cuz
- ↳ text updates only after a fixed interval
- ↳ if we tryna ↓ dis fixed interval, to say 0.1 s, we get 10 reqs from each client to server per sec. If we have $> 10^4$ clients, server's clogged

Date:
M T W T F S S

STREAMING (Client listens (for updates) - Server talks (pushes updates))



Instead of issuin reqs, client opens a long lived connec^n with server thru a socket

Socket →
- a file livin on your PC, that PC can write to / read frm to comm. with another PC in a long lived connec^n manner
- portal to another machine that you can reach thru / comm. thru to enable comm.^{n} b/w 2 machines w/o repeatedly send req.
- open connec^n as long as
  - one of the machine closes connec^n
  - network connec^n is healthy.

∴ Here, a client will open a long lived connec^n with the server, thru a socket and listen to the server for any data that server might push to client thru socket.

Client → Client is streamin data frm server. will basically just listen for data (no req.) as long as connec^n is intact

Server is proactive → sends data to client as a push whenever it has an update
∴ logic is implemented on server side.

Pushin → server proactively sendin to data to client ie. it doesn't wait for a req. to do so

Streamin allows sys to hv a cont stream of data so long as server pushes the data updates it receives → instantaneously and correctly

Streamin provides instantaneous experience w/o hvin to issue too many (or any if we're bein real here) to the server.

Pollin	Streamin
Client talks (issues reqs.)	Client listens (collects updates)
Server listens (fulfils reqs.)	Server talks (pushes updates)
Broken stream of data at regular intervals. Req-resp. relat'ship b/w client-server	Cont. stream of data. Open connec" relat'ship b/w client-server.
use: temp. updates, stock updates after each hr (no live-tradin)	use: chat apps, currency exchange updates

LINK TO VIDEO OF CHAT APP POLL-STREAM EXAMPLE

polling_and_streaming — node server.js — 93×24

~/Documents/Content/Design_Fundamentals/Examples/polling_and_streaming — node server.js

```
Clements-MBP:polling_and_streaming clementmihailescu$ node server.js
Listening on port 3001!
```

polling_and_streaming — node client.js — 93×24

~/Documents/Content/Design_Fundamentals/Examples/polling_and_streaming — node client.js

```
Clements-MBP:polling_and_streaming clementmihailescu$ (for i in `seq 1 10000`; do sleep 1; echo $i; done) | NAME=Bot node client.js
```

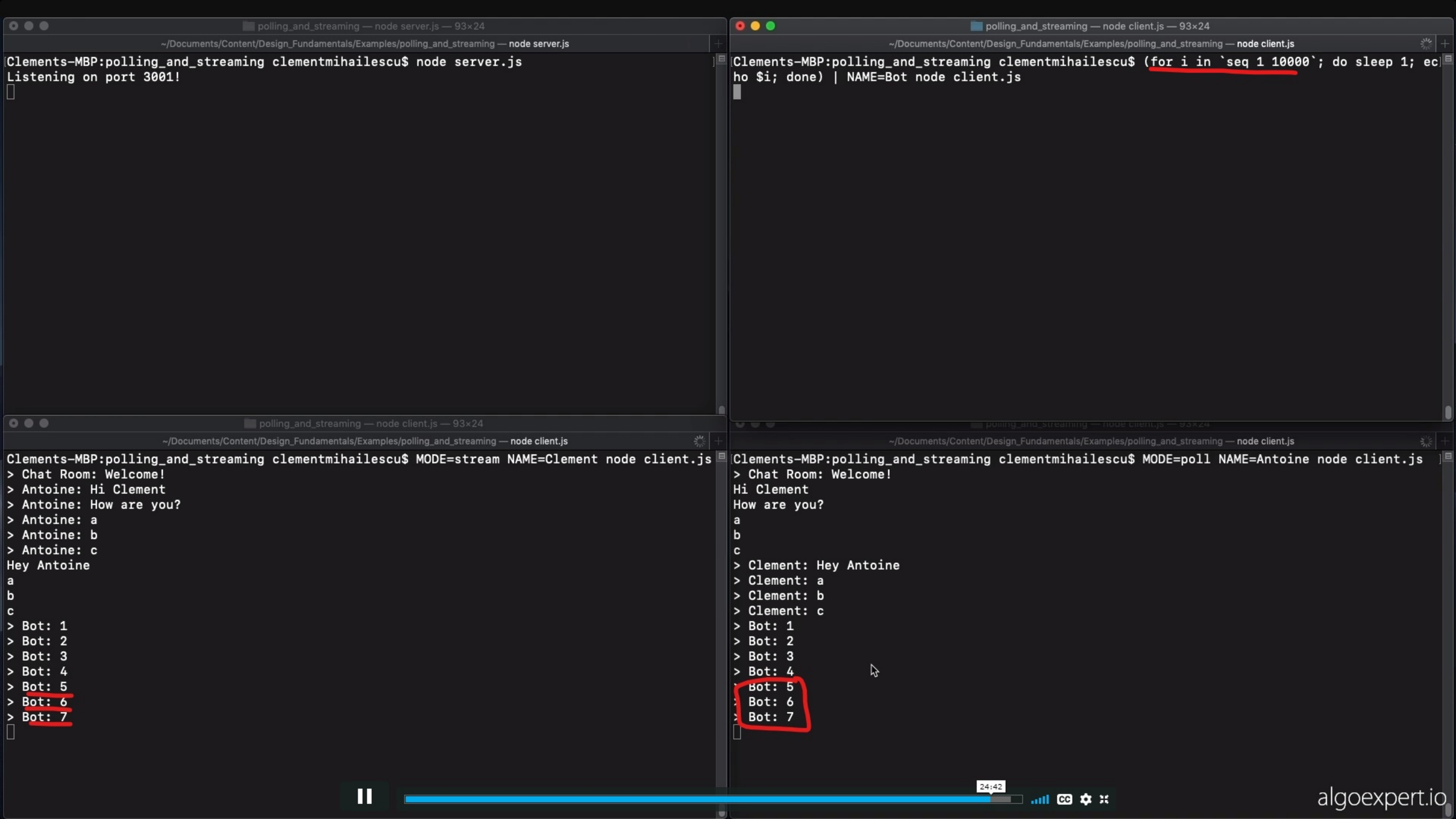
polling_and_streaming — node client.js — 93×24

~/Documents/Content/Design_Fundamentals/Examples/polling_and_streaming — node client.js

```
Clements-MBP:polling_and_streaming clementmihailescu$ MODE=stream NAME=Clement node client.js
> Chat Room: Welcome!
> Antoine: Hi Clement
> Antoine: How are you?
> Antoine: a
> Antoine: b
> Antoine: c
Hey Antoine
a
b
c
> Bot: 1
> Bot: 2
> Bot: 3
> Bot: 4
> Bot: 5
> Bot: 6
```

~/Documents/Content/Design_Fundamentals/Examples/polling_and_streaming — node client.js

```
Clements-MBP:polling_and_streaming clementmihailescu$ MODE=poll NAME=Antoine node client.js
> Chat Room: Welcome!
Hi Clement
How are you?
a
b
c
> Clement: Hey Antoine
> Clement: a
> Clement: b
> Clement: c
> Bot: 1
> Bot: 2
> Bot: 3
> Bot: 4
```


polling_and_streaming — node server.js — 93×24
~/Documents/Content/Design_Fundamentals/Examples/polling_and_streaming — node server.js
Clements-MBP:polling_and_streaming clementmihailescu$ node server.js
Listening on port 3001!
polling_and_streaming — node client.js — 93×24
~/Documents/Content/Design_Fundamentals/Examples/polling_and_streaming — node client.js
Clements-MBP:polling_and_streaming clementmihailescu$ (for i in `seq 1 10000`; do sleep 1; echo $i; done) | NAME=Bot node client.js
polling_and_streaming — node client.js — 93×24
~/Documents/Content/Design_Fundamentals/Examples/polling_and_streaming — node client.js
Clements-MBP:polling_and_streaming clementmihailescu$ MODE=stream NAME=Clement node client.js
> Chat Room: Welcome!
> Antoine: Hi Clement
> Antoine: How are you?
> Antoine: a
> Antoine: b
> Antoine: c
Hey Antoine
a
b
c
> Bot: 1
> Bot: 2
> Bot: 3
> Bot: 4
> Bot: 5
> Bot: 6
> Bot: 7
~/Documents/Content/Design_Fundamentals/Examples/polling_and_streaming — node client.js
Clements-MBP:polling_and_streaming clementmihailescu$ MODE=poll NAME=Antoine node client.js
> Chat Room: Welcome!
Hi Clement
How are you?
a
b
c
> Clement: Hey Antoine
> Clement: a
> Clement: b
> Clement: c
> Bot: 1
> Bot: 2
> Bot: 3
> Bot: 4
> Bot: 5
> Bot: 6
> Bot: 7
24:42
algoexpert.io