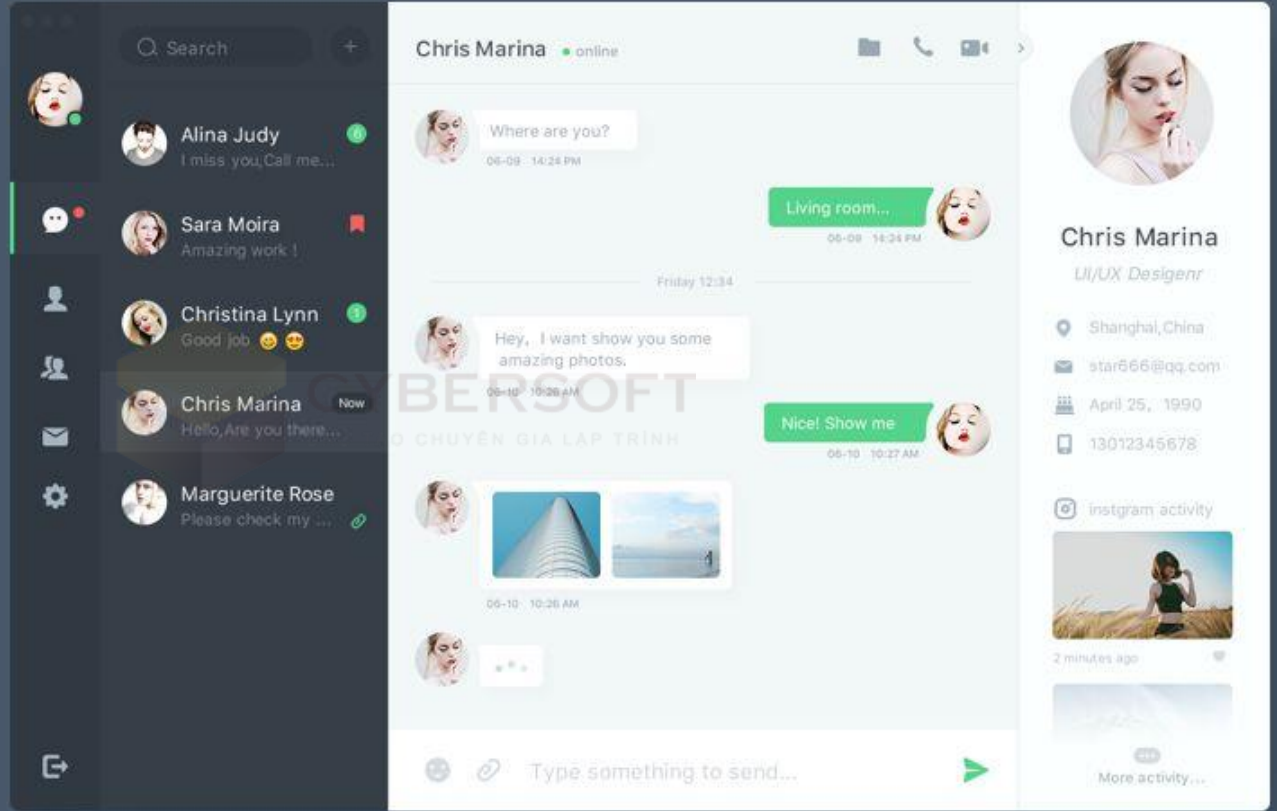




Socket.10

CYBERSOFT
HÀNG ĐẦU CHUYÊN GIA LẬP TRÌNH

Chat App



Các kiến thức được học



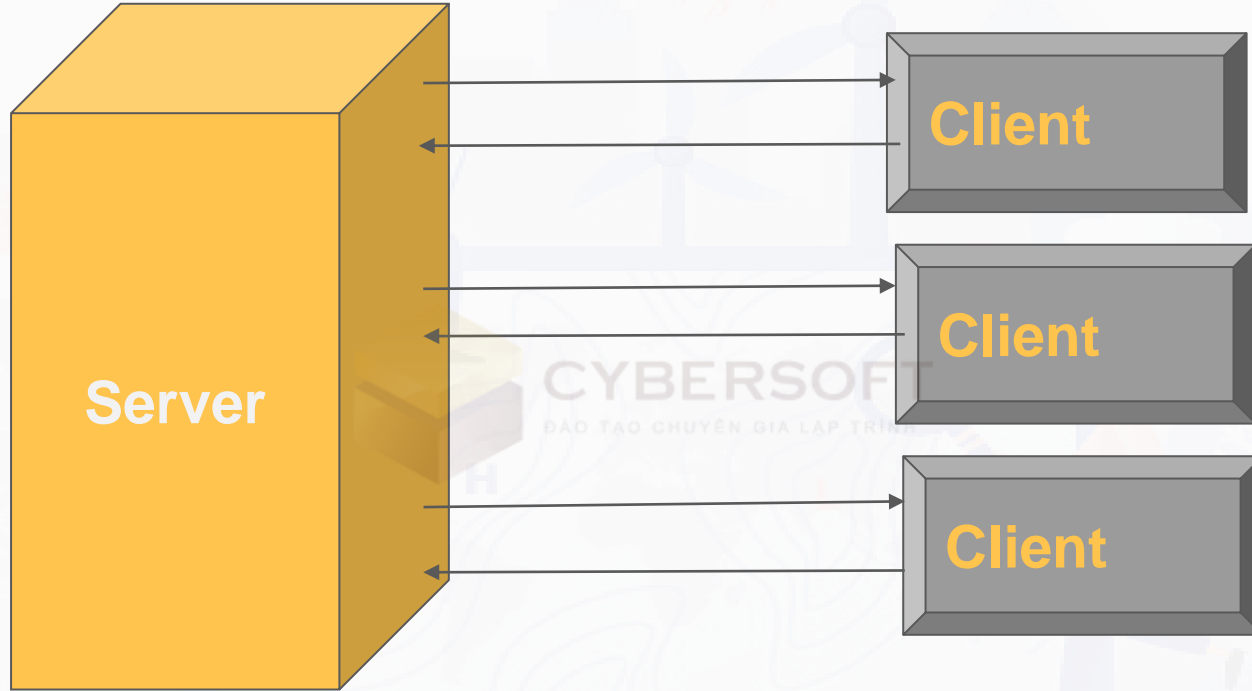
Các kiến
thức về
socket.io

- 01 Kết hợp socket.io vào dự án expressjs
- 02 Tương tác server và client bằng realtime
- 03 Emitting events
- 04 Broadcasting



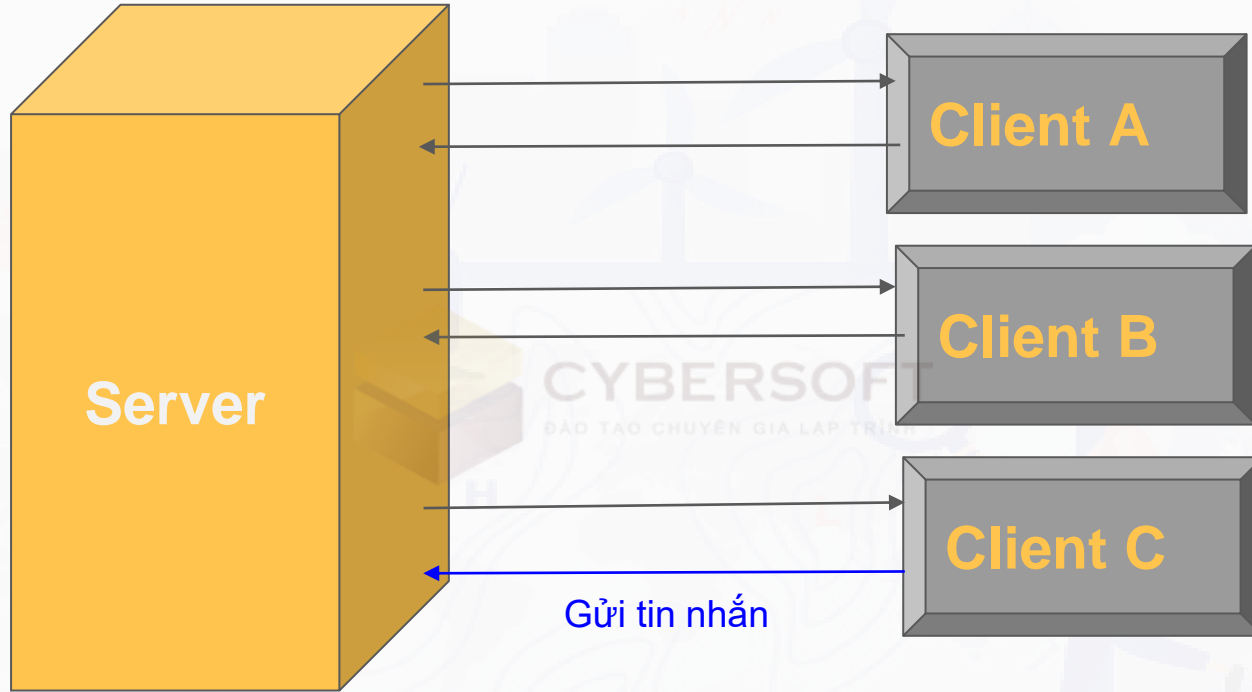
Trang chủ Socket.io : <https://socket.io/>

Realtime



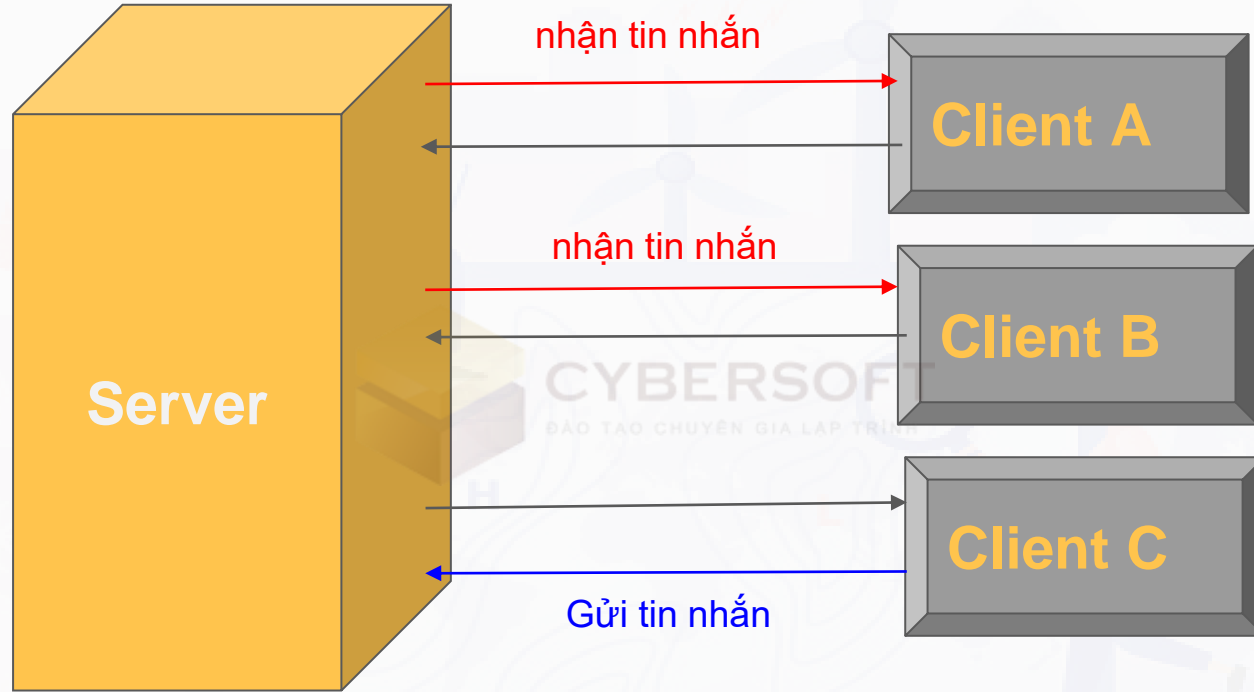
Real time cũng giống như req ,res nhưng nó có tốc độ truyền dữ liệu gần như là bằng thời gian thực tế.

WebSocket



Cơ chế gửi tin nhắn : là gửi từ client lên server

WebSocket



Cơ chế nhận tin nhắn : là gửi từ server về client

Increment Socket

node


JS[®]

+

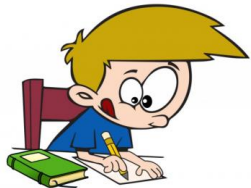


socket.io

Cài Đặt Yarn

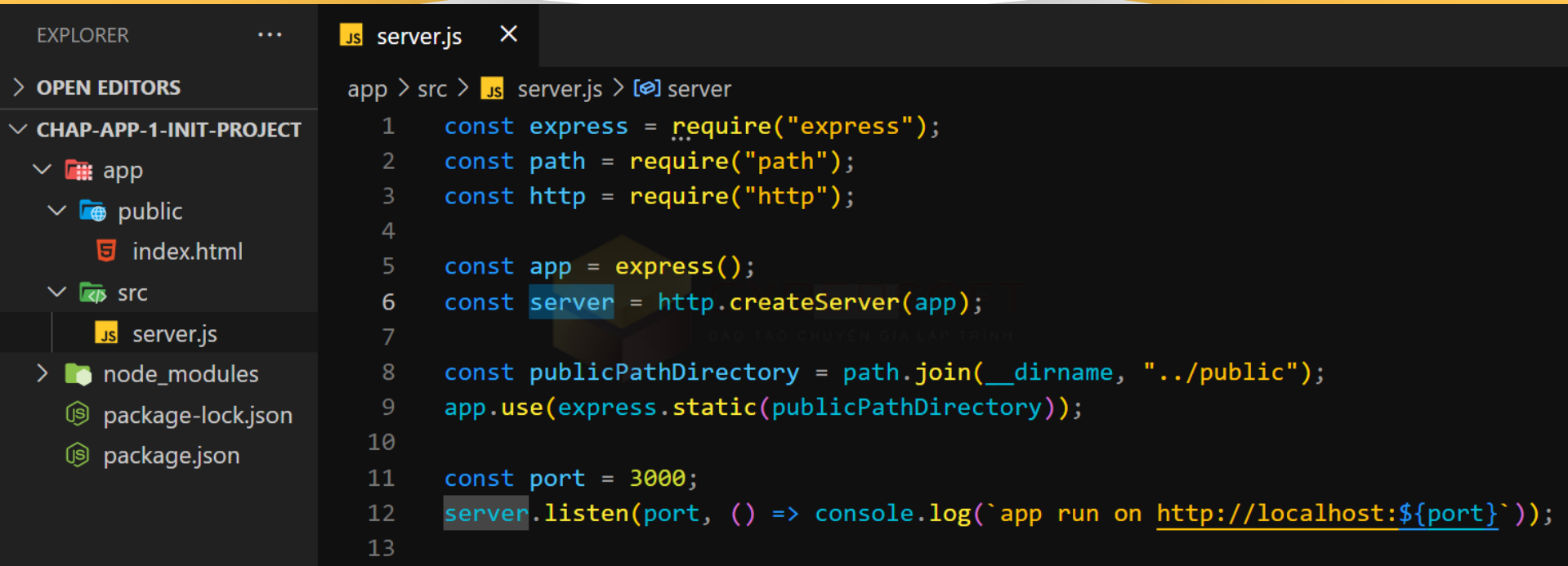


```
EXPLORER  ...  
  
> OPEN EDITORS  
  
✓ CHAP-APP-1-INIT-PROJECT  
  ✓ app  
    ✓ public  
      index.html  
    ✓ src  
      JS server.js  
  > node_modules  
    package-lock.json  
    package.json  
  
JS server.js  ✕  
  
app > src > JS server.js > ...  
1  const express = require("express");  
2  const path = require("path");  
3  const app = express();  
4  
5  const publicPathDirectory = path.join(__dirname, "../public");  
6  app.use(express.static(publicPathDirectory));  
7  
8  const port = 3000;  
9  app.listen(port, () => console.log(`app run on http://localhost:${port}`));  
10 |
```



Khởi tạo Express như các project trước đó nhé !

Khởi tạo Server



The screenshot shows the VS Code interface. On the left, the Explorer sidebar displays a project structure: 'CHAP-APP-1-INIT-PROJECT' containing an 'app' folder (with a subfolder 'public' and 'index.html') and a 'src' folder (containing 'server.js'). The main editor area shows the 'server.js' file with the following code:

```
app > src > JS server.js > [🔗] server
1  const express = require("express");
2  const path = require("path");
3  const http = require("http");
4
5  const app = express();
6  const server = http.createServer(app);
7
8  const publicPathDirectory = path.join(__dirname, "../public");
9  app.use(express.static(publicPathDirectory));
10
11 const port = 3000;
12 server.listen(port, () => console.log(`app run on http://localhost:${port}`));
13
```



Khởi tạo server để giao tiếp websocket với client

Khởi tạo Socket

```
D:\cybersoft\make-course\fullstack-foundation\socket-io\chap-app-1-init-project>npm i socket.io
npm WARN chap-app-1-init-project@1.0.0 No description
npm WARN chap-app-1-init-project@1.0.0 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.3.2 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin", "arch":"any"} (current: {"os":"win32", "arch":"x64"})
```



Cài đặt Socket.io bằng npm

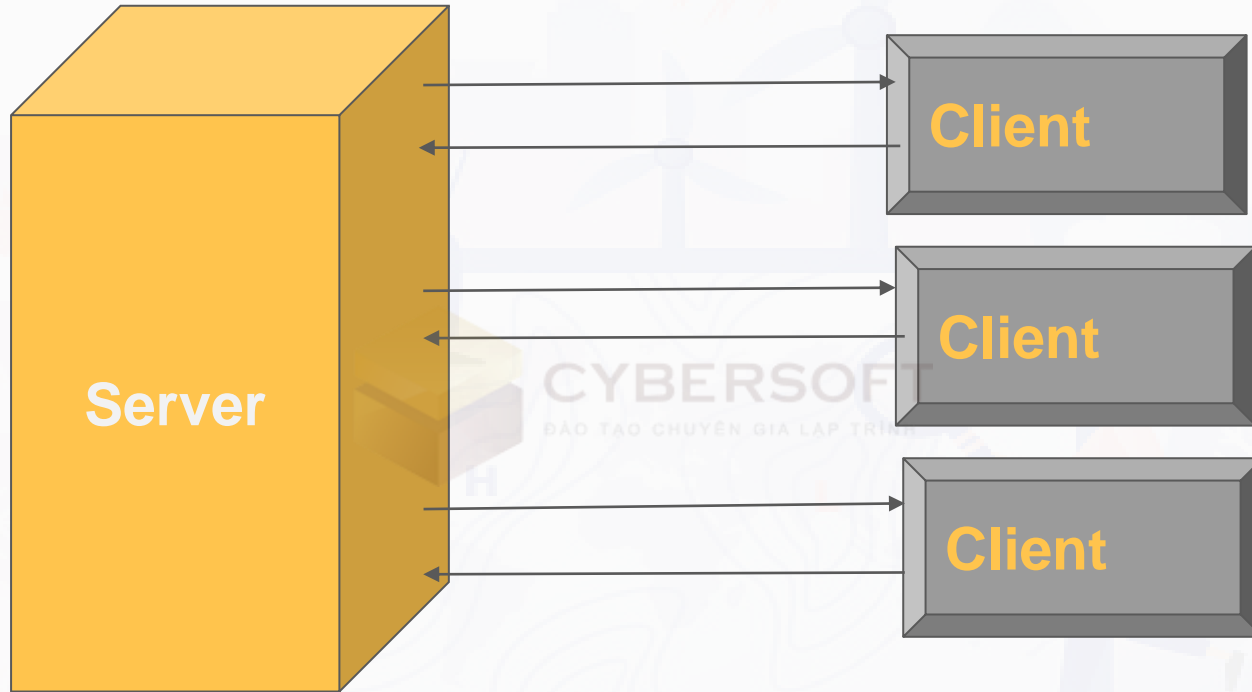
Khởi tạo Socket

JS server.js ✕

app > src > JS server.js > [io]

```
1  const express = require("express");
2  const path = require("path");
3  const http = require("http");
4  const socketio = require("socket.io");
5
6  const app = express();
7  const server = http.createServer(app);
8  const io = socketio(server);
9
10 const publicPathDirectory = path.join(__dirname, "../public");
11 app.use(express.static(publicPathDirectory));
12
13 const port = 3000;
14 server.listen(port, () => console.log(`app run on http://localhost:${port}`));
```

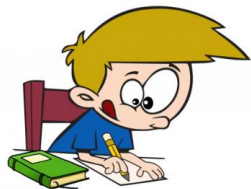
Kết Nối



Thực hành kết nối Server với nhiều Client

Connection Socketio

```
// client kết nối với server
io.on("connection", () => {
  console.log("New Client Connection");
});
```



đây là setup connection ở phía server :

- connection là từ khóa sự kiện kết nối do socketio quy định.
- arrow function là hàm sẽ chạy khi có kết nối từ client tới server

Connection Socketio

```
<!-- nhúng cdn và khởi tạo socketio -->  
<script src="/socket.io/socket.io.js"></script>  
<script>  
  var socket = io();  
</script>
```



đây là setup connection ở phía client :

- nhúng cdn để có thể dùng được socketio ở phía client
- io() là sẽ gửi một sự kiện kết nối lên server

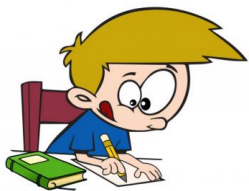
Disconnect Socketio

```
// client kết nối với server
io.on("connection", (socket) => {
  console.log("New Client Connection");
  // client ngắt kết nối với server
  socket.on("disconnect", () => {
    console.log("client disconnected");
  });
});
```



đây là setup disconnect ở phía server :

- disconnect là từ khóa sự kiện ngắt kết nối do socketio quy định.
- arrow function là hàm sẽ chạy khi có sự ngắt kết nối từ client tới server





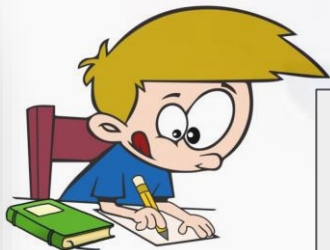
Research Events



Vấn đề

01 Làm sao gửi dữ liệu từ Client lên Server

02 Làm sao gửi dữ liệu từ Client lên Server



- dùng emitting events trong socketio để giải quyết nhé

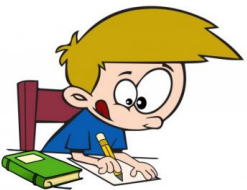
Emitting Events

```
<button id="btn-increment" class="btn btn-info">+1</button>
```

```
var socket = io();  
// sự kiện click  
document.getElementById("btn-increment")  
.addEventListener("click", () => {  
  // gửi events lên server  
  socket.emit("increment");  
});
```

client gửi events increment lên server :

- emit là phương thức giúp chúng ta gửi sự kiện từ client lên server hoặc ngược lại.
- "increment" là tên sự kiện do chúng ta đặt.



Emitting Events

```
let count = 1;
// client kết nối với server
io.on("connection", (socket) => {
  console.log("New Client Connection");
  // nhận event increment từ client gửi lên
  socket.on("increment", () => {
    count++;
    console.log(count);
  });

  // client ngắt kết nối với server
  socket.on("disconnect", () => {
    console.log("client disconnected");
  });
});
```

Server nhận lại event increment từ Client gửi lên :

- on là phương thức giúp chúng ta nhận sự kiện từ client gửi lên server hoặc ngược lại.
- “increment” là tên sự kiện do chúng ta đặt client đặt.



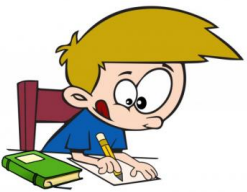
Emitting Events

```
// nhận event increment từ client gửi lên
socket.on("increment", () => {
  count++;
  console.log(count);
  // gửi event sendIncrement từ server về client
  socket.emit("sendIncrement", count);
});
```



Server gửi events về Client :

- emit là phương thức giúp chúng ta gửi sự kiện , tham số thứ 2 là data truyền về cho client
- “sendIncrement” là tên sự kiện do chúng ta đặt.



Emitting Events

```
// client nhận event sendIncrement từ server gửi về
socket.on("sendIncrement", (count) => {
  console.log("count : ", count);
  document.getElementById("content-count").innerHTML = count;
});
```

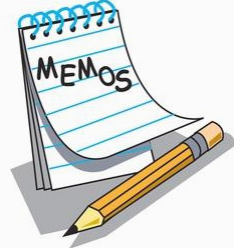
Client Nhận events từ server :

- on là phương thức giúp chúng ta nhận sự kiện từ server gửi về client hoặc ngược lại. Tham số thứ 2 là hàm trả về data cần thiết
- “sendIncrement” là tên sự kiện được thiết kế trên server ,





Tóm lại



Truyền nhận sự kiện

01 Emit là gửi sự kiện

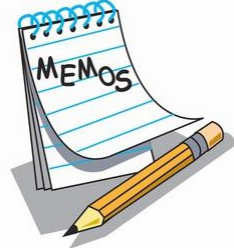
02 On là lắng nghe sự kiện



vậy còn socket và io là gì ?



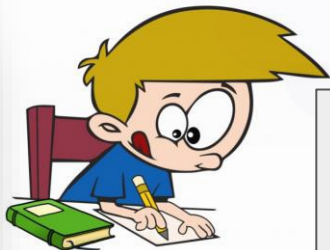
Research Events



Vấn đề

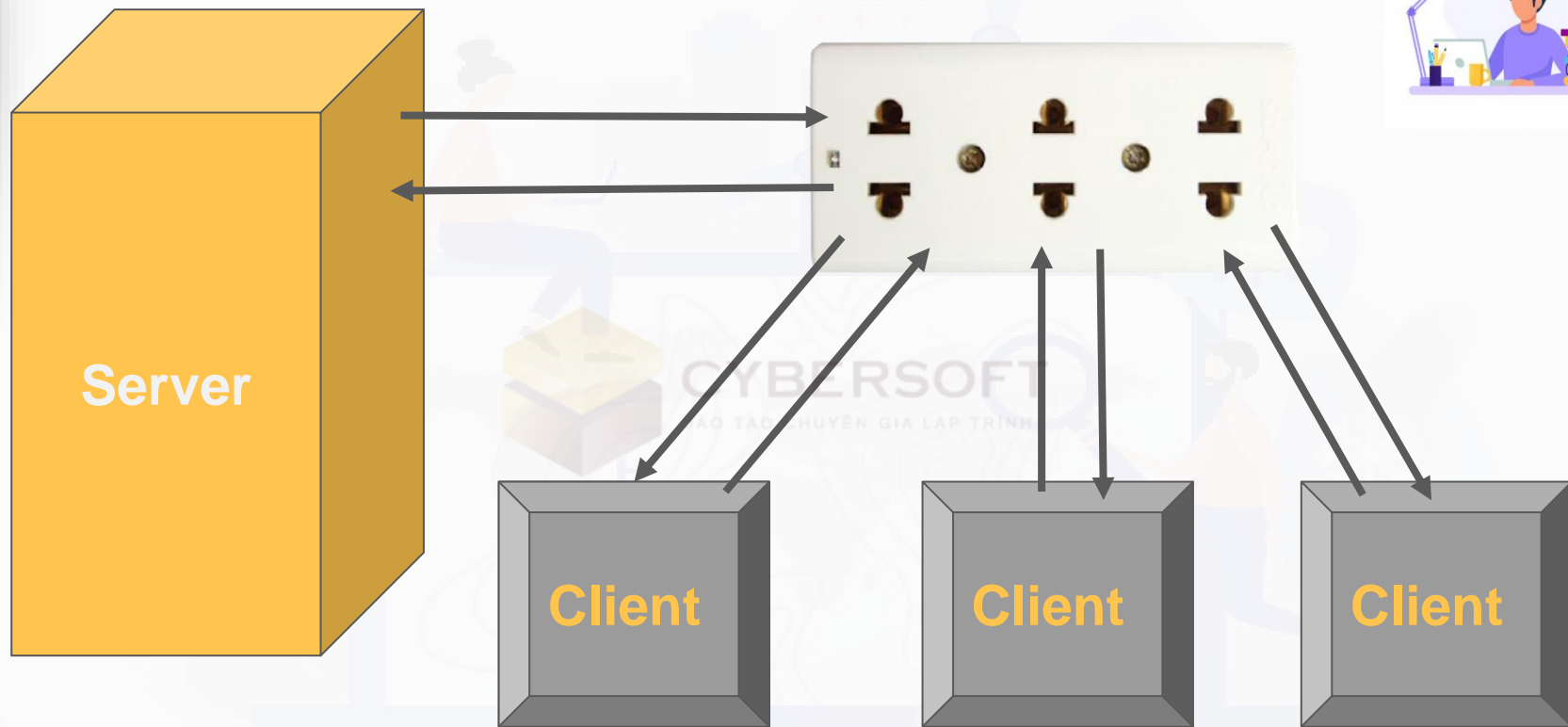
01 Socket là gì ? khi nào dùng ?

02 IO là gì ? khi nào dùng ?

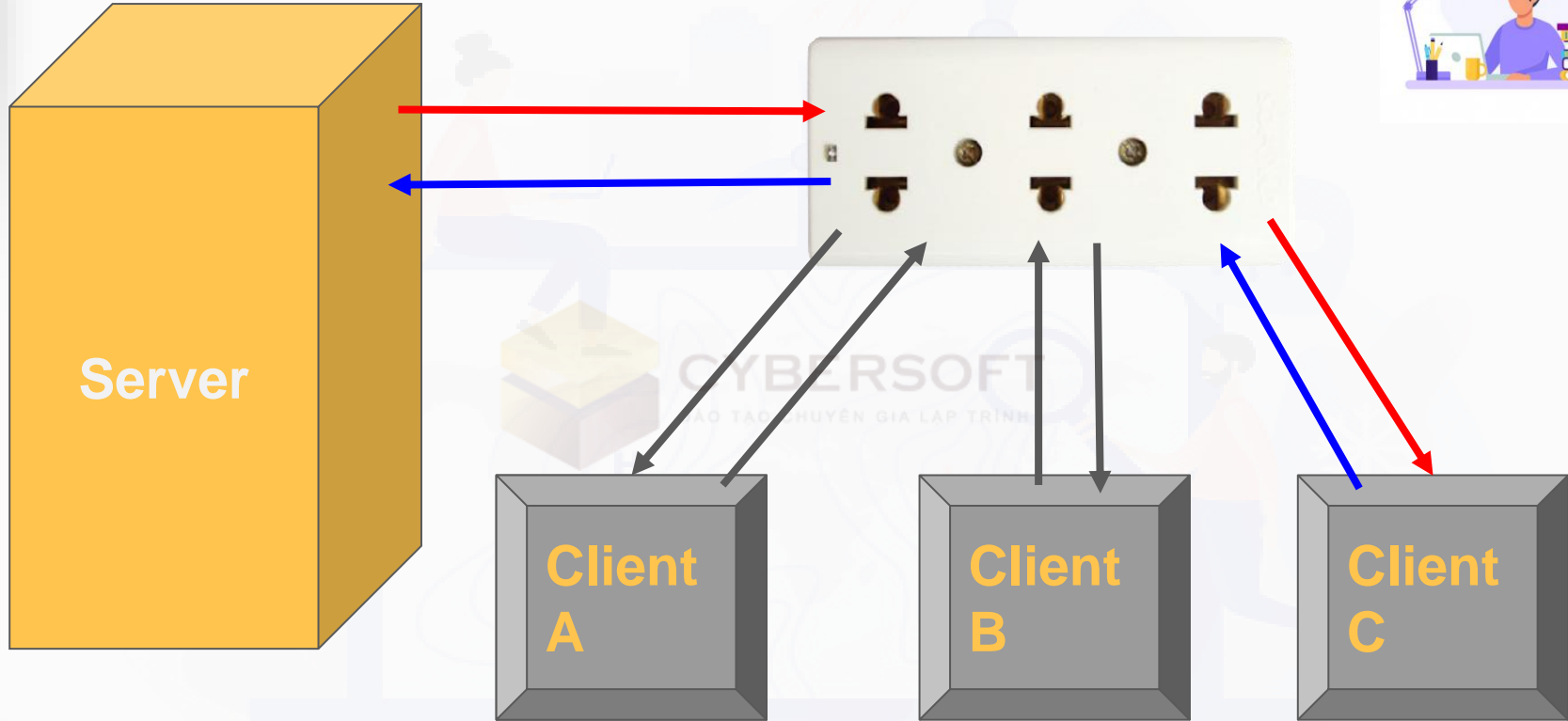


Next Slide

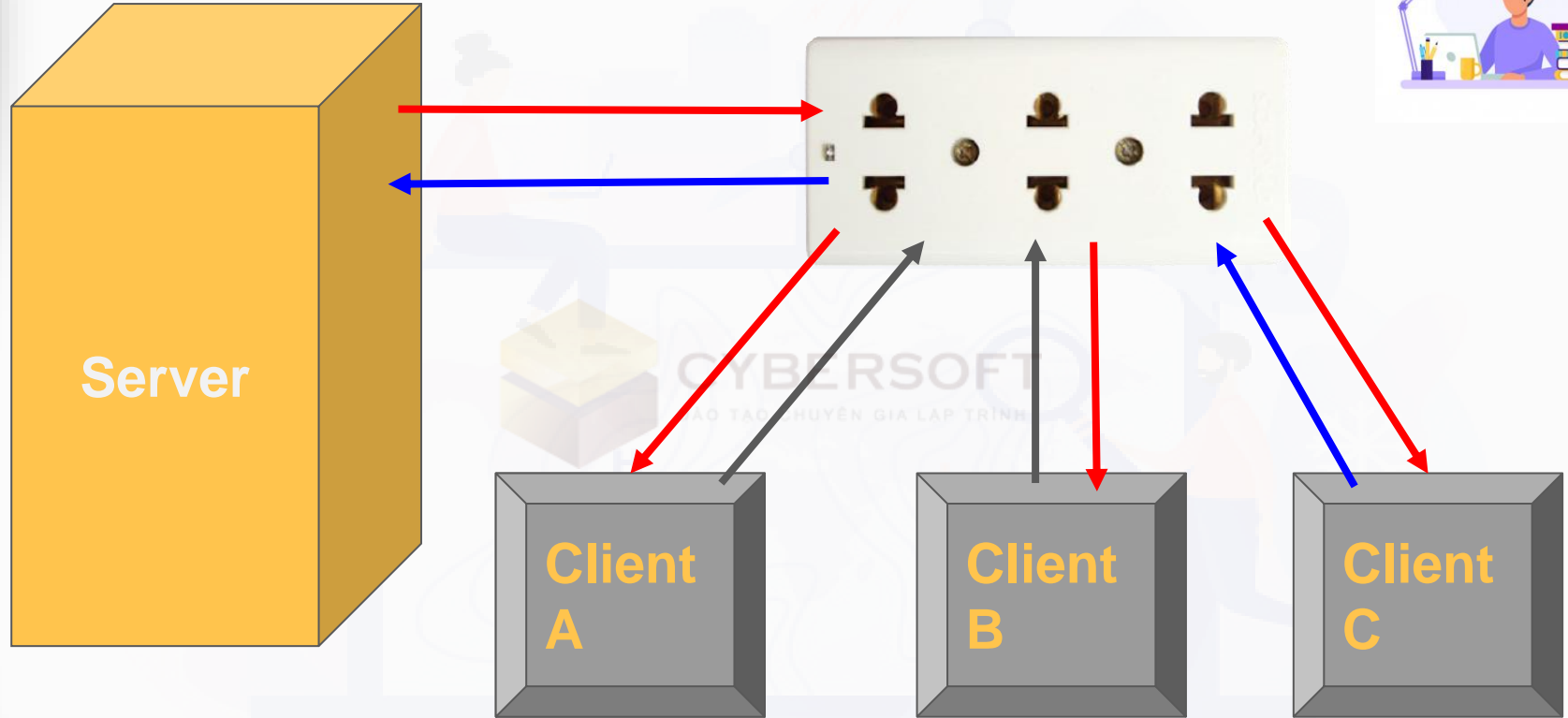
Mô Hình Socket và IO



Dùng Socket để emit event



Dùng IO để emit event



Emitting Events

```
// nhận event increment từ client gửi lên
socket.on("increment", () => {
  count++;
  console.log(count);
  // gửi event sendIncrement từ server về client
  // socket.emit("sendIncrement", count);
  io.emit("sendIncrement", count);
});
```



Server gửi events “sendIncrement” về tất cả client, giúp đồng bộ được biến count.



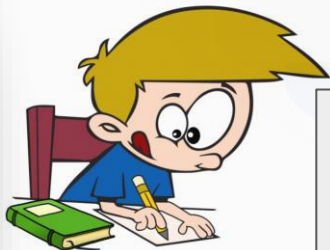
Tóm lại



khi emit nên dùng cái nào ?

01 Dùng socket nếu server chỉ phản hồi cho 1 client (chính là client gửi event lên cho server).

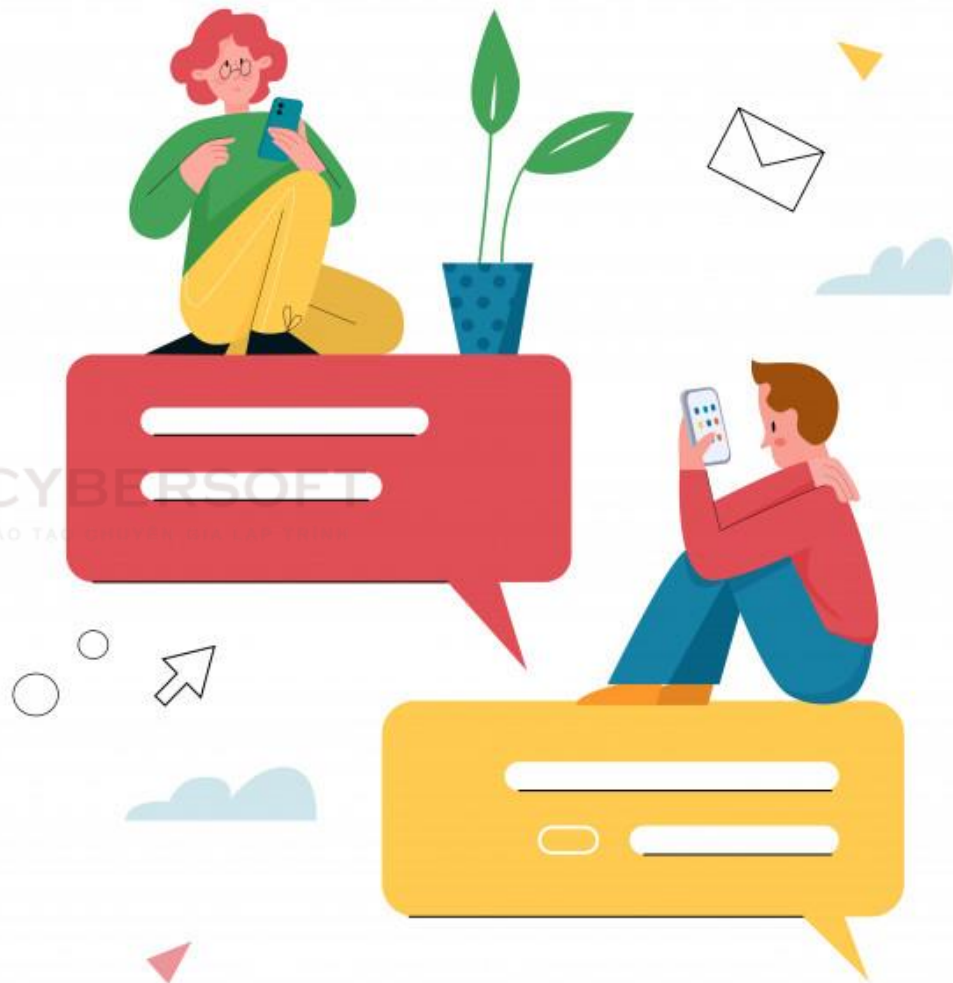
02 Dùng io nếu server muốn phản hồi cho tất cả các client đang kết nối với nó.



Thực hành luôn để hiểu rõ hơn nhé , Next Slide.

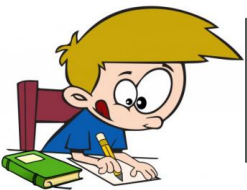
CyberChat

CYBERSOFT
ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH



Giao Diện

```
<main>
  <h1>Chat App</h1>
  <form id="form-messages">
    <input type="text" name="input-message" id="input-message" />
    <button type="submit">Send</button>
  </form>
</main>
```

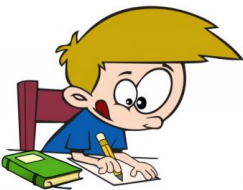


UI cho phép nhập tin nhắn vào , sau đó nhấn send sẽ gửi đi nhé !

Gửi event lên server

```
var socket = io();  
// sự kiện submit  
document  
  .getElementById("form-messages")  
  .addEventListener("submit", (e) => {  
    e.preventDefault();  
    const messageText = document.getElementById("input-message").value;  
    socket.emit("sendMessagesToServer", messageText);  
  });
```


ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH



- form khi submit sẽ bị load trang do đó cần preventDefault nó nhé
- lấy value của input và gửi lên server bằng emit events.

Server xử lý events

```
// nhận event increment từ client gửi lên
socket.on("sendMessagesToServer", (messageText) => {
  // gửi đến tất cả client đang kết nối với server
  io.emit("sendMessagesToClient", messageText);
});
```

- 
- socket.on để nhận event mà client vừa gửi lên.
 - io.emit để gửi messageText đến tất cả client trong server.

Client xử lý Events

```
// nhận lại sự kiện  
socket.on("sendMessagesToClient", (messageText) => {  
  console.log(messageText);  
});
```



- socket.on nhận lại sự kiện
- log messageText lên màn hình console xem trước.



- Chưa hiển thị lên được màn hình , sẽ quay lại dựng giao diện sau nhé , tập trung làm chức năng trước nào !

Bad words



Chức năng xử lý các “bad-words”

khi Client gửi
các từ khóa
tục tũ

01 không cho các từ khóa đó gửi về các client khác .

02 thông báo là tin nhắn không lệ với client đã gửi nó.



Giải quyết vấn đề này như thế nào nhỉ ?

Events Acknowledgements

```
const acknowledgements = () => {  
  console.log("đã gửi tin nhắn");  
};  
socket.emit("sendMessagesToServer", messageText, acknowledgements);
```

```
// nhận event sendMessagesToServer từ client gửi lên  
socket.on("sendMessagesToServer", (messageText, callback) => {  
  // gửi đến tất cả client đang kết nối với server  
  io.emit("sendMessagesToClient", messageText);  
  // gọi lại hàm acknowledgements ở gười client  
  callback();  
});
```

- phía client gửi hàm **acknowledgements** vào làm tham số thứ 3 của **emit**.
- phía server nhận lại hàm **acknowledgements** là **callback**.

Bad-words

```
D:\cybersoft\make-course\fullstack-foundation\socket-io\chap-app-1-init-project>npm install bad-words
npm WARN chap-app-1-init-project@1.0.0 No description
npm WARN chap-app-1-init-project@1.0.0 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.3.2 (node_modules\fsevents):
```



- cài đặt package : npm i bad-words
- giúp chúng ta kiểm tra có từ tục tĩu hay không

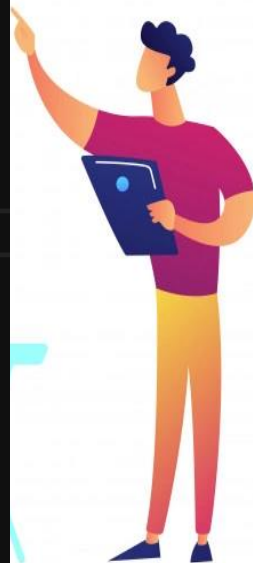


- Làm thế nào để kiểm tra được nhỉ ?

Bad-words server

```
const Filter = require("bad-words");
```

```
// nhận event sendMessagesToServer từ client gửi lên
socket.on("sendMessagesToServer", (messageText, callback) => {
  const filter = new Filter();
  // nếu có từ khóa tục tĩu thì code như sau :
  if (filter.isProfane(messageText)) {
    return callback("tin nhắn không hợp lệ");
  }
  // nếu không có từ khóa tục tĩu thì code như sau :
  // gửi đến tất cả client đang kết nối với server
  io.emit("sendMessagesToClient", messageText);
  // gọi lại hàm acknowledgements ở gười client
  callback();
});
```



Bad-words client

```
const acknowledgements = (errors) => {  
  if (errors) {  
    console.log(("errors :", errors));  
  }  
  console.log("tin nhắn đã được gửi đi");  
};  
socket.emit("sendMessageToServer", messageText, acknowledgements);
```



- nếu có lỗi từ server gọi callback thì thông báo lỗi cho client đó biết
- không lỗi thì thông báo là “tin nhắn đã được gửi đi”.



Chức năng kết nối



Khi có client
kết nối vào
server

01 Gửi cho client đó text : “Chào Mừng đến
với **CyberChat**”

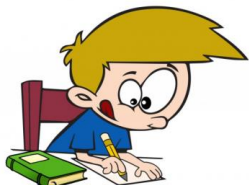
02 Gửi cho các client khác text : “Một người
dùng mới vừa tham gia vào CyberChat”



Giải quyết vấn đề này như thế nào nhỉ ?

Broadcasting

```
// client kết nối với server
io.on("connection", (socket) => {
  console.log("New Client Connection");
  // gửi cho client vừa tham gia vào server
  socket.emit("sendMessagesToClient", "Chào Mừng đến với CyberChat");
  // gửi cho tất cả client còn lại
  socket.broadcast.emit(
    "sendMessagesToClient",
    "Một người dùng mới vừa tham gia vào CyberChat"
  );
});
```

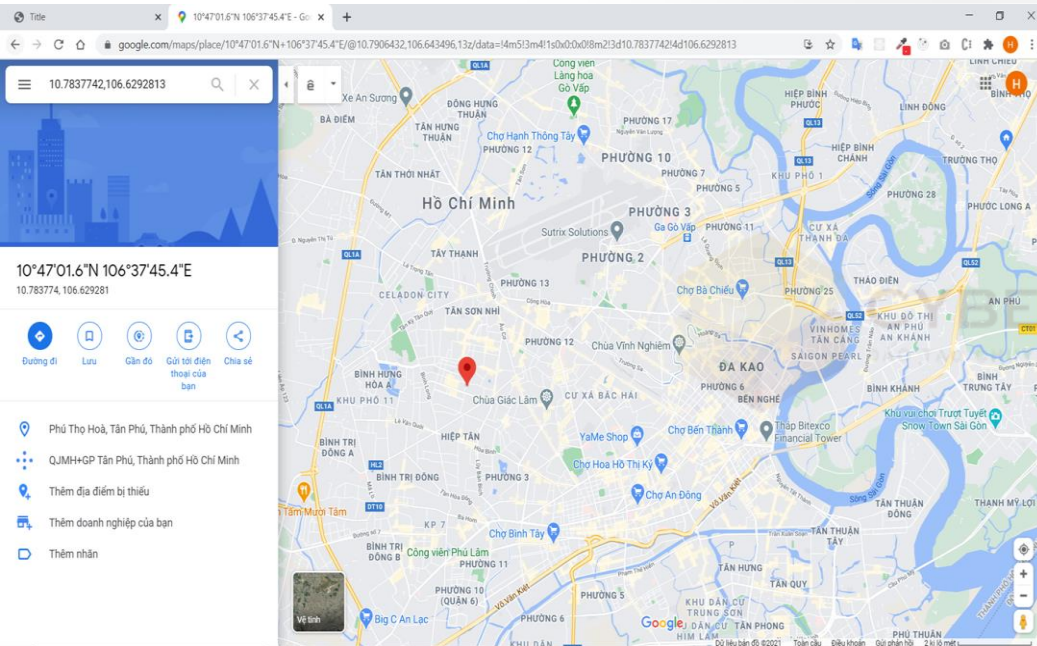


- socket.emit gửi câu chào cho client đã gửi sự kiện lên
- socket.broadcast.emit gửi câu thông báo cho tất cả client còn lại

Share Locations



Lấy vị trí bằng Google Map



- url :
<https://www.google.com/maps?q=10.7815271,106.64093129999999>
- 10.7815271 là vĩ độ
- 106.64093129999999 là kinh độ



- Làm sao lấy được kinh độ và vĩ độ vậy ta ?

Kinh Độ & Vĩ Độ

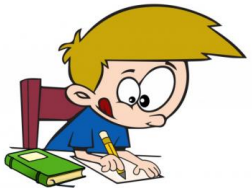
```
// share locations
document
.getElementById("btn-share-location")
.addEventListener("click", () => {
  if (!navigator.geolocation) {
    return alert("trình duyệt không hỗ trợ tính năng này");
  }

  navigator.geolocation.getCurrentPosition((position) => {
    const location = {
      latitude: position.coords.latitude,
      longitude: position.coords.longitude,
    };
    socket.emit("sendLocationToServer", location);
  });
});
```

- xử lý sự kiện click
- dùng navigator để có được : vĩ độ và kinh độ
- socket.emit để gửi kinh độ và vĩ độ lên server

Xử lý Chia Sẻ vị trí

```
// xử lý share location
socket.on("sendLocationToServer", (location) => {
  const linkLocation = `https://www.google.com/maps?q=${location.latitude},${location.longitude}`;
  io.emit("sendLocationToClient", linkLocation);
});
```



- socket.on giúp chúng ta nhận lại kinh độ và vĩ độ do client chia sẻ
- tạo link theo cú pháp google map
- socket.emit để gửi link vị trí cho tất cả client trong server

Các client nhận link vị trí

```
// xử lý sự kiện locations từ server trả về
socket.on("sendLocationToClient", (linkLocation) => {
  console.log(linkLocation);
});
```



- Chưa hiển thị lên được màn hình , sẽ quay lại dựng giao diện sau nhé , tập trung làm chức năng trước nào !

Time



CYBERSOFT
ĐÀO TẠO CHUYÊN NGHIỆP LẬP TRÌNH

Định Dạng thời gian

```
D:\cybersoft\make-course\fullstack-foundation\socket-io\chap-app-1-init-project>npm i date-format
npm WARN chap-app-1-init-project@1.0.0 No description
npm WARN chap-app-1-init-project@1.0.0 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.3.2 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin", "arch":"any"} (current: {"os":"win32", "arch":"x64"})
```



- cài đặt : `npm i date-format`
- package này giúp chúng ta định dạng thời gian chuẩn hơn và dễ dàng hơn



- dùng package này ở đâu (server hay client) ?

xử lý thời gian tạo ra tin nhắn

```
// nhận event sendMessagesToServer từ client gửi lên
socket.on("sendMessagesToServer", (messageText) => {
  const message = {
    messageText,
    createdAt: dateFormat("dd/MM/yyyy - hh:mm:ss", new Date()),
  };
  // gửi đến tất cả client đang kết nối với server
  io.emit("sendMessagesToClient", message);
});
```

- new Date() giúp cho chúng ta có thể lấy được thời gian hiện tại.



Room

CYBER
ĐÀO TẠO CHUẨN QUỐC TẾ



Page Room

The image shows a VS Code editor interface. On the left is the Explorer sidebar with the following structure:

- EXPLORER
- OPEN EDITORS 1 UNSAVED
- CHAP-APP-1-INIT-PROJECT
 - app
 - public
 - js
 - chat.html
 - index.html**
 - src
 - node_modules
 - package-lock.json
 - package.json

The main editor area shows the content of `index.html`:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 > <head> ...
19 </head>
20 <body>
21   <form action="/chat.html">
22     <h1>Join</h1>
23     <input type="text" name="room" id="room" placeholder="room" /
24     <input type="text" name="username" id="username" placeholder=
25     <button type="submit">Join</button>
26   </form>
27 </body>
28 </html>
```

A watermark "ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH" is visible in the background of the code editor.

Url params

```
// query params  
const { room, username } = Qs.parse(location.search, {  
  ignoreQueryPrefix: true,  
});
```



- nhúng cdn : <https://cdnjs.cloudflare.com/ajax/libs/qs/6.9.6/qs.min.js>
- parse : giúp chuyển chuỗi tham số thành object để dễ truy xuất
- ignoreQueryPrefix giúp bỏ dấu “?” khỏi chuỗi tham số

Gửi events lên server

```
// gửi events join lên server  
socket.emit("join", { room, username });
```

ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH



- gửi events join lên server
- data chuyển lên là room và username

Xử lý room

```
// nhận events join từ client
socket.on("join", ({ room, username }) => {
  // xử lý cho client vào phòng
  socket.join(room);
});
```



- join giúp cho client được add vào room phù hợp

Xử lý emit event với room

```
// gửi đến tất cả client đang kết nối với server
io.to(room).emit("sendMessageToClient", message);
```

```
// gửi cho tất cả client còn lại
socket.broadcast
  .to(room)
  .emit(
    "sendMessageToClient",
    `Một người dùng mới vừa tham gia vào ${room}`
  );
```

- tóm lại trước khi emit bạn thêm **.to(room)** vào để biết là mình sẽ gửi event tới room nào

Xử lý emit event với room

```
// gửi đến tất cả client đang kết nối với server  
io.to(room).emit("sendMessageToClient", message);
```

```
// gửi cho tất cả client còn lại  
socket.broadcast  
  .to(room)  
  .emit(  
    "sendMessageToClient",  
    `Một người dùng mới vừa tham gia vào ${room}`  
  );
```

- tóm lại trước khi emit bạn thêm **.to(room)** vào để biết là mình sẽ gửi event tới room nào

User List



Tạo các phương thức

```
let userList = [
  { ... },
  { ... },
  {
    id: 3,
    name: "Nguyễn Phong Dinh",
    room: "sasuke",
  },
];

const addUser = (newUser) => (userList = [...userList, newUser]);
const getUserByRoom = (room) => userList.filter((user) => user.room === room);
const removeUser = (id) =>
  (userList = userList.filter((user) => user.id !== id));

module.exports = {
  addUser,
  getUserByRoom,
  removeUser,
};
```

- tạo danh sách user
- viết các phương thức : addUser , getUserByRoom, removeUser
- export các phương thức đã viết ra.

trả danh sách user về cho client

```
// xử lý user list khi kết nối
io.to(room).emit(
  "send user list from server to client",
  getUserByRoom(room)
);
```


Server

```
// nhận user list và hiển thị lên màn hình
socket.on("send user list from server to client", (userList) => {
  console.log(userList);
});
```

Client

Thêm User khi tham gia

```
// thêm user
const newUser = {
  id: socket.id,
  name: username,
  room: room,
};
addUser(newUser);
// xử lý user list khi kết nối
io.to(room).emit(
  "send user list from server to client",
  getUserByRoom(room)
);
```



- khi join vào room thì chúng ta addUser đó vào userList
- sau đó gửi danh sách mới nhất về cho client

Xóa User khi thoát phòng

```
// ngắt kết nối
socket.on("disconnect", () => {
  removeUser(socket.id);
  // xử lý user list khi kết nối
  io.to(room).emit(
    "send user list from server to client",
    getUserByRoom(room)
  );
  console.log("client left server");
});
```

- khi join vào room thì chúng ta addUser đó vào userList
- sau đó gửi danh sách mới nhất về cho client

User Interface

Living On
Video

CYBER
ĐÀO TẠO CHUẨN QUỐC TẾ



Happy Hacking



CYBERSOFT
ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH