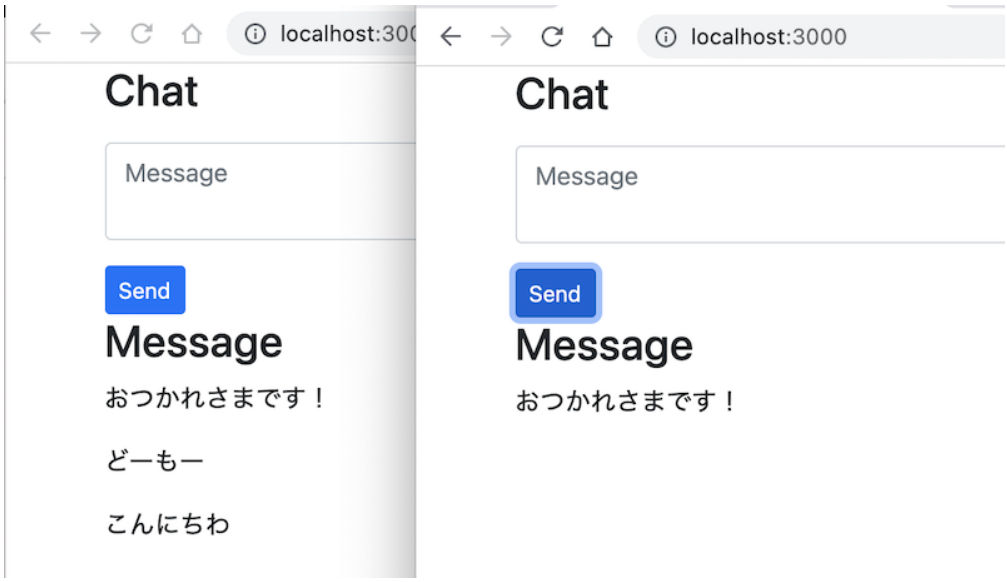
19. Socket.ioでメッセージ送受信

サーバ作成

クライアントから `Socket.io` で送信されたデータをサーバ側で処理します。



ファイル構成

インストール

`npm` で初期化して、必要なモジュールをインストールします。

- socket.io
- express
- dotenv

```
% npm init -y
% npm i socket.io express dotenv
```

.env の作成

`.env` ファイルを作成し、ホストとポートを設定します。

.env

```
HOST=localhost
PORT=3000
```

Express の作成

`server.js` で `Express` を作成します。

```
const express = require('express')
const app = express()
```

http サーバの作成

`http` サーバを作成して express を受け渡します。

```
const http = require('http').createServer(app)
```

socket.io の利用

socket.io モジュールで、http を引数に Socket.io を作成します。

```
const io = require('socket.io')(http)
```

その他設定

`public` フォルダのアクセス許可と、URLエンコードの設定をします。

```
app.use(express.urlencoded({ extended: true }));
app.use(express.static(__dirname + '/public'));
```

サーバ待機

`http` サーバを待機します。

```
http.listen(port, host, () => {
    console.log(`listening on http://${host}:${port}`);
})
```

サーバ受信処理

`connect` イベントで、クライアントの接続が確立したら処理するようにします。 `socket` オブジェクトが利用できます。

```
io.on('connection', (socket) => {

})
```

接続中のユーザに送信

サーバがイベントを受信したら、io.emit() で接続中のすべてのクライントに データ送信します。
イベント名は `message` にしました。

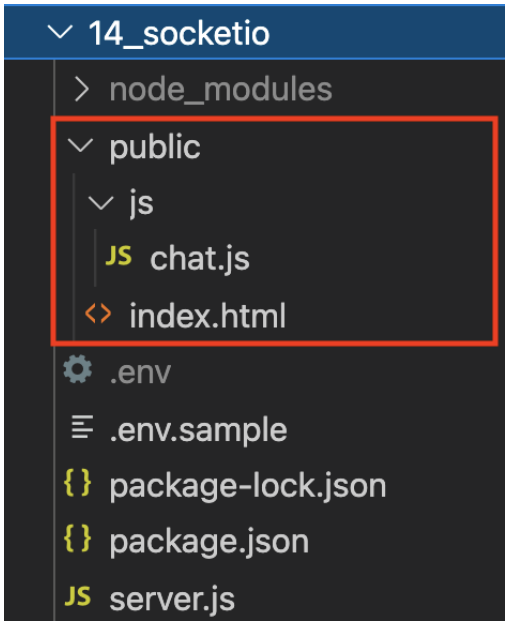
```
io.on('connection', (socket) => {
    socket.on('message', (data) => {
        console.log(data);
        io.emit('message', data);
    })
})
```

クライアント（HTML）

クライアントの `WebSocket` でチャットメッセージを送受信します。

ファイル構成

`public/` に `index.html` と `js/chat.js` を作成して、クライアント処理を実装します。



Socket.io クライアントのインストール

`socket.io.js` を読み込みます。

```
<script type="text/javascript" src="/socket.io/socket.io.js"></script>
```

jQuery インストール

`JS` の処理は簡略化するために `jQuery` を利用します。

```
<script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
```

Bootstrap インストール

`CSS` は `Bootstrap` を利用します。

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/css/bootstrap.min.css"
rel="stylesheet">
```

メインコンテンツ

`body` タグにメッセージ送信フォームとチャット一覧の領域を作成します。また、メインプログラム `js/chat.js` も読み込んでおきます。

```
...
<body>
  <div class="container">
    <h3 class="h3">Chat</h3>
    <div class="form-group">
      <textarea type="text" class="form-control mt-3 mb-3" id="message" placeholder="Message"></textarea>
      <button class="btn btn-sm btn-primary" id="send">Send</button>
    </div>

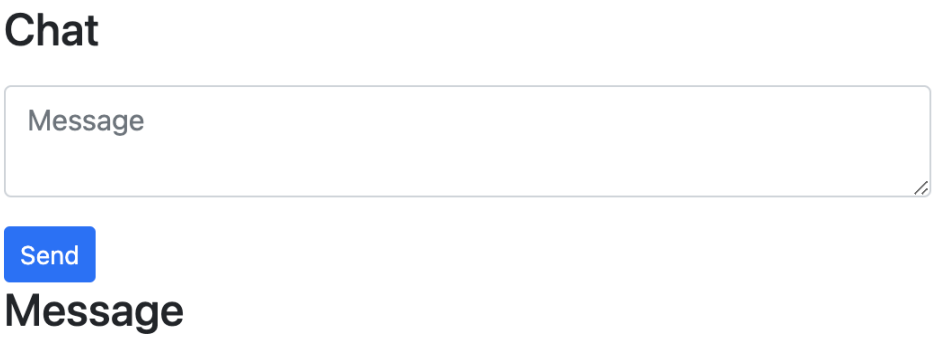
    <h3 class="h3">Message</h3>
    <div id="chatList"></div>
  </div>

  <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/css/bootstrap.min.css"
rel="stylesheet">
  <script type="text/javascript" src="js/chat.js"></script>
</body>
...
```

HTML の確認

`server.js` を起動して、HTMLを確認してみましょう。

```
% node server
```



クライアント（JS）

jQuery の onload イベント

jQuery の onload イベントで、HTML読み込み完了後の処理を実装します。

```
$(() => {
    //この中に処理を実装
})

### サーバ接続
*io.connect()* メソッドにURL指定して、**Socket.io でサーバ接続**します。

```js
$(() => {
    const url = '';
    let socket = io.connect(url);
})
```

DOM の定義
送信メッセージとチャットメッセージの *DOM* を定義します。
```js
$(() => {
    const url = '';
    let socket = io.connect(url);

    const message = $('#message');
    const myChatList = $('#chatList');
})
```

ボタンクリックイベント

jQuery でボタンをクリックイベントを登録します。

```
$('#send').on('click', () => {
    if (!message.val()) return;

    message.val('');
})
```
```

データ送信イベント

`socket.emit()` でクライアントからサーバにデータを送信します。イベント名は、サーバの `socket.on()` とあわせます。

```
socket.emit(イベント名, データ);
```

ボタンをクリックしたら、サーバにメッセージ送信します。イベント名は `message` です。

```
$('#send').on('click', () => {
    if (!message.val()) return;
        socket.emit('message', {
            message: message.val(),
    })
    message.val('');
})
```

データ受信イベント

`socket.on()` でサーバからのイベントを登録できます。 イベント名はサーバで `emit()` したのとあわせます。

```
socket.on(イベント名, 処理);
```

`socket.on()` でサーバ受信イベントを登録します。 イベント名はサーバで設定した `message` です。

```
socket.on('message', (data) => {

})
```

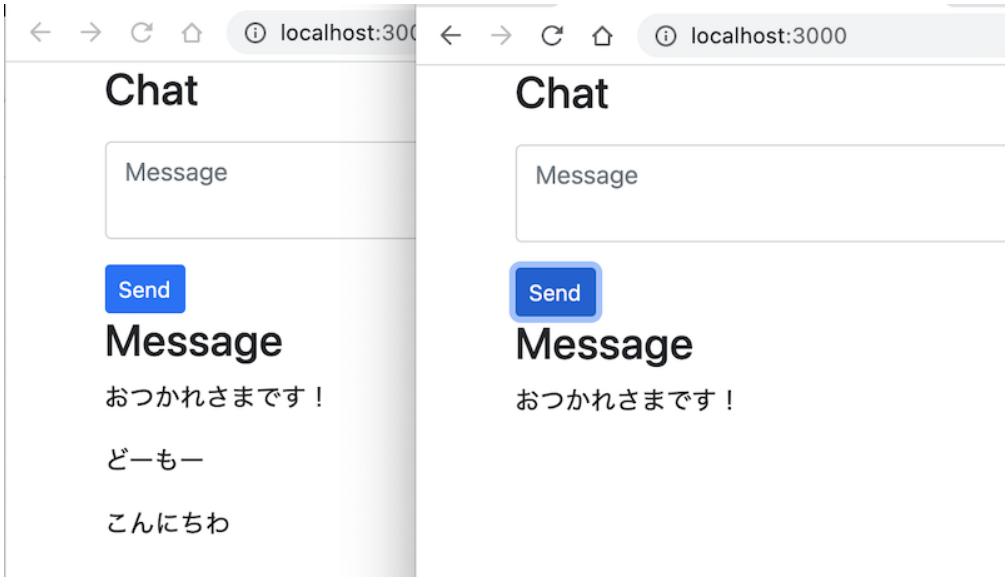
メッセージ表示処理

データを受信したら `HTML` にメッセージを表示します。

```
socket.on('message', (data) => {
    console.log('message', data)
    let chatElement = $('<p>').append(data.message);
    myChatList.prepend(chatElement);
})
```

動作確認

複数のブラウザで、チャットメッセージの送受信を確認してみましょう。



サーバログ

サーバ側は、コンソールでログが確認できます。

```
{ message: 'こんにちわ' }
{ message: 'こんにちわ' }
{ message: 'どーもー' }
{ message: 'おつかれさまです！\n' }
```

クライアントログ

クライアント側は `DevTools` でログが確認できます。

```
message ▼Object i
            message: "こんにちわ"
          ▶[[Prototype]]: Object
message ▼Object i
            message: "どーもー"
          ▶[[Prototype]]: Object
message ▼Object i
            message: "おつかれさまです！\n"
          ▶[[Prototype]]: Object
```

ソース

server.js

```
const express = require('express');
const app = express();
const http = require('http').createServer(app);
const io = require('socket.io')(http);

const dotenv = require('dotenv');
dotenv.config();
const host = process.env.HOST
const port = process.env.PORT

app.use(express.urlencoded({ extended: true }));
app.use(express.static(__dirname + '/public'));

http.listen(port, host, () => {
    console.log(`listening on http://${host}:${port}`);
})

io.on('connection', (socket) => {
    socket.on('message', (data) => {
        console.log(data);
        io.emit('message', data);
    })
})
```

public/js/chat.js

```
$(() => {
    const url = '';
    let socket = io.connect(url);

    const message = $('#message');
    const myChatList = $('#chatList');

    // メッセージ受信
    socket.on('message', (data) => {
        console.log('message', data)
        let chatElement = $('<p>').append(data.message);
        myChatList.prepend(chatElement);
    });

    // メッセージ送信
    $('#send').on('click', () => {
        if (!message.val()) return;
        socket.emit('message', {
            message: message.val(),
        });
        message.val('');
```

```
    });
})
```

public/index.html

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="UTF-8">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/css/bootstrap.min.css"
 rel="stylesheet">
  <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
  <script type="text/javascript" src="/socket.io/socket.io.js"></script>
</head>

<body>
  <div class="container">
    <h3 class="h3">Chat</h3>
    <div class="form-group">
      <textarea class="form-control mt-3 mb-3" id="message" placeholder="Message"></textarea>
      <button class="btn btn-sm btn-primary" id="send">Send</button>
    </div>

    <h3 class="h3">Message</h3>
    <div id="chatList"></div>
  </div>

  <script type="text/javascript" src="js/chat.js"></script>
</body>

</html>
```