Lab4 Report

Yiwen Xu (48377645)

Document API

app.js

This is a module that contains the implementation to communicate as a server using the WebSocket protocol and to coordinate the work being processed by the client.

Attributes

Name	Type
app	Express
server	http.Server
wss	WebSocketServer
ws	WebSocket.WebSocket
clients	Array[WebSocket.WebSocket]
colors	Array[String]
userID	Number
receivedData	Number
startTime	Array[Number]
endTime	Number
execTimes	Array[Number]

Methods

Name	Description
wss.on('connection', function(ws))	Used to handle the logic after the client connects to the server.
ws.on('message', function(msg))	Logical processing after receiving the message.
ws.on('close', function())	Used to handle the logic after the client close the connection with the server.
app.configure()	Configure Express.
app.get('/', (req, res) ⇒{})	Set up such a single route with Express.

browser.js

This is a module that contains the implementation to communicate as a client using the WebSocket protocol and processes the work.

Attributes

Name	Type
myColor	String
myName	String
connection	WebSocket
data	Number

Methods

Name	Description
WebSocket.onopen	The socket's open event listener, used to handle the logic after the socket is successfully opened.
WebSocket.onmessage	The socket's message event listener, used to handle logic after receiving a message
WebSocket.onclose	The socket's close event listener, used to handle the logic after the socket is closed.
input.keydown	Event listener for the Enter key. Sends a message to the server when the Enter key is pressed.
addMessage	Compose the received content into div blocks for display on the HTML page.
work	Client's work process.

Main.cpp

This is a module that contains the implementation to communicate as a client using the WebSocket protocol and processes the work, which is a comparison version of `browser.js` .

Attributes

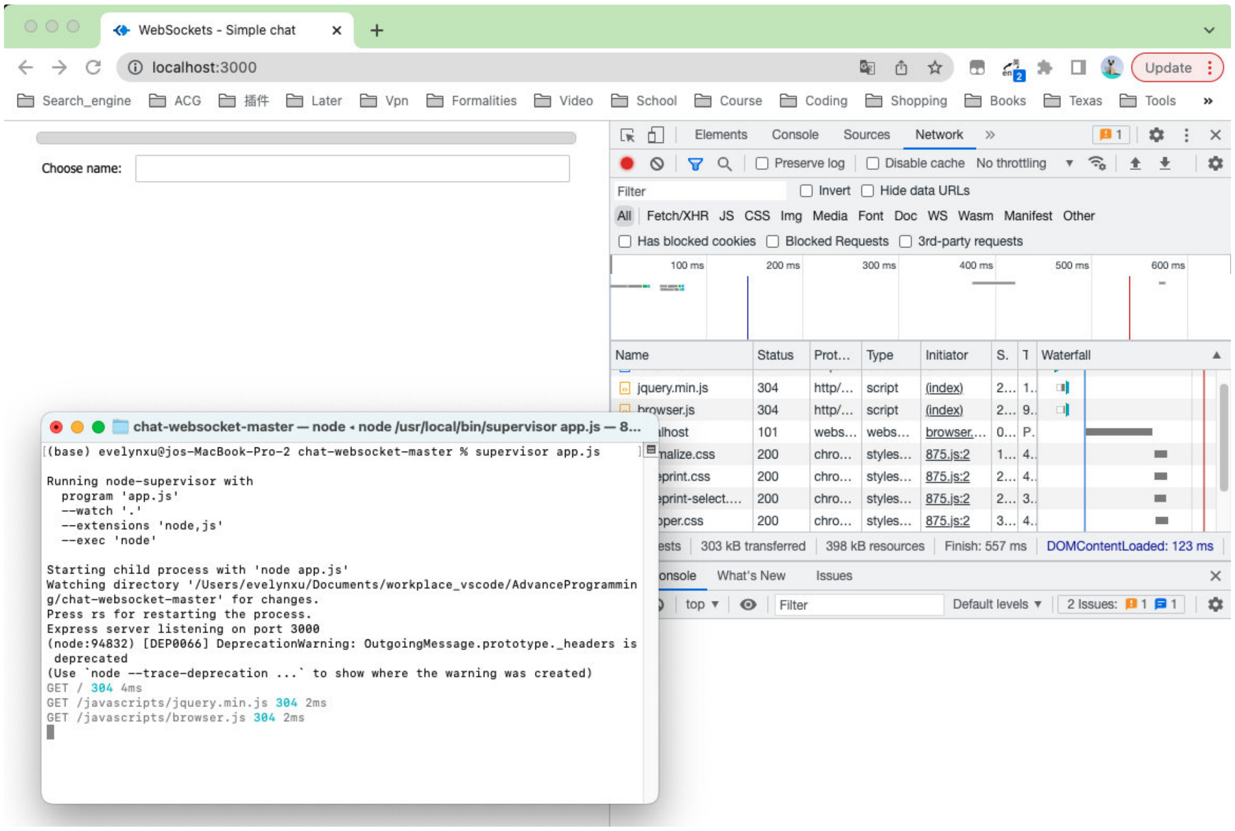
Name	Type
myName	string
socket	EMSCRIPTEN_WEBSOCKET_T
data	double

Methods

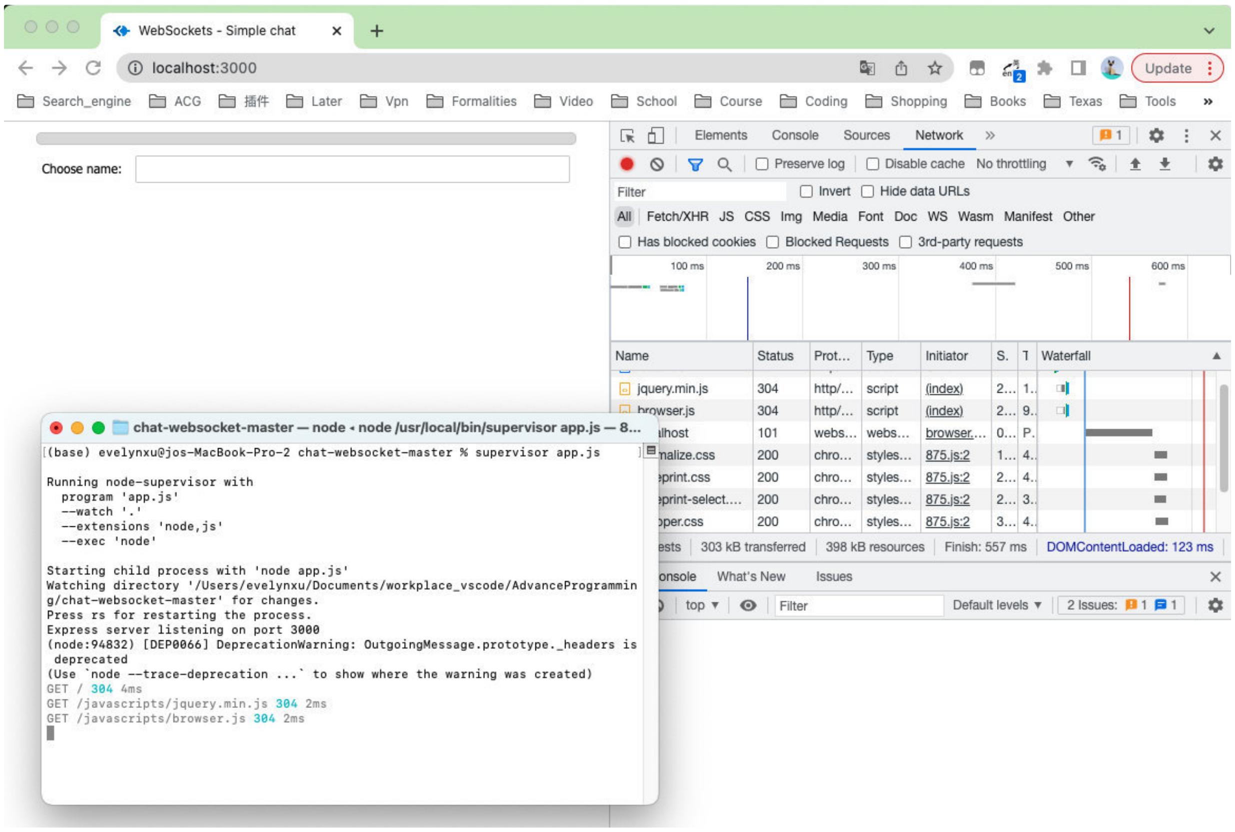
Name	Input	Output	Description
WebSocketOpen	eventTypr:int, e:EmscriptenWebSocketOpenEvent*, userData:void*	EM_BOOL	
WebSocketClose	eventTypr:int, e:EmscriptenWebSocketCloseEvent*, userData:void*	EM_BOOL	
WebSocketMessage	eventTypr:int, e:EmscriptenWebSocketMessageEvent*, userData:void*	EM_BOOL	
work	Client's work process.	void	

Networking Architecture

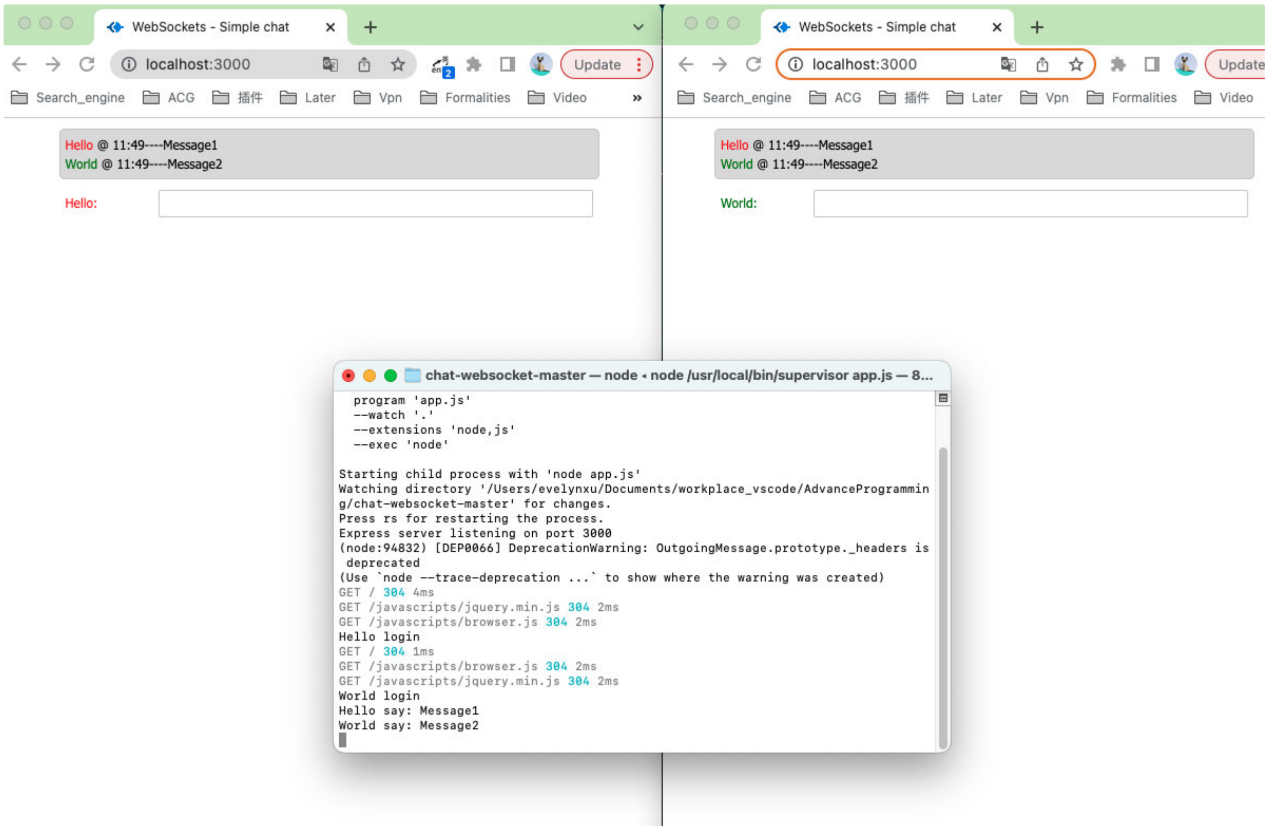
The code base of this project is coming from a simple chat demo. As the screenshots shows below.



```
(base) evelynxu@jos-MacBook-Pro-2 chat-websocket-master % supervisor app.js

Running node-supervisor with
  program 'app.js'
  --watch '.'
  --extensions 'node,js'
  --exec 'node'

Starting child process with 'node app.js'
Watching directory '/Users/evelynxu/Documents/workplace_vscode/AdvanceProgrammin
g/chat-websocket-master' for changes.
Press rs for restarting the process.
Express server listening on port 3000
(node:94832) [DEP0066] DeprecationWarning: OutgoingMessage.prototype._headers is
 deprecated
(Use `node --trace-deprecation ...` to show where the warning was created)
GET / 304 4ms
GET /javascripts/jquery.min.js 304 2ms
GET /javascripts/browser.js 304 2ms
```



```
(base) evelynxu@jos-MacBook-Pro-2 chat-websocket-master % supervisor app.js

Running node-supervisor with
  program 'app.js'
  --watch '.'
  --extensions 'node,js'
  --exec 'node'

Starting child process with 'node app.js'
Watching directory '/Users/evelynxu/Documents/workplace_vscode/AdvanceProgrammin
g/chat-websocket-master' for changes.
Press rs for restarting the process.
Express server listening on port 3000
(node:94832) [DEP0066] DeprecationWarning: OutgoingMessage.prototype._headers is
 deprecated
(Use `node --trace-deprecation ...` to show where the warning was created)
GET / 304 4ms
GET /javascripts/jquery.min.js 304 2ms
GET /javascripts/browser.js 304 2ms
```

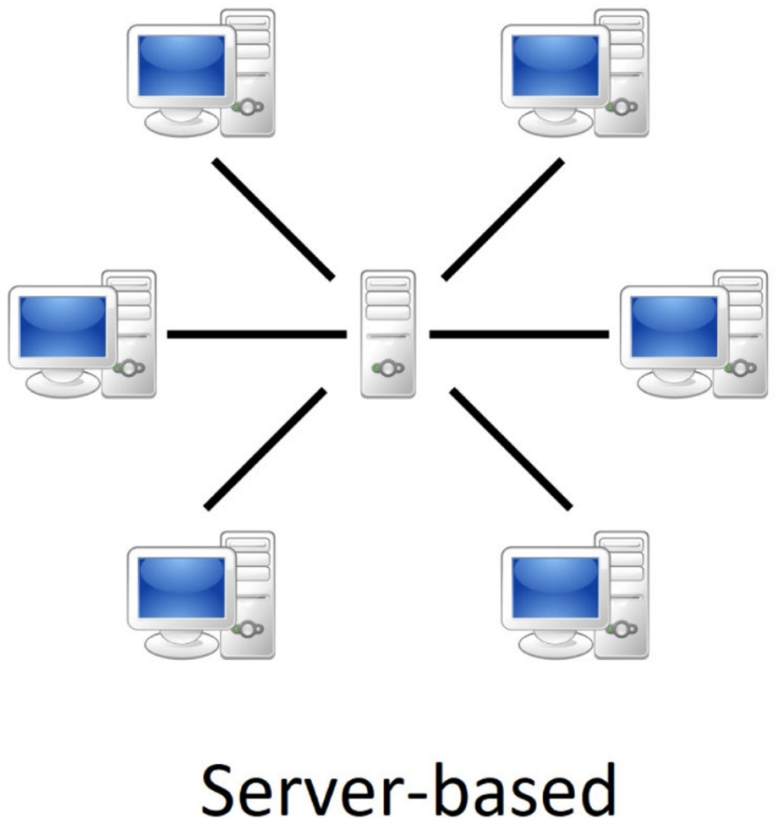


```
Hello @ 11:49----Message1
World @ 11:49----Message2
Hello:
```

```
Hello @ 11:49----Message1
World @ 11:49----Message2
World:
```

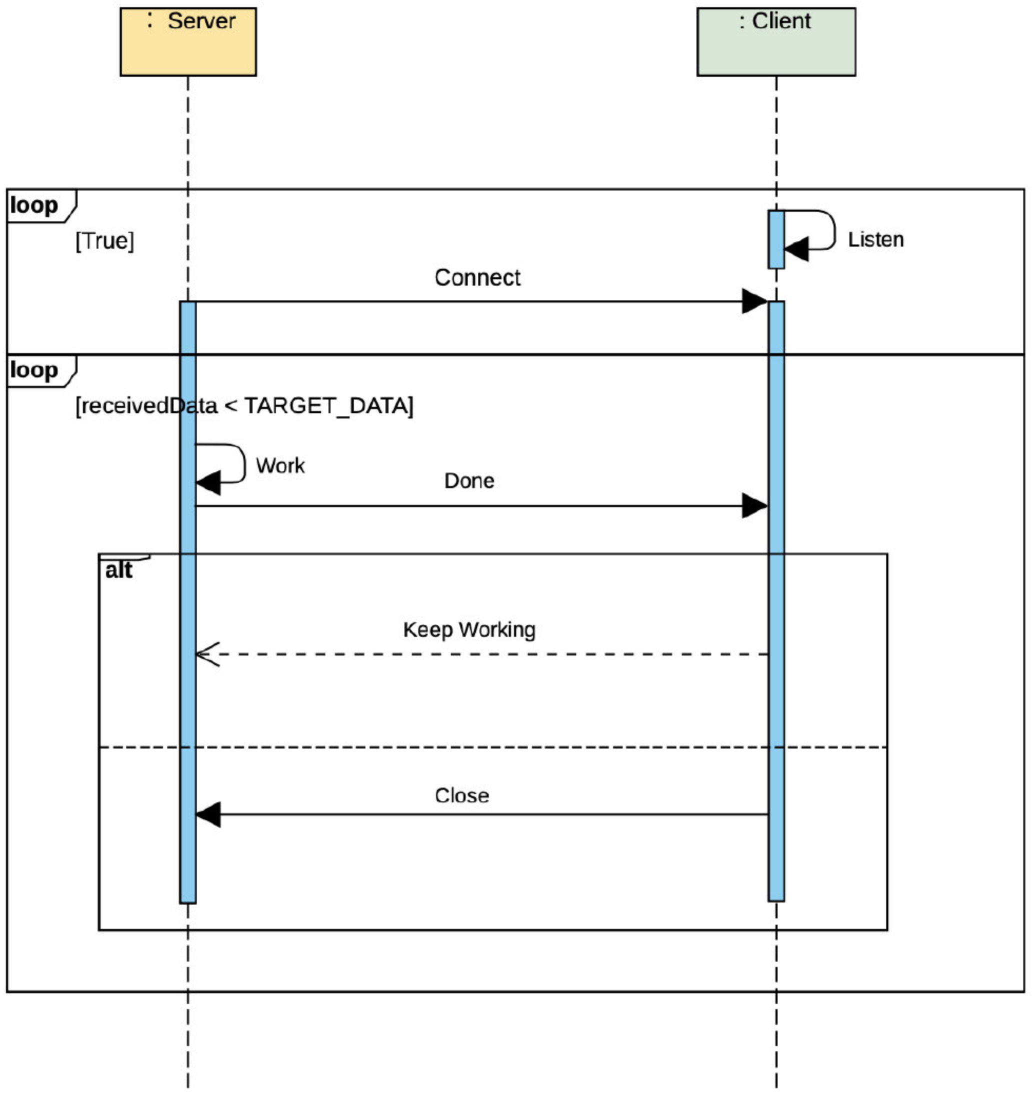
```
chat-websocket-master — node ‹ node /usr/local/bin/supervisor app.js — 8...
  program 'app.js'
  --watch '.'
  --extensions 'node,js'
  --exec 'node'

Starting child process with 'node app.js'
Watching directory '/Users/evelynxu/Documents/workplace_vscode/AdvanceProgrammin
g/chat-websocket-master' for changes.
Press rs for restarting the process.
Express server listening on port 3000
(node:94832) [DEP0066] DeprecationWarning: OutgoingMessage.prototype._headers is
 deprecated
(Use `node --trace-deprecation ...` to show where the warning was created)
GET / 304 4ms
GET /javascripts/jquery.min.js 304 2ms
GET /javascripts/browser.js 304 2ms
Hello login
GET / 304 1ms
GET /javascripts/browser.js 304 2ms
GET /javascripts/jquery.min.js 304 2ms
World login
Hello say: Message1
World say: Message2
```

To meet the requirements, I modified it to implement a part of producer-consumer model which just like Lab3. In this case, each client is represents a machine and sever is represents a factory which can coordinate the work that being processed by machines. When the client connects to the server, it will start working. After each round of work, it will send messages and data to the server. The server will check if the target number of data has been reached, or let the client continue working.



The networking architecture

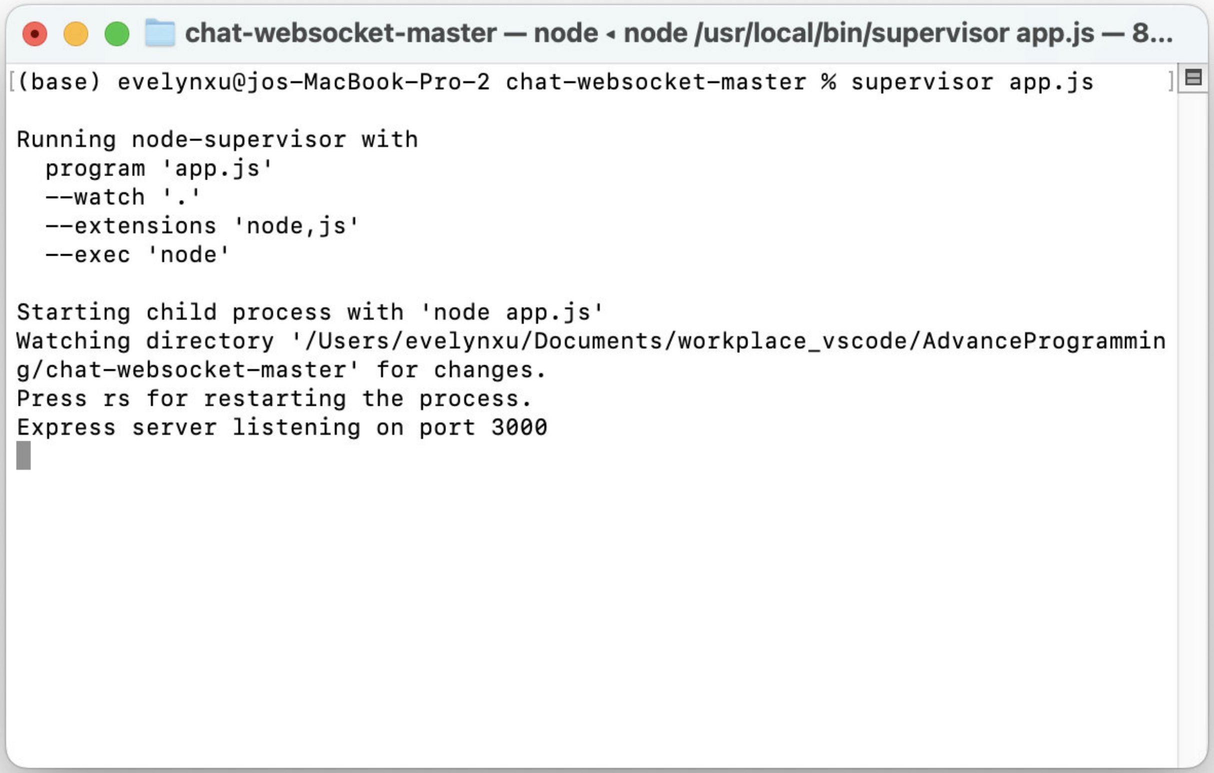
As the sequence diagram shows below, we can see the flow logic of this project and how a server actually coordinates the work process.



Demonstration

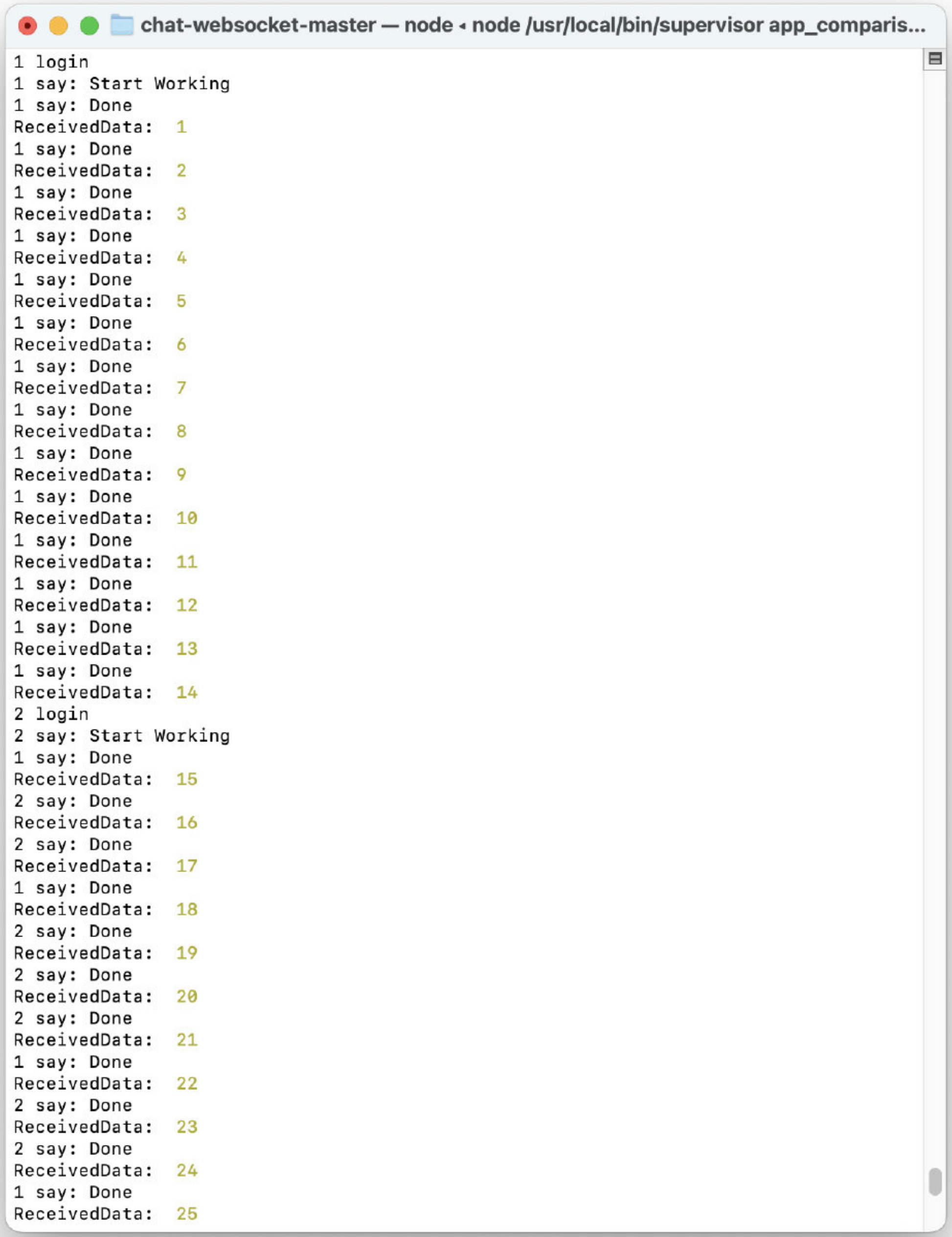
Server

The server is implemented with JavaScript using node server. It also using Express.js, which is a web application framework for Node.js, enables you to build server applications in Node.js. And the logs of message and progress will be printed during runtime.



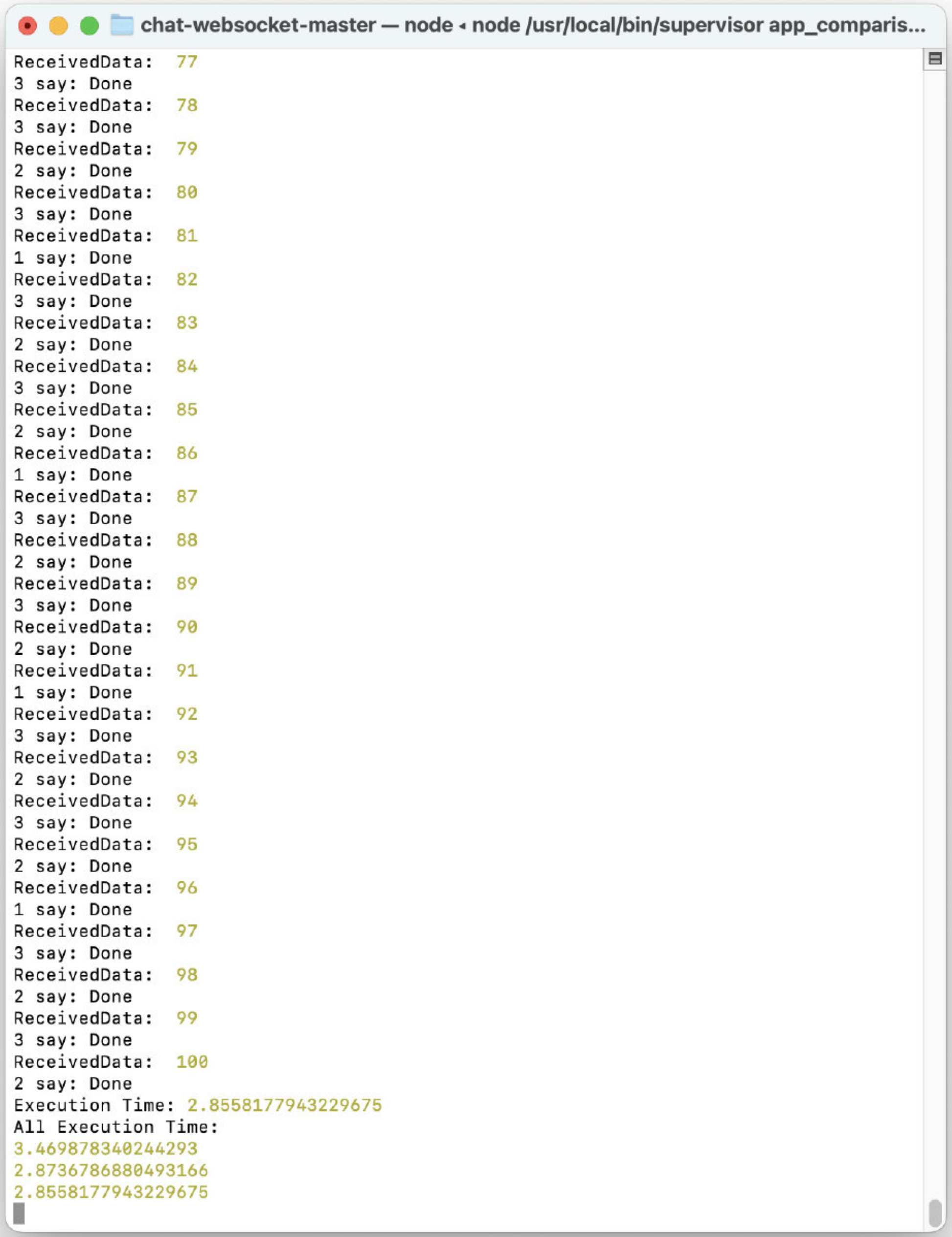
chat-websocket-master — node ◂ node /usr/local/bin/supervisor app.js — 8...

```
(base) evelynxu@jos-MacBook-Pro-2 chat-websocket-master % supervisor app.js

Running node-supervisor with
  program 'app.js'
  --watch '.'
  --extensions 'node,js'
  --exec 'node'

Starting child process with 'node app.js'
Watching directory '/Users/evelynxu/Documents/workplace_vscode/AdvanceProgrammin
g/chat-websocket-master' for changes.
Press rs for restarting the process.
Express server listening on port 3000
```



chat-websocket-master — node ◂ node /usr/local/bin/supervisor app_comparis...

```
1 login
1 say: Start Working
1 say: Done
ReceivedData:  1
1 say: Done
ReceivedData:  2
1 say: Done
ReceivedData:  3
1 say: Done
ReceivedData:  4
1 say: Done
ReceivedData:  5
1 say: Done
ReceivedData:  6
1 say: Done
ReceivedData:  7
1 say: Done
ReceivedData:  8
1 say: Done
ReceivedData:  9
1 say: Done
ReceivedData:  10
1 say: Done
ReceivedData:  11
1 say: Done
ReceivedData:  12
1 say: Done
ReceivedData:  13
1 say: Done
ReceivedData:  14
2 login
2 say: Start Working
1 say: Done
ReceivedData:  15
2 say: Done
ReceivedData:  16
2 say: Done
ReceivedData:  17
1 say: Done
ReceivedData:  18
2 say: Done
ReceivedData:  19
2 say: Done
ReceivedData:  20
2 say: Done
ReceivedData:  21
1 say: Done
ReceivedData:  22
2 say: Done
ReceivedData:  23
2 say: Done
ReceivedData:  24
1 say: Done
ReceivedData:  25
```

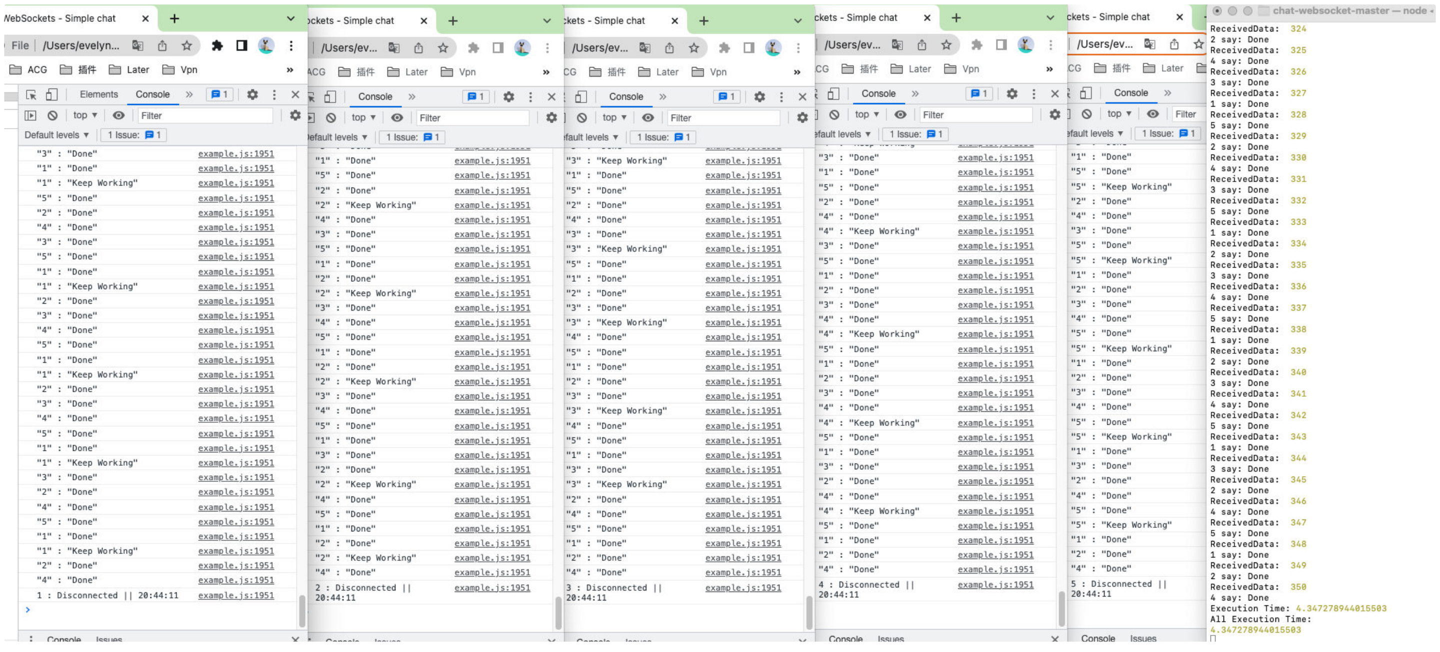

chat-websocket-master — node ◂ node /usr/local/bin/supervisor app_comparis...

```
ReceivedData:  77
3 say: Done
ReceivedData:  78
3 say: Done
ReceivedData:  79
2 say: Done
ReceivedData:  80
3 say: Done
ReceivedData:  81
1 say: Done
ReceivedData:  82
3 say: Done
ReceivedData:  83
2 say: Done
ReceivedData:  84
3 say: Done
ReceivedData:  85
2 say: Done
ReceivedData:  86
1 say: Done
ReceivedData:  87
3 say: Done
ReceivedData:  88
2 say: Done
ReceivedData:  89
3 say: Done
ReceivedData:  90
2 say: Done
ReceivedData:  91
1 say: Done
ReceivedData:  92
3 say: Done
ReceivedData:  93
2 say: Done
ReceivedData:  94
3 say: Done
ReceivedData:  95
2 say: Done
ReceivedData:  96
1 say: Done
ReceivedData:  97
3 say: Done
ReceivedData:  98
2 say: Done
ReceivedData:  99
3 say: Done
ReceivedData:  100
2 say: Done
Execution Time: 2.8558177943229675
All Execution Time:
3.469878340244293
2.8736786880493166
2.8558177943229675
```

Client

C++

The native comparison application is written in C++ code using Emscripten C++ WebSockets API. It can be run in the browser, but the output will be displayed in the console.



```
"3" : "Done"                     example.js:1951
"1" : "Done"                     example.js:1951
"1" : "Keep Working"             example.js:1951
"5" : "Done"                     example.js:1951
"2" : "Done"                     example.js:1951
"4" : "Done"                     example.js:1951
"3" : "Done"                     example.js:1951
"5" : "Done"                     example.js:1951
"1" : "Done"                     example.js:1951
"1" : "Keep Working"             example.js:1951
"2" : "Done"                     example.js:1951
"3" : "Done"                     example.js:1951
"4" : "Done"                     example.js:1951
"5" : "Done"                     example.js:1951
"1" : "Done"                     example.js:1951
"1" : "Keep Working"             example.js:1951
"2" : "Done"                     example.js:1951
"3" : "Done"                     example.js:1951
"4" : "Done"                     example.js:1951
"5" : "Done"                     example.js:1951
"1" : "Done"                     example.js:1951
"1" : "Keep Working"             example.js:1951
"3" : "Done"                     example.js:1951
"2" : "Done"                     example.js:1951
"4" : "Done"                     example.js:1951
"5" : "Done"                     example.js:1951
"1" : "Done"                     example.js:1951
"1" : "Keep Working"             example.js:1951
"2" : "Done"                     example.js:1951
"4" : "Done"                     example.js:1951
1 : Disconnected || 20:44:11     example.js:1951
```

```
"1" : "Done"                     example.js:1951
"5" : "Done"                     example.js:1951
"2" : "Done"                     example.js:1951
"2" : "Keep Working"             example.js:1951
"4" : "Done"                     example.js:1951
"3" : "Done"                     example.js:1951
"5" : "Done"                     example.js:1951
"1" : "Done"                     example.js:1951
"2" : "Done"                     example.js:1951
"2" : "Keep Working"             example.js:1951
"3" : "Done"                     example.js:1951
"4" : "Done"                     example.js:1951
"5" : "Done"                     example.js:1951
"1" : "Done"                     example.js:1951
"2" : "Done"                     example.js:1951
"2" : "Keep Working"             example.js:1951
"3" : "Done"                     example.js:1951
"4" : "Done"                     example.js:1951
"5" : "Done"                     example.js:1951
"1" : "Done"                     example.js:1951
"3" : "Done"                     example.js:1951
"2" : "Done"                     example.js:1951
"2" : "Keep Working"             example.js:1951
"4" : "Done"                     example.js:1951
"5" : "Done"                     example.js:1951
"1" : "Done"                     example.js:1951
"2" : "Done"                     example.js:1951
"2" : "Keep Working"             example.js:1951
"4" : "Done"                     example.js:1951
2 : Disconnected ||              example.js:1951
20:44:11
```

```
"3" : "Keep Working"             example.js:1951
"1" : "Done"                     example.js:1951
"5" : "Done"                     example.js:1951
"2" : "Done"                     example.js:1951
"4" : "Done"                     example.js:1951
"3" : "Done"                     example.js:1951
"3" : "Keep Working"             example.js:1951
"5" : "Done"                     example.js:1951
"1" : "Done"                     example.js:1951
"2" : "Done"                     example.js:1951
"3" : "Done"                     example.js:1951
"3" : "Keep Working"             example.js:1951
"4" : "Done"                     example.js:1951
"5" : "Done"                     example.js:1951
"1" : "Done"                     example.js:1951
"2" : "Done"                     example.js:1951
"3" : "Done"                     example.js:1951
"3" : "Keep Working"             example.js:1951
"4" : "Done"                     example.js:1951
"5" : "Done"                     example.js:1951
"1" : "Done"                     example.js:1951
"3" : "Done"                     example.js:1951
"3" : "Keep Working"             example.js:1951
"2" : "Done"                     example.js:1951
"4" : "Done"                     example.js:1951
"5" : "Done"                     example.js:1951
"1" : "Done"                     example.js:1951
"2" : "Done"                     example.js:1951
"4" : "Done"                     example.js:1951
3 : Disconnected ||              example.js:1951
20:44:11
```

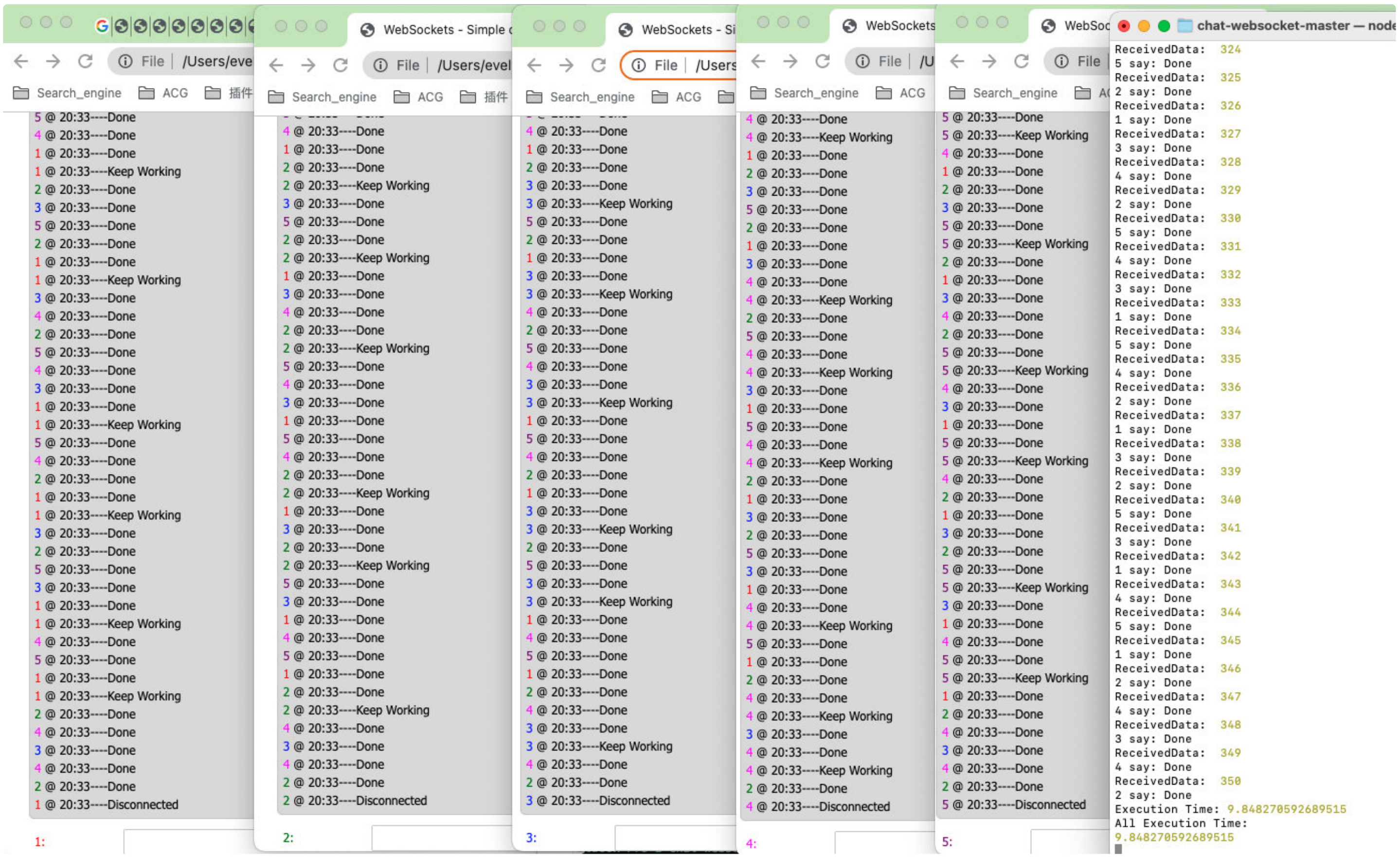
```
"3" : "Done"                     example.js:1951
"1" : "Done"                     example.js:1951
"5" : "Done"                     example.js:1951
"2" : "Done"                     example.js:1951
"4" : "Done"                     example.js:1951
"4" : "Keep Working"             example.js:1951
"3" : "Done"                     example.js:1951
"5" : "Done"                     example.js:1951
"1" : "Done"                     example.js:1951
"2" : "Done"                     example.js:1951
"3" : "Done"                     example.js:1951
"4" : "Done"                     example.js:1951
"4" : "Keep Working"             example.js:1951
"5" : "Done"                     example.js:1951
"1" : "Done"                     example.js:1951
"2" : "Done"                     example.js:1951
"3" : "Done"                     example.js:1951
"4" : "Done"                     example.js:1951
"4" : "Keep Working"             example.js:1951
"5" : "Done"                     example.js:1951
"1" : "Done"                     example.js:1951
"3" : "Done"                     example.js:1951
"2" : "Done"                     example.js:1951
"4" : "Done"                     example.js:1951
"4" : "Keep Working"             example.js:1951
"5" : "Done"                     example.js:1951
"1" : "Done"                     example.js:1951
"2" : "Done"                     example.js:1951
"4" : "Done"                     example.js:1951
4 : Disconnected ||              example.js:1951
20:44:11
```

```
"1" : "Done"
"5" : "Done"
"5" : "Keep Working"
"2" : "Done"
"4" : "Done"
"3" : "Done"
"5" : "Done"
"5" : "Keep Working"
"1" : "Done"
"2" : "Done"
"3" : "Done"
"4" : "Done"
"5" : "Done"
"5" : "Keep Working"
"1" : "Done"
"2" : "Done"
"3" : "Done"
"4" : "Done"
"5" : "Done"
"5" : "Keep Working"
"1" : "Done"
"3" : "Done"
"2" : "Done"
"4" : "Done"
"5" : "Done"
"5" : "Keep Working"
"1" : "Done"
"2" : "Done"
"4" : "Done"
5 : Disconnected ||
20:44:11
```

```
ReceivedData:  324
2 say: Done
ReceivedData:  325
4 say: Done
ReceivedData:  326
3 say: Done
ReceivedData:  327
1 say: Done
ReceivedData:  328
5 say: Done
ReceivedData:  329
2 say: Done
ReceivedData:  330
4 say: Done
ReceivedData:  331
3 say: Done
ReceivedData:  332
5 say: Done
ReceivedData:  333
1 say: Done
ReceivedData:  334
2 say: Done
ReceivedData:  335
3 say: Done
ReceivedData:  336
4 say: Done
ReceivedData:  337
5 say: Done
ReceivedData:  338
1 say: Done
ReceivedData:  339
2 say: Done
ReceivedData:  340
3 say: Done
ReceivedData:  341
4 say: Done
ReceivedData:  342
5 say: Done
ReceivedData:  343
1 say: Done
ReceivedData:  344
3 say: Done
ReceivedData:  345
2 say: Done
ReceivedData:  346
4 say: Done
ReceivedData:  347
5 say: Done
ReceivedData:  348
1 say: Done
ReceivedData:  349
2 say: Done
ReceivedData:  350
4 say: Done
Execution Time: 4.347278944015503
All Execution Time:
4.347278944015503
```

JavaScript

The comparison version for test application is written in JavaScript. In can be run in the browser, and display the performance data on the page.



```
5 @ 20:33----Done
4 @ 20:33----Done
1 @ 20:33----Done
1 @ 20:33----Keep Working
2 @ 20:33----Done
3 @ 20:33----Done
5 @ 20:33----Done
2 @ 20:33----Done
1 @ 20:33----Done
1 @ 20:33----Keep Working
3 @ 20:33----Done
4 @ 20:33----Done
2 @ 20:33----Done
5 @ 20:33----Done
4 @ 20:33----Done
3 @ 20:33----Done
1 @ 20:33----Done
1 @ 20:33----Keep Working
5 @ 20:33----Done
4 @ 20:33----Done
2 @ 20:33----Done
1 @ 20:33----Done
1 @ 20:33----Keep Working
3 @ 20:33----Done
2 @ 20:33----Done
5 @ 20:33----Done
3 @ 20:33----Done
1 @ 20:33----Done
1 @ 20:33----Keep Working
4 @ 20:33----Done
5 @ 20:33----Done
1 @ 20:33----Done
1 @ 20:33----Keep Working
2 @ 20:33----Done
4 @ 20:33----Done
3 @ 20:33----Done
4 @ 20:33----Done
2 @ 20:33----Done
1 @ 20:33----Disconnected

1:
```

```
4 @ 20:33----Done
1 @ 20:33----Done
2 @ 20:33----Done
2 @ 20:33----Keep Working
3 @ 20:33----Done
5 @ 20:33----Done
2 @ 20:33----Done
2 @ 20:33----Keep Working
1 @ 20:33----Done
3 @ 20:33----Done
4 @ 20:33----Done
2 @ 20:33----Done
2 @ 20:33----Keep Working
5 @ 20:33----Done
4 @ 20:33----Done
3 @ 20:33----Done
1 @ 20:33----Done
5 @ 20:33----Done
4 @ 20:33----Done
2 @ 20:33----Done
2 @ 20:33----Keep Working
1 @ 20:33----Done
3 @ 20:33----Done
2 @ 20:33----Done
2 @ 20:33----Keep Working
5 @ 20:33----Done
3 @ 20:33----Done
1 @ 20:33----Done
4 @ 20:33----Done
5 @ 20:33----Done
1 @ 20:33----Done
2 @ 20:33----Done
2 @ 20:33----Keep Working
4 @ 20:33----Done
3 @ 20:33----Done
4 @ 20:33----Done
2 @ 20:33----Done
2 @ 20:33----Disconnected

2:
```

```
4 @ 20:33----Done
1 @ 20:33----Done
2 @ 20:33----Done
3 @ 20:33----Done
3 @ 20:33----Keep Working
5 @ 20:33----Done
2 @ 20:33----Done
1 @ 20:33----Done
3 @ 20:33----Done
3 @ 20:33----Keep Working
4 @ 20:33----Done
2 @ 20:33----Done
5 @ 20:33----Done
4 @ 20:33----Done
3 @ 20:33----Done
3 @ 20:33----Keep Working
1 @ 20:33----Done
5 @ 20:33----Done
4 @ 20:33----Done
2 @ 20:33----Done
1 @ 20:33----Done
3 @ 20:33----Done
3 @ 20:33----Keep Working
2 @ 20:33----Done
5 @ 20:33----Done
3 @ 20:33----Done
3 @ 20:33----Keep Working
1 @ 20:33----Done
4 @ 20:33----Done
5 @ 20:33----Done
1 @ 20:33----Done
2 @ 20:33----Done
4 @ 20:33----Done
3 @ 20:33----Done
3 @ 20:33----Keep Working
4 @ 20:33----Done
2 @ 20:33----Done
3 @ 20:33----Disconnected

3:
```

```
4 @ 20:33----Done
4 @ 20:33----Keep Working
1 @ 20:33----Done
2 @ 20:33----Done
3 @ 20:33----Done
5 @ 20:33----Done
2 @ 20:33----Done
1 @ 20:33----Done
3 @ 20:33----Done
4 @ 20:33----Done
4 @ 20:33----Keep Working
2 @ 20:33----Done
5 @ 20:33----Done
4 @ 20:33----Done
4 @ 20:33----Keep Working
3 @ 20:33----Done
1 @ 20:33----Done
5 @ 20:33----Done
4 @ 20:33----Done
4 @ 20:33----Keep Working
2 @ 20:33----Done
1 @ 20:33----Done
3 @ 20:33----Done
2 @ 20:33----Done
5 @ 20:33----Done
3 @ 20:33----Done
1 @ 20:33----Done
4 @ 20:33----Done
4 @ 20:33----Keep Working
5 @ 20:33----Done
1 @ 20:33----Done
2 @ 20:33----Done
4 @ 20:33----Done
4 @ 20:33----Keep Working
3 @ 20:33----Done
4 @ 20:33----Done
4 @ 20:33----Keep Working
2 @ 20:33----Done
4 @ 20:33----Disconnected

4:
```

```
5 @ 20:33----Done
5 @ 20:33----Keep Working
4 @ 20:33----Done
1 @ 20:33----Done
2 @ 20:33----Done
3 @ 20:33----Done
5 @ 20:33----Done
5 @ 20:33----Keep Working
2 @ 20:33----Done
1 @ 20:33----Done
3 @ 20:33----Done
4 @ 20:33----Done
2 @ 20:33----Done
5 @ 20:33----Done
5 @ 20:33----Keep Working
4 @ 20:33----Done
3 @ 20:33----Done
1 @ 20:33----Done
5 @ 20:33----Done
5 @ 20:33----Keep Working
4 @ 20:33----Done
2 @ 20:33----Done
1 @ 20:33----Done
3 @ 20:33----Done
2 @ 20:33----Done
5 @ 20:33----Done
5 @ 20:33----Keep Working
3 @ 20:33----Done
1 @ 20:33----Done
4 @ 20:33----Done
5 @ 20:33----Done
5 @ 20:33----Keep Working
1 @ 20:33----Done
2 @ 20:33----Done
4 @ 20:33----Done
3 @ 20:33----Done
4 @ 20:33----Done
2 @ 20:33----Done
5 @ 20:33----Disconnected

5:
```

```
ReceivedData:  324
5 say: Done
ReceivedData:  325
2 say: Done
ReceivedData:  326
1 say: Done
ReceivedData:  327
3 say: Done
ReceivedData:  328
4 say: Done
ReceivedData:  329
2 say: Done
ReceivedData:  330
5 say: Done
ReceivedData:  331
4 say: Done
ReceivedData:  332
3 say: Done
ReceivedData:  333
1 say: Done
ReceivedData:  334
5 say: Done
ReceivedData:  335
4 say: Done
ReceivedData:  336
2 say: Done
ReceivedData:  337
1 say: Done
ReceivedData:  338
3 say: Done
ReceivedData:  339
2 say: Done
ReceivedData:  340
5 say: Done
ReceivedData:  341
3 say: Done
ReceivedData:  342
1 say: Done
ReceivedData:  343
4 say: Done
ReceivedData:  344
5 say: Done
ReceivedData:  345
1 say: Done
ReceivedData:  346
2 say: Done
ReceivedData:  347
4 say: Done
ReceivedData:  348
3 say: Done
ReceivedData:  349
4 say: Done
ReceivedData:  350
2 say: Done
Execution Time: 9.848270592689515
All Execution Time:
9.848270592689515
```

Timing Analysis Comparison

In this case, I set the code to loop 40 times and automatically outputting a running log to reflect the functionality achieved by the program and recording all of the execution time.

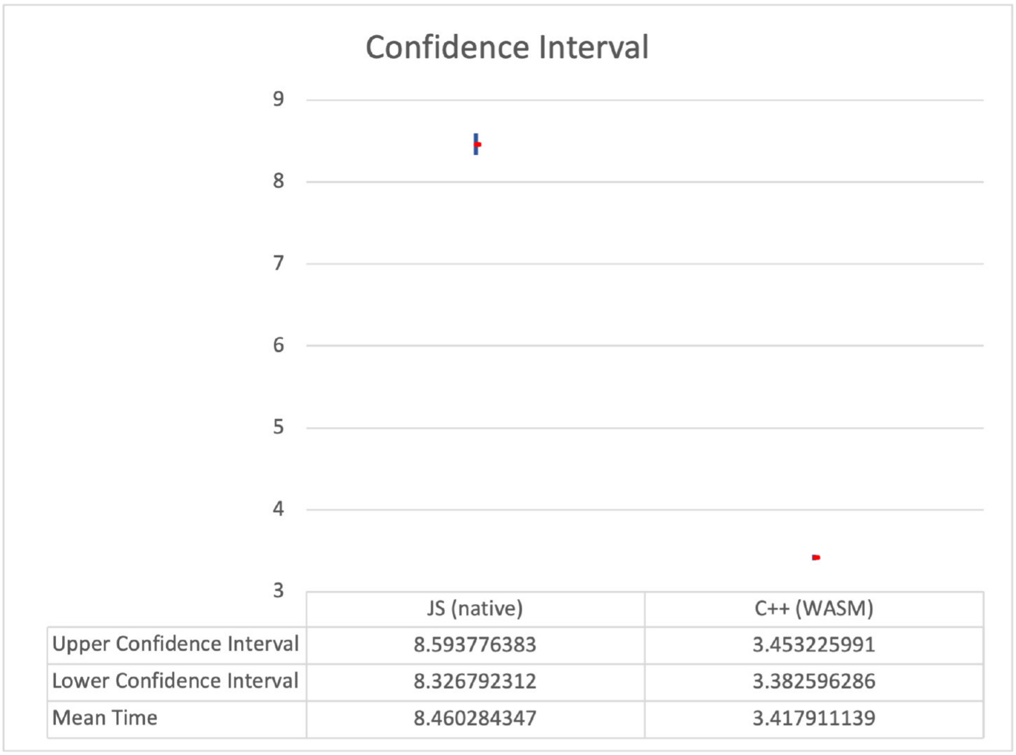
Native vs. JavaScript

To compare the performance of native code base and comparison version (JavaScript). I run both of them with four clients and same end termination condition.

I put all of the execution time data into `Data/ExecutionTime.xlsx` and calculate the 95% confidence interval. The obtained results are displayed below:

- For the native application results:
  - The 95% confidence interval is [8.326792312, 8.593776383], the average execution time is 8.460284347 s.
- For the comparison application results:
  - The 95% confidence interval is [3.382596286, 3.453225991], the average execution time is 3.417911139 s.

We can plot these confidence intervals on a figure to compare them.



	JS (native)	C++ (WASM)
Upper Confidence Interval	8.593776383	3.453225991
Lower Confidence Interval	8.326792312	3.382596286
Mean Time	8.460284347	3.417911139

As we can see in the plot, the results of native application are much higher than comparison version. And the lower confidence intervals of native application is higher than the upper confidence interval of comparison application. Therefore, we can learn that the results are statistically significant, which means the run speed of native application is slower than comparison application that transpiled from C++.

Different number of clients (JS)

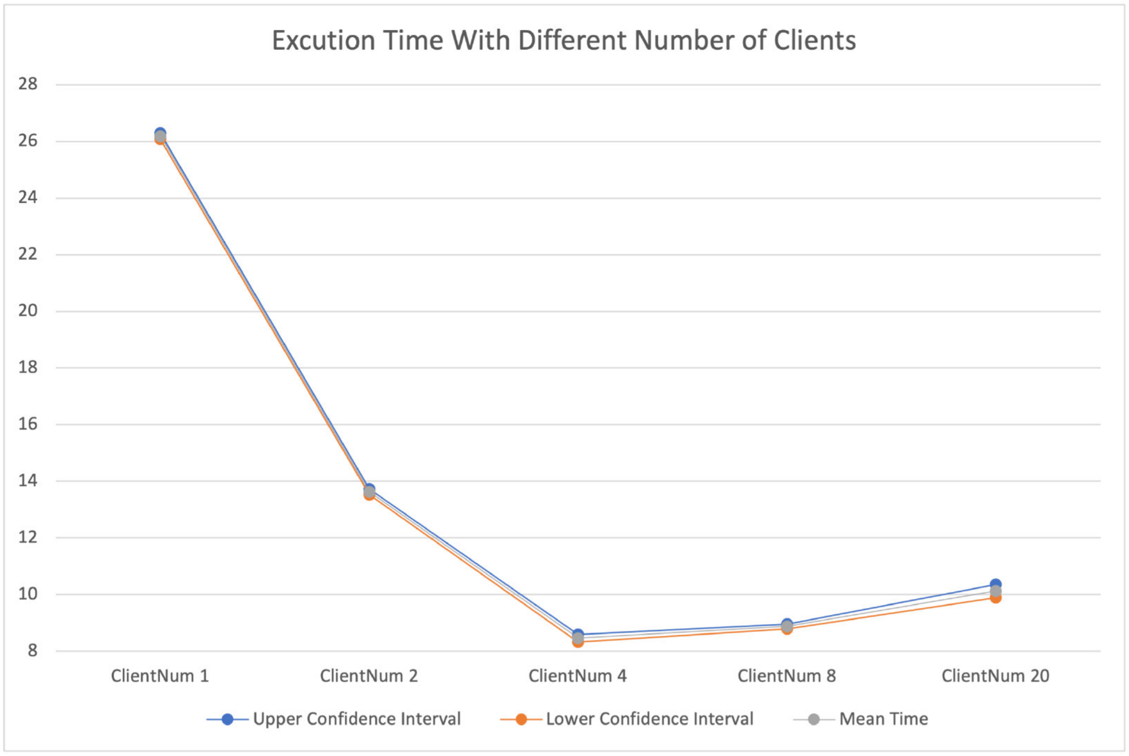
For the work logic of one server with multi-client is similar to how the main thread controlled worker threads. I am curious about the performance of this application with different number of clients.

I set the code to loop 40 times with open different number of clients (1 client, 2 clients, 4 clients, 8 clients, 20 clients).

I put all of the execution time data into `Data/ExecutionTime.xlsx` and calculate the 95% confidence interval. The obtained results are displayed below:

- The results of the native application with 1 client:
  - The 95% confidence interval is [26.30179803, 26.07560145], the average execution time is 26.18869974 s.
- The results of the native application with 2 client:
  - The 95% confidence interval is [13.52318919, 13.72795211], the average execution time is 13.62557065 s.
- The results of the native application with 4 client:
  - The 95% confidence interval is [8.32679231, 8.59377638], the average execution time is 8.46028435 s.
- The results of the native application with 8 client:
  - The 95% confidence interval is [8.79380625, 8.95787786], the average execution time is 8.87584205 s.
- The results of the native application with 20 client:
  - The 95% confidence interval is [9.89321460, 10.35209640], the average execution time is 10.12265550 s.

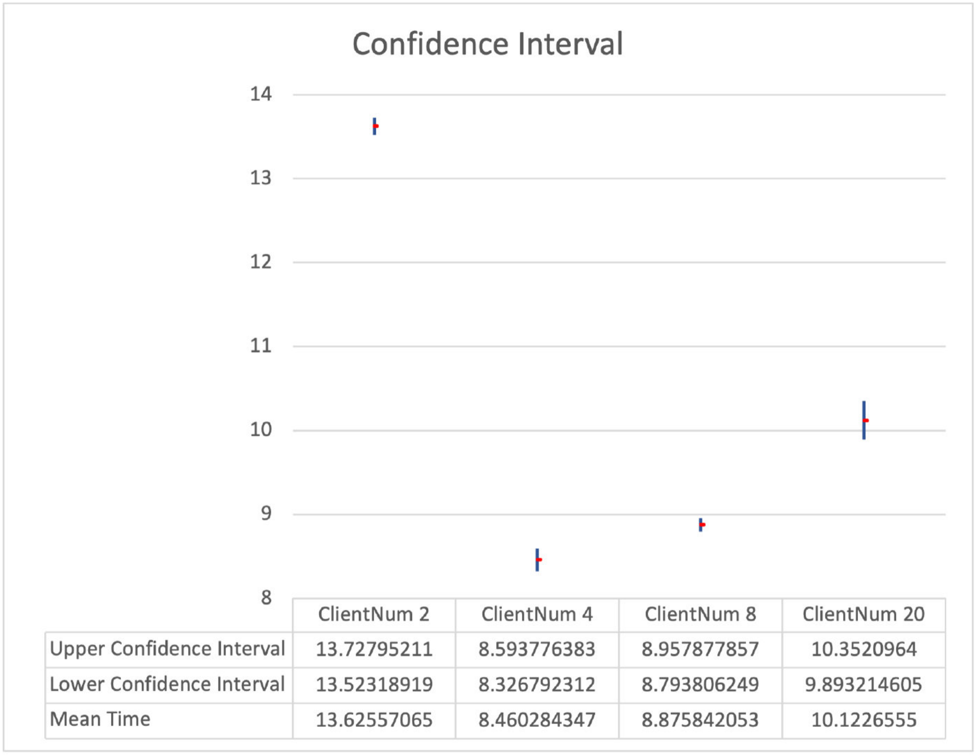
I plot these confidence intervals on a line plot to see the trend of execution time change with the number of clients.



As the plot shown before, we can see that the execution time decreases as the number of clients increases at first. However, after the number of clients is four, the execution time increases as the number of clients increases.

The reason of the line going down at first is multiple clients working at the same time, which reduced the amount of work time that needed to be spent on getting the same target, and server wait time. But the more clients, the less the bonus, because more clients means we need to spend more time on client connections and sending messages. Finally, the disadvantages outweigh the benefits, and the execution time increases as the number of clients increases.

To better analyze the results with number 2,4,8,20 clients, I plot these confidence intervals on a figure to compare them. As the results, we can considered all of them are statistically significant for the same reason was mentioned before.



	ClientNum 2	ClientNum 4	ClientNum 8	ClientNum 20
Upper Confidence Interval	13.72795211	8.593776383	8.957877857	10.3520964
Lower Confidence Interval	13.52318919	8.326792312	8.793806249	9.893214605
Mean Time	13.62557065	8.460284347	8.875842053	10.1226555