Le Xuan Duy

Follow

Bước đầu làm quen với NodeJS và Socket.io

Express Node.js socket.io Education chat

Aug 28th, 10:21 am

2332 8 10 Report

Chào mọi người!

Mở bài

Đôi điều chia sẻ trước khi đi vào nội dung chính của bài viết

Hôm nay là bài viết đầu tiên mình được viết trên Viblo.asia. Mình sẽ chia sẻ một số kiến thức mình tìm hiểu được về Nodejs và socket.io đều là kiến thức rất cơ bản thôi nhưng mình nghĩ nó có thể sẽ hữu ích cho những bạn mới bắt đầu học NodeJS như mình.

Quan điểm của mình là không cần thực sự giỏi về một technique nào đó thì bạn mới có thể viết bài chia sẻ kiến thức cho người khác. Mình nghĩ một người mới tìm hiểu cũng có thể chia sẽ những thứ mới mẻ mà mình đã tìm hiểu được qua đó có thể được mọi người đóng góp ý kiến vừa để học hỏi vừa để khắc sâu kiến thức nên nếu trong bài viết có gì sai xót rất mong được các bạn bỏ qua và góp ý cho mình nhé :x

Thông tin cần biết về NodeJs, express, Socket.io

NodeJS

```
NodeJS không phải là một JavaScript framework nhưng hầu hết các module của nó được
viết bằng JavaScript.

Nếu bạn đã từng làm việc với JavaScript bạn có thấy nó thật tuyệt giống mình không?
Giờ không những bạn có thể sử dụng nó cho các ứng dụng phía client mà nay bạn có
thể sử dụng nó để lập trình phía server điều này đúng là "awesome" mà.
```

Điểm nổi bật của NodeJS so với các ngôn ngữ lập trình khác

- Hoạt động với một luồng duy nhất và có khả năng asynchronous (bất đồng bộ). Không giống như server được viết bằng PHP thì mỗi ông request đến server thì server sẽ tạo ra một thread để xử lý trong khi đó server Node xử lý mọi hành động trong một thread duy nhất.

  => **Cách thiết kế như này sẽ có hỗ trợ như thế nào?**

  - Tối ưu hóa thời gian thực hiện tiến trình
  - Có khả năng mở rộng trong các ứng dụng web với nhiều hoạt động I/O liên tục.
  - Phù hợp để xây dựng các ứng dụng web stream hay các game chơi trên nền web đảm bảo việc độ trễ thời gian xử lý hành động là nhỏ nhất.
  - Dễ dàng để xây dựng các ứng dụng real-time.
  - Cách viết ứng dụng với Node đó là các ứng dụng được cấu tạo từ các module nhỏ sau đó được kết hợp lại với nhau điều này đảm bảo cho việc sửa đổi bảo trì một cách nhanh chóng.
  - Hiệu năng cao

  Nhưng nó cũng phải tồn tại cái gọi là nhược điểm chứ đúng không?

  Ở đây mình sẽ chỉ liệt kê ra một số điểm chưa tốt của NodeJs mà không đi vào tìm hiểu sâu, các bạn có thể tìm hiểu kỹ hơn thông qua việc tìm kiếm theo từ khóa đó

  - Bad concurrency
  - Single - Threaded:

    Đơn luồng cũng có thể là một điểm kém của NodeJs. Lợi thế của single-thread là việc không gặp phải vấn đề synchronize giữa các tiến trình, không cần chia sẻ trạng thái hiện tại... Bạn có thể dễ dàng viết một đoạn code mà sẽ ngốn của server khá nhiều thời gian để xử lý xong một vài trường hợp nào đó và có thể dẫn tới lock toàn bộ server.

  - Lack of inherent code organization

khác nhau giữa các thành viên hay có thể sử dụng nhiều design pattern khác nhau.

- Leak message

NodeJS core-module

Là những modules được biên dịch vào trong Node binary, đó là những modules khá quan trọng và được biên dịch kèm với Node khi cài đặt

- Cung cấp những functionalities cơ bản nhất của Node
- Gồm có: filesystem access, HTTP, HTTPS interfaces, và một số modules khác.
- Để sử dụng core module trong trương chình cần sử dụng phương thức require trong JavaScript file.
- Ví dụ muốn sử dụng module file system trong chương trình:

```
var fs = require('fs');
```

Khi module được require, Node sẽ tìm module đó trong thư mục core_modules. Sau đó ta có thể sử dụng được các phương thức mà module đó cung cấp.

Node.js third-party modules

Mình tạm hiểu nó là bên thứ ba cung cấp các sản phẩm, dịch vụ sẵn có cho mình ở đây đó là module. Các module này không được cài đặt sẵn sàng cùng với node như các core module mình đã giới thiệu phía trên.

Để sử dụng third-party modules trước tiên cần cài đặt những modules muốn sử dụng thông qua 'npm' (node manage package).

Ví dụ muốn sử dụng 'express' module cần thực hiện lệnh sau trên terminal:

```
npm install express
```

Sau đó Node sẽ tải express module vào thư mục 'node_modules' dưới đường dẫn thư mục đang làm việc

Để sử dụng thì ta sẽ load module vào chương trình sử dụng require như bình thường. Express module là một third-party rất quan trọng được sử dụng rất nhiều và nó cũng đem đến rất nhiều phương thức hữu ích giúp dễ dàng trong việc thực hiện các ứng dụng web

Socket.io

```
- Trong trình duyệt mà được hỗ trợ WebSockets protocol, một kết nối giữa server và browser được tạo ra quanh HTTP và được gọi là một “HTTP hanshake” (cái bắt tay). Một kết nối được tạo ra trình duyệt và server mở ra một cổng giao tiếp liên tục thông qua một TCP socket. Nó sẽ hỗ trợ cho cả việc gửi và truy vấn message trên một cổng kết nối. Điều này giúp server load ít hơn, giảm số message bị trễ, “and unify PUSH communication using standalone connection”.

- Tuy nhiên WebSocket mắc phải vấn đề đó là HTTP proxies, firewall và hosting provider. Khi Websocket sử dụng một phương thức giao tiếp ngoài HTTP, một phần nhiều trong số đó chưa được hỗ trợ và block bất cứ kết nối socket nào. Vấn đề này chỉ được giải quyết khi sử dụng thư viện trừu tượng mà có thể dễ dàng thay đổi giữa các giao thức dựa trên resoures có sẵn.

 - Socket.io được xây dựng để giải quyết vấn đề này và nó luôn sẵn sàng được sử dụng cho NodeJS developer.

**Socket.io**
- Là một module của NodeJs
- Được xây dựng nhằm mục đích tạo ra real time NodeJS application. Socket.io cung cấp cho lập trình viên các đặc trưng như event, room và tự động phục hồi lại kết nối.

- Khi chúng ta include Socket.io module vào trong ứng dụng của mình nó sẽ cung cấp cho chúng ta hai object đó là: socket server quản lý functionality phía server và socket client điều khiển funtionality phía client.
```

- Khi client muốn kết nối tới Socket.io server, nó sẽ gửi cho server một “handshake HTTP request”. Server sẽ phân tích request đó với những thông tin cần thiết trong suốt quá trình kết nối. Nó sẽ tìm cấu hình của middleware mà đã được đăng ký với server và thực thi chúng trước khi đưa ra sự kiện kết nối. Khi kết nối thành công thì connection event listener được thực thi, tạo ra một instance mới của socket có thể coi như định danh của client mà mỗi một client kết nối tới sẽ có 1 định danh. Các bạn có thể thấy rõ khi xem hình dưới đây

```
![socket.client.png](/uploads/3a115a7c-af19-48a7-9dc3-7b0e16deabd2.png)

- Một module khác của Node.js là LightStreamer-adapter cũng có tạo các kết nối từ client tới server nhưng không trực tiếp mà thông qua LightStreamer Server, đó là các máy chủ theo thời gian thực và nằm ngoài tiến trình của Node.js Server
```

Sau khi đã có một số khái niệm cơ bản như mình đã nói ở trên bây giờ ta có thể bắt đầu vào cài đặt và xây dựng một ứng dụng nhỏ sử dụng NodeJs và Socket.io

Cài đặt ứng dụng demo

=))

Hệ điều hành mình đang dùng là Ubuntu 15.10, bạn cũng thể sử dụng 14.04 hay 16.04 là bản mới nhất đều ok.

- Trước tiên để cài đặt các thành phần cần thiết bạn nên update lại hệ thống của mình để đảm bảo mọi thứ luôn được cập nhật và hỗ trợ tốt nhất:

  ```
  $ sudo apt-get update
  ```

- Cài đặt Nodejs
  Nếu như bạn đã có cài đặt phiên bản nodejs trước đó thì có thể cập nhật sang phiên bản mới nhất hiện nay bằng cài gỡ đi các cài đăt cũ như sau:

  ```
  $ sudo apt-get autoremove nodejs
  ```

  Tải bản cài đặt mới nhất của Nodejs là version 6

  ```
  $ curl -sL https://deb.nodesource.com/setup_6.x | sudo -E bash
  ```

  Sau đó sử dụng câu lệnh sau để tự động cài đặt nodejs và npm (node package manage - công cụ quản lý các package của bạn, cài đặt các gói)

  ```
  $ sudo apt-get install -y nodejs
  ```

  Tạo thư mục chứa project của bạn

  Đi tới đường dẫn bạn muốn đặt project ở đó rồi gõ lệnh:

  `$ mkdir awesomechat` (awesomechat ở đây là tên app của mình, bạn có thể lấy tên gì tùy ý)

  Bây giờ hãy cd vào trong project folder mình vừa tạo rồi chạy lệnh

  ```
  npm init
  ```

  Nó sẽ tạo ra một file package.json

  Chúng ta sẽ khai bao những gì cần required cho ứng dụng, mô tả về ứng dụng như phiên bản, tên, description.. Mọi thông tin trong package.json được viết dưới định dạng JSON. Bạn có thể tìm hiểu rõ hơn về package.json tại đây

  ```
  https://docs.npmjs.com/files/package.json
  ```

  Cài đặt express

  ```
  npm install --save express
  ```

  Tùy chọn --save sẽ lưu thông tin về express trong dependencies list được biểu diễn trong file package.json. Nên sử dụng cài đặt express với tùy chọn --save vì khi đó bạn biết được phiên

Bạn có thể không cần phải làm các bước trên từ bước tạo thư mục project với nodejs ta có thể sử dụng một công cụ hỗ trợ rất đắc lực cho những việc làm trên đó là sử dụng application generator tool đó là express-generator. Để có thể nhanh chóng tạo ra một bộ khung sẵn cho ứng dụng của chúng ta.

Để cài đặt express-generator làm như sau:

```
$ npm install express-generator -g
```

Sau đó để generate ra một project ta làm như sau:

```
$ express awesomechat
```

Khi đó express-generator sẽ tạo ra một thư mục có tên awesomechat và chứa sẵn các folder để chúng ta có thể tổ chức code trong đó, bao gồm cả file package.json được mô tả sẵn một số module thường dùng nhất đối với ứng dụng NodeJs. Sau đó công việc của chúng ta chỉ là chạy câu lệnh `$ npm install` là các module đó sẽ tự động được cài đặt, muốn cài đặt thêm module khác ngoài được định nghĩa sẵn ta có thể thêm mô tả trong dependencies rồi chạy lại lệnh đó hoặc có thể cài trực tiếp sử dụng `$ npm install packageName`

Cài đặt socket.io

```
$ npm install --save socket.io
```

Nếu qúa trình cài đặt gặp vấn đề gì với quyền truy cập mình sẽ sử dụng thêm với từ khóa `sudo`

Sau khi cài đặt xong ta có một thư mục awesomechat với cấu trúc như sau

```
├── bin
│   └── www
├── package.json
├── public
│   ├── images
│   ├── javascripts
│   └── stylesheets
│       └── style.css
├── routes
│   ├── index.js
│   └── users.js
└── views
    ├── error.jade
    ├── index.jade
    └── layout.jade

7 directories, 9 files
```

File package.json của chúng ta lúc này sẽ có dạng như sau:

```
{
  "name": "awesomechat",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
    "body-parser": "~1.15.1",
    "cookie-parser": "~1.4.3",
    "debug": "~2.2.0",
    "express": "~4.13.4",
    "jade": "~1.11.0",
    "morgan": "~1.7.0",
    "serve-favicon": "~2.3.0",
    "socket.io": "^1.4.8"
  }
}
```

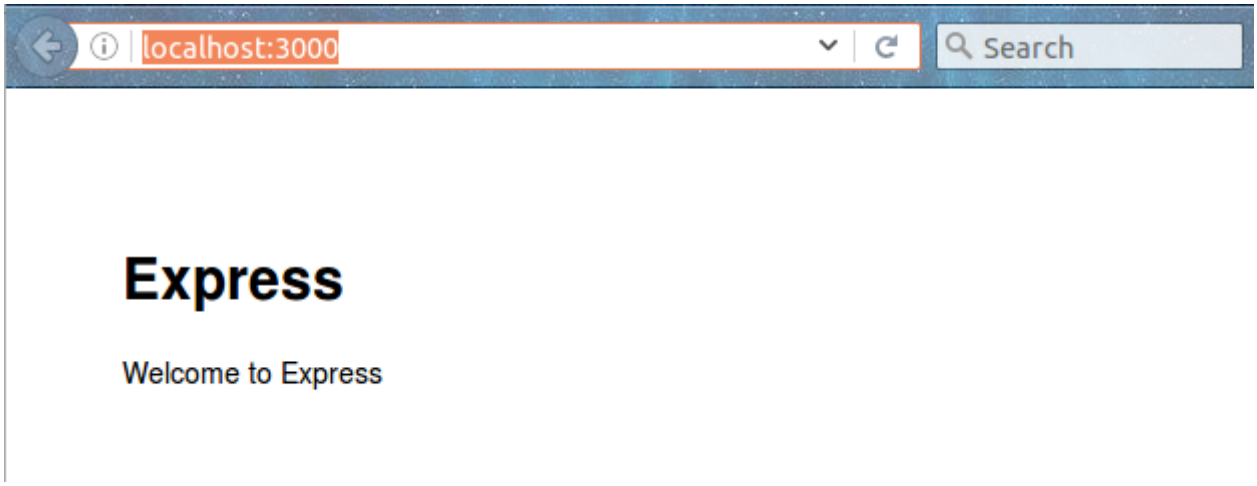
Theo các required dependencies chúng ta sẽ khai báo như trên, ở đây express-genertator đã làm việc đó, công việc của mình lúc này chỉ là chạy lệnh `npm install` để được cài đặt các module là được. Khi đó các module sẽ được tìm kiếm như đã được khai báo trong `denpendencies` và được tải về vào thư mục node module

Bây giờ ta đã cơ bản có một bộ khung cho ứng dụng của mình. Ta sẽ bắt đầu start server để

```
"scripts": { "start": "node ./bin/www" }
```

Khi chạy lệnh npm start thì nó sẽ tìm trong file package.json tới câu lệnh đã được gán cho "start" và thực thi nó. Trong file ./bin/www thì app.js đã được required trong đó, trong app.js là toàn bộ code phía server hiện tại mà ta có, trong www sẽ bổ xung thêm đó là tạo ra một server http và lắng nghe các request tới ứng dụng của mình

Giờ vào trình duyệt với đường dẫn localhost:3000 nếu nhận được kết quả như bên dưới là ta đã cài đặt thành công một ứng dụng nodejs cơ bản với ndoejs sử dụng express framework. Sau đó "let's do it" làm ứng dụng awesome chat của chúng ta thôi :x



Xây dựng awesomechat với Socket.io

Đoạn dưới đây là phần code phía server của mình, mình sẽ tạo một file awesomechat.js cùng thư mục với app.js, và toàn bộ code server mình để trong đó, bạn cũng có thể chỉnh sửa code trong app.js đã có sẵn và sử dụng `npm start` để khởi động ứng dụng. Còn đối với mình mình sẽ sử dụng

```
node [đường dẫn tới thư mục chứa awesomechat]/awesomechat.js
```

Hoặc thay vì required app.js trong file .bin/www thì bạn có thể required awesomechat.js và bỏ đi đoạn code cũ trong đó. Khi đó ta cũng có thể start server bằng cách

```
npm start
```

Phần code server của mình như sau

```
var express = require('express'),
app = express(),
server = require('http').createServer(app),
io = require('socket.io').listen(server),
users = {};
server.listen(8888);
app.use(express.static(__dirname + '/public'));
app.get('/', function(req, res){
res.sendfile(__dirname + '/views/chatui.html');
});

io.sockets.on('connection', function(socket){
socket.on('new user', function(name, data){
        if (name in users){
            data(false);
        }else{
            data(true);
        socket.nickname = name;
        users[socket.nickname] = socket;
        console.log('add nickName');
        updateNickNames();
    }

});

function updateNickNames(){
    io.sockets.emit('usernames', Object.keys(users));
}
socket.on('open-chatbox', function(data){
    users[data].emit('openbox', {nick: socket.nickname});
});
socket.on('send message',function(data, sendto){
    users[sendto].emit('new message',{msg: data, nick: socket.nickname, sendto:
sendto});
    users[socket.nickname].emit('new message',{msg: data, nick: socket.nickname,
sendto: sendto});

    console.log(data);
});
socket.on('disconnect', function(data){
    if (!socket.nickname) return;
    delete users[socket.nickname];
    updateNickNames();
});
});
```

- Nhìn vào đoạn code trên bạn có thể thấy:

- Tạo một server http và lắng nghe các request đến với app
- Tạo một thể hiện socket.io lắng nghe server htttp mình vừa tạo
- Mình sẽ cho server listen port 8888 để việc gửi và nhận dữ liệu thông qua cổng này, bạn có thể thay đổi tùy ý nhưng nên để số port cao lên khoảng trên 8000 để tránh xung đột với cổng của những ứng dụng khác trong máy của bạn.
- Bây giờ qua phần xử lý hành động của server socket.io
- Mọi event của client cần được xử lý bạn để trong

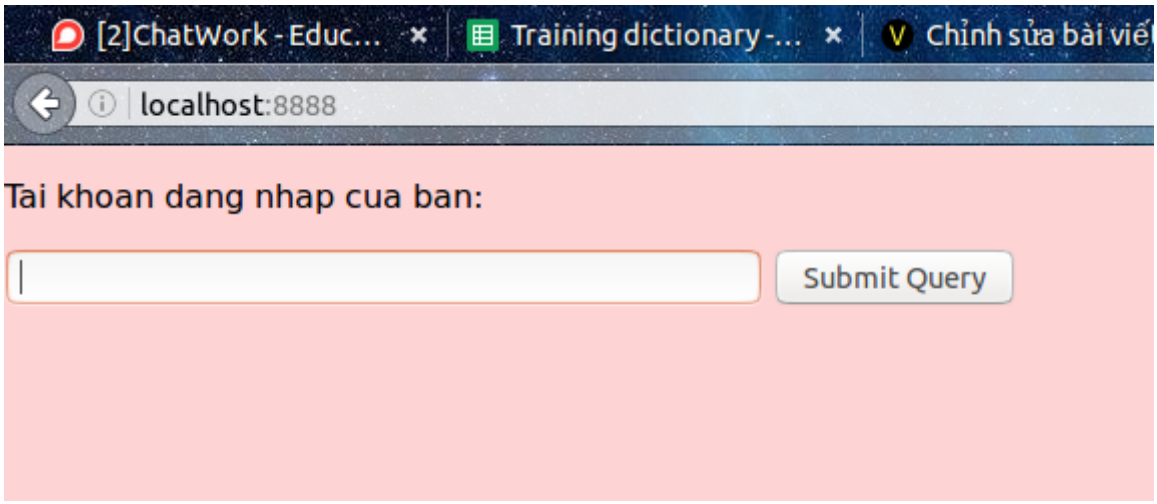
  ```
  io.sockets.on('connection', function(socket) {}
  ```

- Nghĩa là khi bạn kết nối tới server thì lúc này bạn đã có 1 định danh socket riêng của mình và server xử lý request và trả về response được khai báo xử lý trong callback của `io.sockets.on('connection', function(socket)`
- Một số thao tác chính bên server:
  - "new user" : Khi phía client phát lên sự kiện "new user" kèm theo dữ liệu mà nó gửi lên thì phía server mình sẽ quản lý và lưu thông tin tin của người dùng này vào object **users** đã được khai báo phía server.
  - "open-chatbox": Khi người dùng mở một hộp chat với user khác, thì server sẽ nhận được thông tin dữ liệu của người dùng gửi yêu cầu và người dùng được kết nối để mở một kênh giao tiếp giữa hai người này.
  - "send message": Khi một message được submit, phía client sẽ emit (phát) ra sự kiện "send message" và kèm theo nội dung của tin nhắn và người nhận tin nhắn. Server mình sẽ nhận lấy gửi về cho người gửi request và người được gửi message sự kiện "new message" gồm nội dung của message. Client chỉ việc "on" bắt lấy sự kiện này, nhận lấy thông tin message và hiển thị.
  - "disconnect": Khi client ngắt kết nối với server là lúc mà bạn tắt trình duyệt kết nối với server đi. Server sẽ nhận thông tin user ngắt kết nối và gửi lại cho toàn bộ client đang được kết nối
- Đây là một ví dụ về sự kiện phía client:
- Khi user click và bảng tên của người mình muốn chat thì client sẽ gửi lên sự kiện "open-chatbox" lên server kèm theo thông tin của người muốn chat. Việc click này và mở "chatbox" giống như việc bạn làm trên facebook vậy

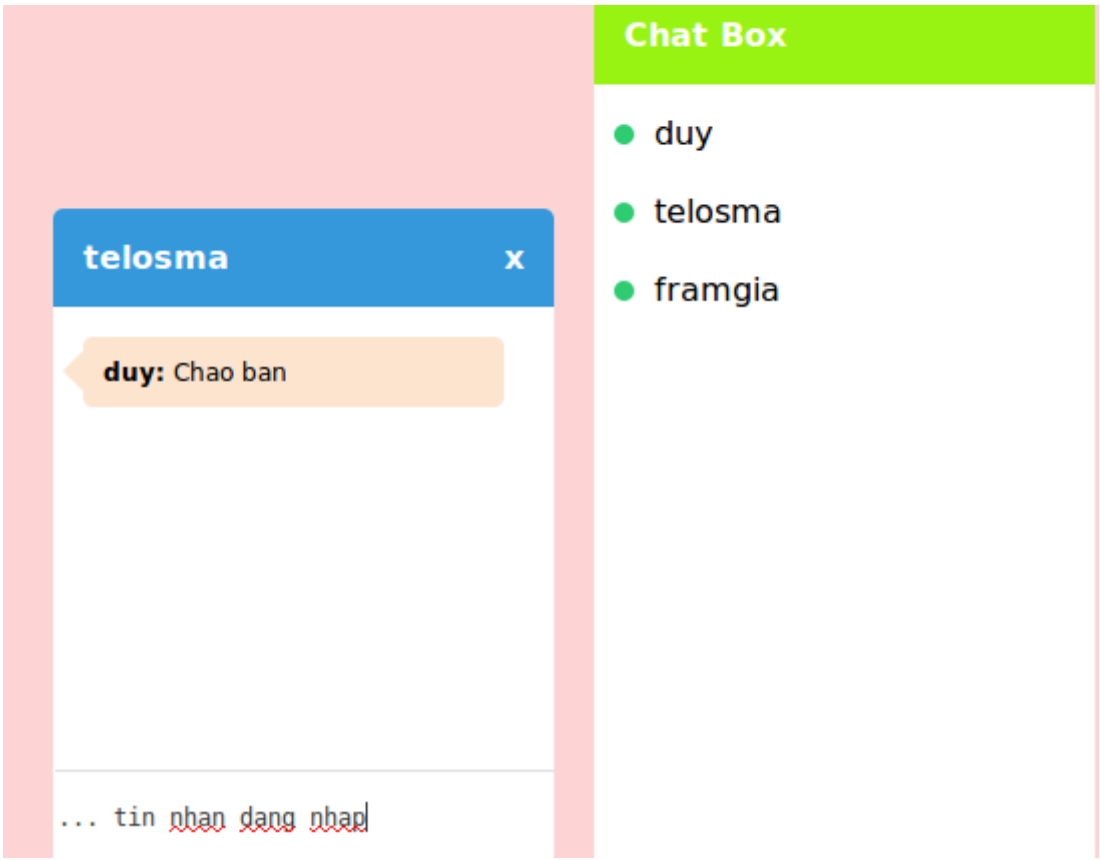
```
        console.log('Hello');
        console.log($(this).text()); // User name

        $('.msg_box').show();
        $('#box_name').html($(this).text());
        socket.emit('open-chatbox', $(this).text());
    });
    }
```

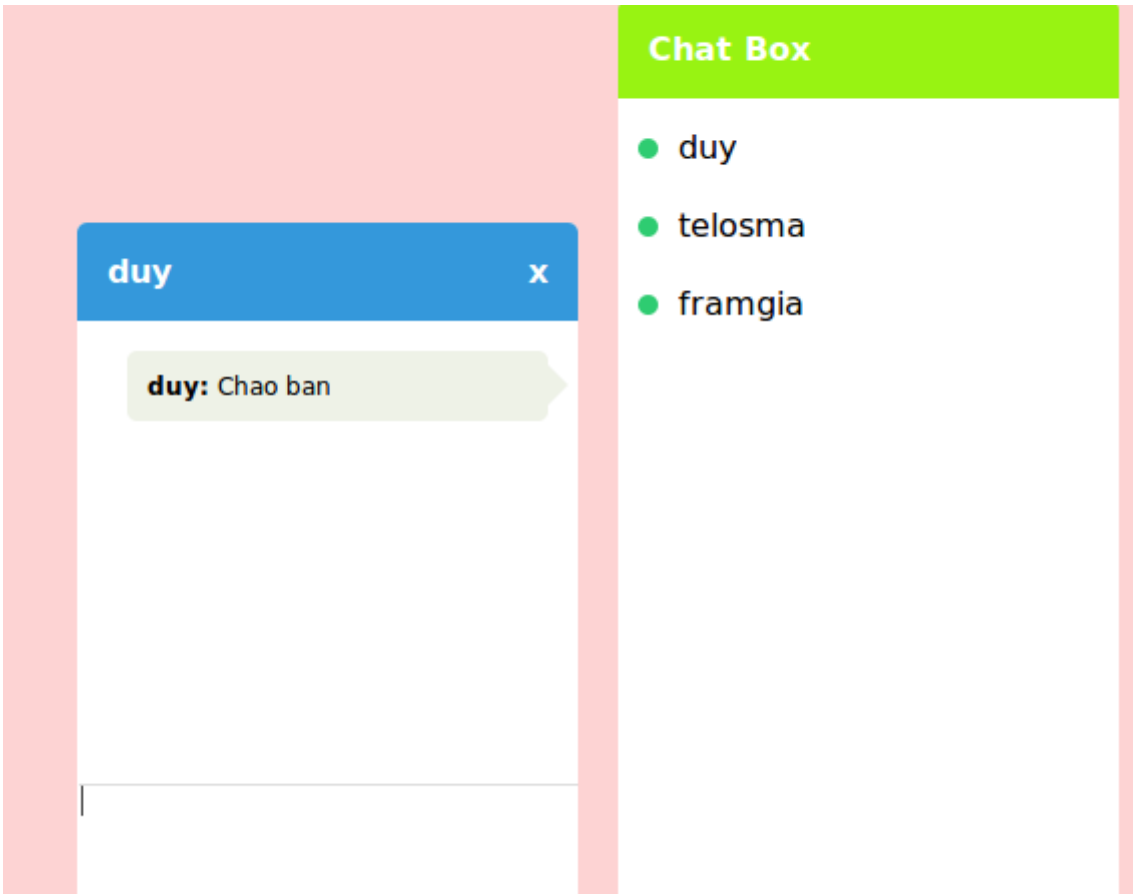
Kết qủa chương trình mình thu được



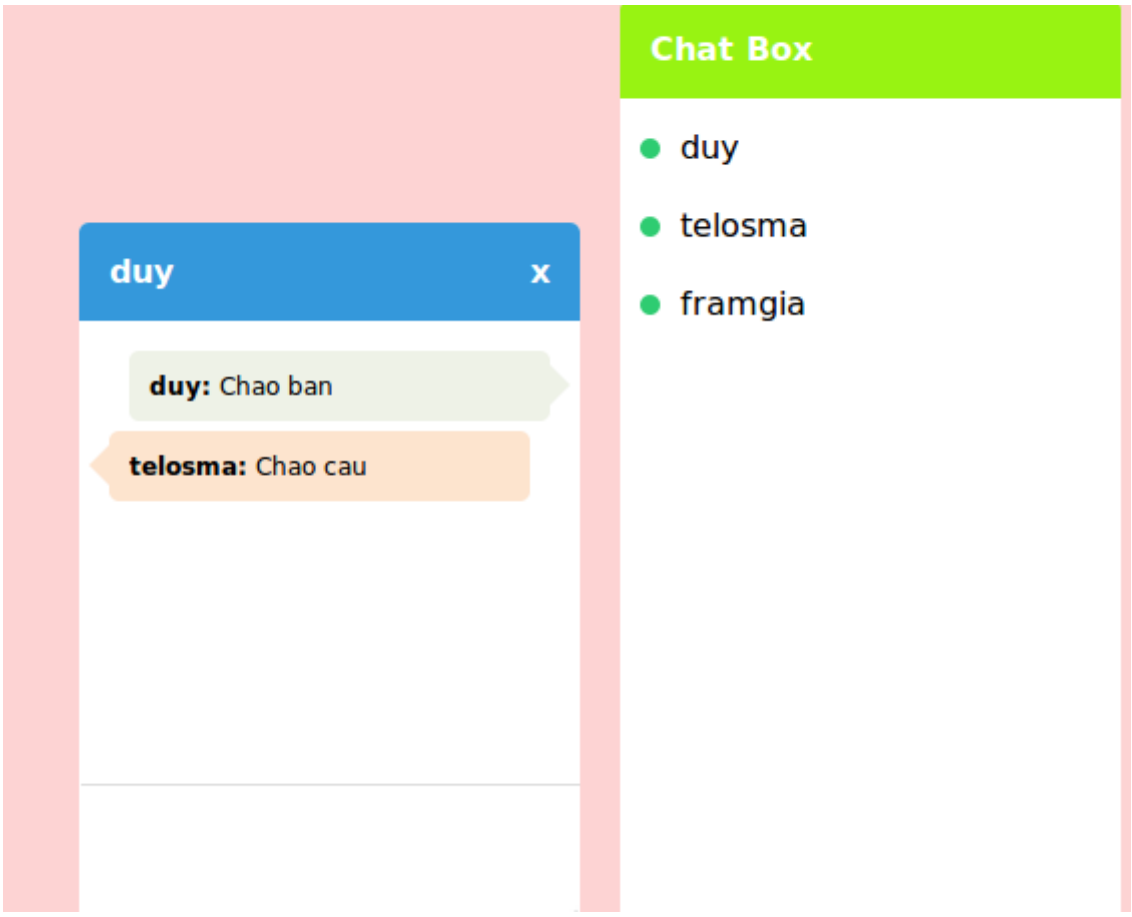
- Giao diện người dùng có username là "duy"



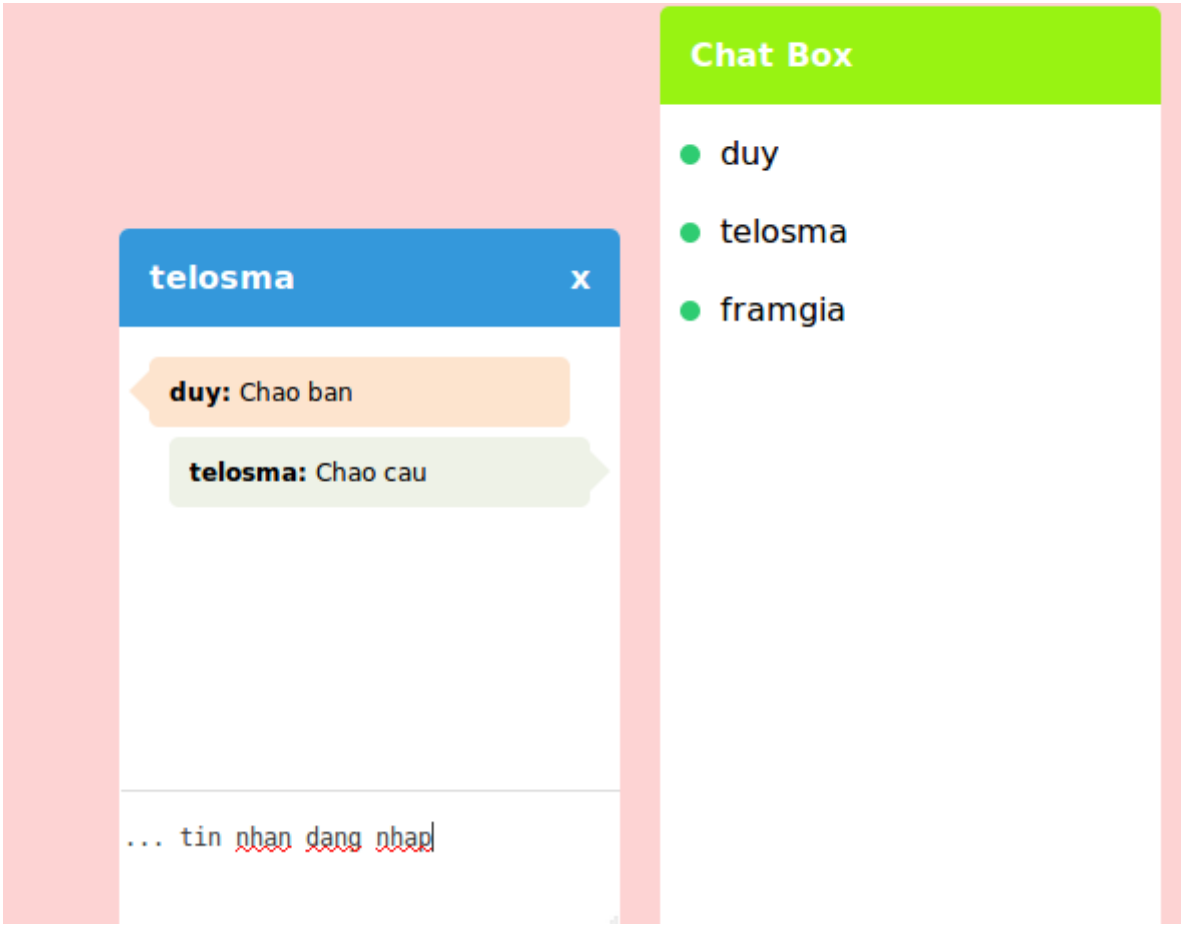
- Giao diện người dùng có username là "telosma"



- Giao diện người dùng có username là "telosma"



- Giao diện người dùng có username là "duy"



Mở rộng ứng dụng

- Chúng ta có thể mở rộng thêm chức năng chát nhóm thông qua việc sử dụng khái niệm "room" trong socket.io

Lời kết

- Mong là bài viết này sẽ có ích cho mọi người, mình rất mong muốn nhận được sự nhận xét và cũng như chia sẻ kiến thức ngược lại từ các bạn. Mình sẽ cố gắng tiếp tục tìm hiểu về NodeJs và đưa ra những bài viết tiếp theo về NodeJs mong nhận được sự ủng hộ từ mọi người.
- Minh xin chân thành cảm ơn!

Related

Từ hàng tỉ phép so sánh đến 10 giây

Luc Duong

Mình đã làm bể cá thông minh như thế nào (phần 1)

Hoàng Hữu Hợi

928 10 13 21

Unit test cho Nodejs RESTful API với Mocha và Chai

Dinh Van Hoang

453 3 4 5

Mình đã làm bể cá thông minh như thế nào - Giới thiệu phần cứng - Wemos (phần 2)

Hoàng Hữu Hợi

403 2 10 6

More from Le Xuan Duy

Cấu hình Virtual Host: myproject.dev thay vì localhost/myproject

Le Xuan Duy

238 3 1 5

Elasticsearch: integrate elasticsearch with laravel (5.2)

Le Xuan Duy

1006 9 0 10

Docker Compose: Xây dựng môi trường phát triển ứng dụng web - PHP, MySql

Le Xuan Duy

1226 11 15 7

Đa ngôn ngữ(Localization-L10n) trong ứng dụng web với Laravel

Le Xuan Duy

Classical
Apr 13th, 5:03 pm

Cảm ơn tuts của bạn. Bạn có thể share code của bạn không

+1

Le Xuan Duy @lengocthuong
Apr 13th, 9:06 pm

Cảm ơn bạn đã comment nhé, mình vừa đẩy lên git https://github.com/telosma/viblo-nodejs-chat
Cài đặt thì bạn đọc trong readme nhé,

0

Classical @telosma
Apr 21st, 9:07 am

Cảm ơn bạn. Mình đang thử tích hợp vào cakePHP mà ít tuts quá. Phần lớn lỗi là do không load được thư viện. Mình đang tìm hiểu thêm. Nếu bạn cũng đang thử với cakePHP thì có gì support mình với nhé

0

Le Xuan Duy @lengocthuong
Apr 21st, 1:35 pm

Mình chưa biết cakePHP nhưng xem qua thì đây cũng là một framework cho Php, do đó bạn cứ tạo server NodeJs là một file js trong ứng dụng của bạn và làm như ứng dụng chat thông thường thôi không cần quan tâm tới code trong cakePHP đâu, mình đã thử tích hợp với Laravel, hoặc bạn có thể tìm hiểu chat với google firebase thay vì socket cũng có khá nhiều bài hướng dẫn đó.
Và ứng dụng demo trên đây của mình thì khá đơn giản, nó chỉ giúp mình hiểu cơ bản về NodeJs và socket.io thôi bạn nhé

0

Classical @telosma
Apr 21st, 1:58 pm

Thanks bạn. Nếu bạn làm chat với chăm sóc khách hàng trên laravel, nếu có thể thì bạn có thể share các tuts hoặc code trên github biết đâu có người cần. Về cơ bản mình cũng có thể học. Mình đang follow bạn trên github rồi :D

0

+5

Cảm ơn bạn nhé, chắt đó mình đang làm đó, giờ đang bận đồ án, hảo hoàn thành mình sẽ viết tut hehe

0

nguyen xuan quang
Apr 20th, 10:12 pm

không chạy được ban ơi. Nó hiện ra lỗi
Store.prototype.**proto** = EventEmitter.prototype;
^

TypeError: Cannot read property 'prototype' of undefined
lên mạng xem fix bug thì nó kêu node 7. trở lên chạy ko được vl thật

+1

Le Xuan Duy @quangakgl
Apr 20th, 11:20 pm

Node mình đang sử dụng là 6.10.0 nhé bạn
Bạn đã fix được chưa?

0

nguyen xuan quang @telosma
Apr 20th, 11:29 pm

Node mình sử dụng là v7.9.0 ko chạy được

0

Le Xuan Duy @quangakgl
Apr 21st, 1:30 am

Bạn ơi mình vừa dùng Docker để cài môi trường Node v7.9 giống bạn để test app nhé, mình có sửa file package.json cập nhật 2 phiên bản mới nhất của express và socket.io đó là
"express": "4.15.2",
"socket.io": "1.7.3"
Bạn xóa folder node_modules và cập nhật phiên bản giống mình rồi "npm install" lại nhé

Các thông số môi trường

+5