How to Create a Chatbot With the ChatGPT API

- https://www.freecodecamp.org/news/how-to-create-a-chatbot-with-the-chatgpt-api/

- menu

OpenAI's ChatGPT is a great tool for getting information as quickly as possible for your coding projects. Even better, you can now integrate the artificial intelligence-powered chat capability of OpenAI's models directly into your application.

Recently, the OpenAI team expanded their API by giving developers access to their pretrained AI models (DALL-E, Codex, and GPT-3). This means that you can send a question to the API, get the response, and use the data in your application, all within seconds.

In this article, you'll learn how to create an OpenAI account, retrieve your API keys and query OpenAI's GPT-3 model from your Node.js application. Let's dive right in!

OpenAI的ChatGPT是一個極為方便的工具，可快速獲取有關編碼專案的資訊。更重要的是，現在您可以直接將OpenAI模型的人工智慧聊天功能整合到您的應用程式中。

最近，OpenAI團隊擴展了他們的API，使開發者可以存取他們預訓練的AI模型（DALL-E、Codex和GPT-3）。這意味著您可以向API發送問題，獲取回應，並在幾秒內在您的應用程式中使用數據。

在這篇文章中，您將學會如何創建一個OpenAI帳戶，檢索您的API金鑰，並從您的Node.js應用程式中查詢OpenAI的GPT-3模型。讓我們馬上開始吧！

How to Sign Up for a ChatGPT Account

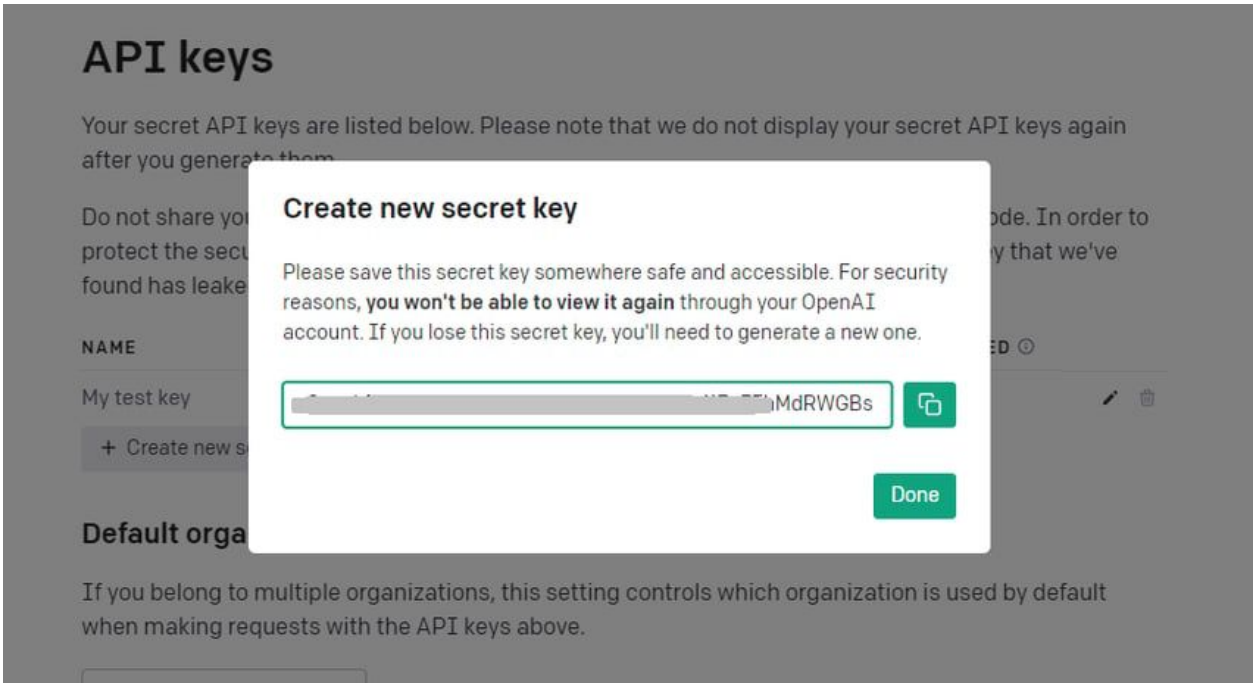
The first thing you need to do is sign up for an OpenAI account if you don't already have one. Once you're in, you'll be redirected back to the homepage.

At the top right corner of the page, click on your profile image, then click on Manage Account. On the sidebar, click API Keys and then click the **create new secret key** button to generate a secret key:

如何註冊ChatGPT帳戶

如果您尚未擁有OpenAI帳戶，首先需要註冊一個。一旦進入，您將被重新導向回主頁。

在頁面的右上角，點擊您的個人資料圖像，然後點擊「Manage Account」。在側邊欄中，點擊「API Keys」，然後點擊「create new secret key」按鈕以生成一個秘密金鑰：



Screenshot of secret key

Copy the secret key and paste it somewhere safe and accessible because you'll need it later to connect your application with the OpenAI API.

With the key safely stored, the next step is to create a Node.js project and spin up an Express server on top of it. Let's start with the installation and basic setup.

複製這個秘密金鑰，並將它貼到一個安全且可存取的地方，因為稍後您將需要它來連接您的應用程式與OpenAI API。

有了金鑰安全地儲存，下一步是創建一個Node.js專案並在其上啟動一個Express伺服器。讓我們從安裝和基本設置開始。

How to Set up the Project

To follow along with this project, you need to have Node.js and npm installed on your local machine. The latest version of Node.js comes with npm, and it's available on the official Node.js website.

Start by creating an empty directory in your computer. Next, launch the command prompt and `cd` into the folder you just created:

設置專案步驟

要參與此專案，您需要在本地機器上安裝Node.js和npm。Node.js的最新版本附帶npm，可在官方Node.js網站上獲得。

首先，在您的計算機上創建一個空目錄。接下來，啟動命令提示字元並進入您剛創建的文件夾：

```
cd path/to/project
```

Once pointed to the directory, run the following command to create a Node project:

指向目錄後，執行以下命令來創建一個Node專案：

```
npm init -y
```

Next, run the following command to install `express` and `openai` libraries from npm:

接下來，執行以下命令從npm安裝express和openai庫：

```
npm i express openai
```

The next step is to create the server.

How to Create an Express Server

For now, this server will only serve the static files. We'll implement the chat API towards the end of this article.

如何創建Express伺服器

目前，此伺服器將僅提供靜態文件。我們將在本文末尾實現聊天API。

Start by creating a file named **server.js** inside the root folder of your project. Next, open the file with a code editor and add the following code:

首先，在您的專案根目錄內創建一個名為`server.js`的文件。然後，用代碼編輯器打開文件並添加以下代碼：

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

app.use(express.static('public'))
```

```
app.listen(5000, ()=> {
    console.log("Server is active")
})
```

With this code, you've created a web server that serves static files (i.e. HTML, CSS) from the **/public** folder.

Next, we'll create the HTML file that renders the chat interface on the web page, as well as the stylesheet file and the JavaScript file.

這段代碼創建了一個網頁伺服器，從 `/public` 文件夾提供靜態文件（即HTML、CSS）。

接下來，我們將創建一個HTML文件，在網頁上呈現聊天界面，以及樣式表文件和JavaScript文件。

How to Create the Chatbot

Start by creating a folder named public inside the root of your project. Then inside the **/public** directory, create a file named **index.html**.

Open the file with a text editor and add the following markup in the file:

如何創建聊天機器人

首先，在專案的根目錄內創建一個名為**public**的文件夾。然後，在 `/public` 目錄內，創建一個名為 `index.html` 的文件。

用文本編輯器打開文件，將以下標記添加到文件中：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-s
    <title>Document</title>
</head>
<body>
    <div id="chat-area">
```

```
    </div>

    <div class="submit-form">
        <div class="input">
            <textarea name="input" id="input" cols="40" rows="3"
            <button id="btn">Submit</button>
        </div>
    </div>

</body>
</html>
```

As you can see above, the page comprises of the chat area (where the messages are displayed) and the submit area (comprising the text area and the submit button).

To style the page, add the following stylesheet between the opening and closing `<head>` tags in your **/public/index.html** file:

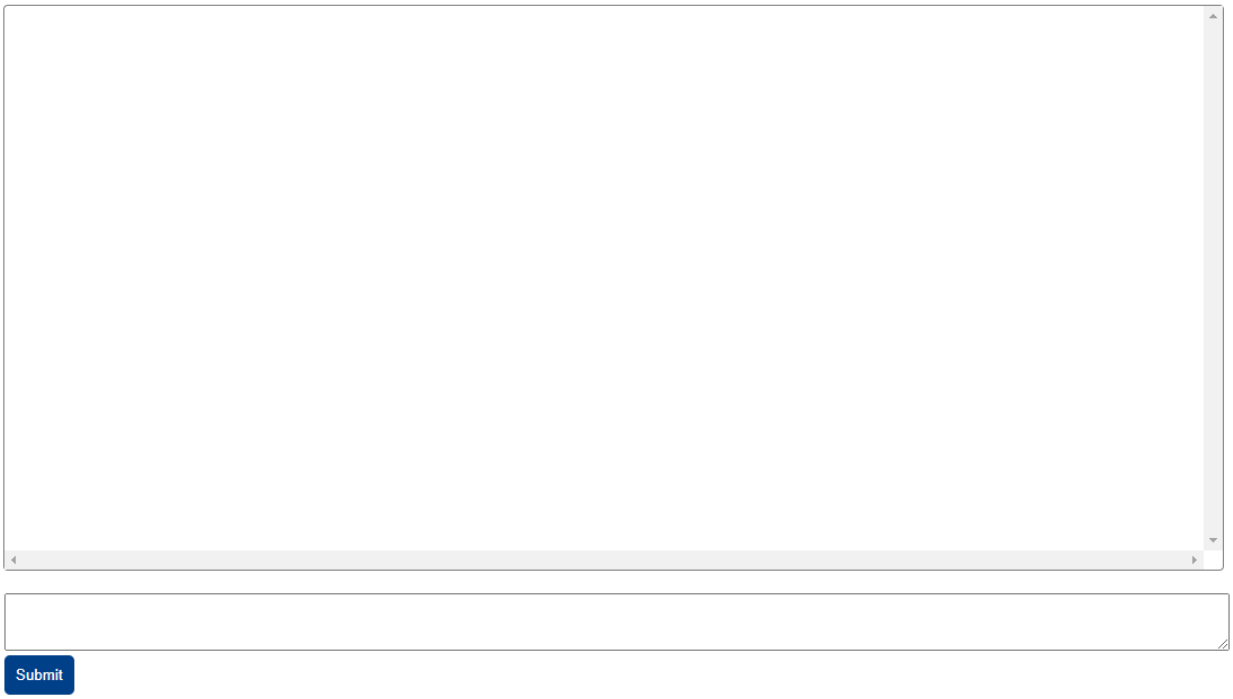
如您上方所見，該頁面包含聊天區（顯示訊息的區域）和提交區（包括文本區域和提交按鈕）。

為了為頁面添加樣式，在您的 **/public/index.html** 文件的開頭和結尾 `<head>` 標籤之間添加以下樣式表：

```
<style>
    #chat-area {
        margin: 0 auto;
        width: 80%;
        height: 500px;
        overflow:scroll;
        border: 1px solid gray;
        border-radius: 4px;
    }

    .input {
        width: 100%;
    }
```

```
    .submit-area{
        justify-content: center;
        display: flex;
        margin: 20px auto;
        width: 80%;
    }

    textarea {
        width: 100%;
    }

    .box {
        width: 96%;
        margin: 0 auto;
        padding: 10px 10px;
        background-color: #C4DBFE;
        margin: 10px auto;
    }

    .answer {
        background-color: aquamarine;
    }

    button {
        background-color: #004089;
        color: white;
        padding: 10px 10px;
        border-radius: 5px;
        border: none;
    }
</style>
```

If you save the file and check the browser, you should find your page like this:



Screenshot of the page

The chat area is empty for now because we haven't submitted any messages yet. To do that, we need to bring in JavaScript.

聊天區現在是空的，因為我們還沒有提交任何訊息。要做到這一點，我們需要引入JavaScript。

How to Add Front-end JavaScript

When the user inputs a message in the text area and clicks on the submit button, we'll send the message to the backend, get the response from the API and display it on the page.

Start by adding an empty script element within the `<body>` tags in **index.html**:

如何添加前端JavaScript

當用戶在文本區域輸入訊息並點擊提交按鈕時，我們將訊息發送到後端，從API獲取回應並顯示在頁面上。

首先，在**index.html**的`<body>`標籤內添加一個空的script元素：

```
<script>

</script>
```

Inside the script tags, we're to call the `getResponse()` function whenever the user clicks on the submit button:

在script標籤內，我們將在用戶點擊提交按鈕時調用`getResponse()`函數：

```
const btn = document.getElementById("btn")

btn.addEventListener('click', getResponse)
```

The `getResponse` function essentially gets the user's question, sends it to our Node.js backend to fetch the answer, and displays the response on the page.

Inside the function, we start by accessing the prompt submitted by the user:

`getResponse`函數基本上獲取用戶的問題，將其發送到我們的Node.js後端以獲取答案，並在頁面上顯示回應。

在函數內部，我們首先獲取用戶提交的提示：

```
async function getResponse() {
  var inputText = document.getElementById("input").value
  const parentDiv = document.getElementById("chat-area")

  // 以下的代碼將放在這個函數內
```

If the value of the text area is empty, we simply return nothing:

如果文本區域的值為空，我們就什麼都不回傳：

```
if(inputText === '') { return }
```

Otherwise, we first add the question to the chat area in the UI:

否則，我們首先將問題添加到UI中的聊天區：

```
const question = document.createElement('div')
question.innerHTML = inputText
question.classList.add("box")
parentDiv.appendChild(question)
```

Next, we reset the text area so it's blank:

接下來，我們重置文本區域，使其為空：

```
document.getElementById("input").value = '';
```

Then we send the question to our server so that the server can send it to the OpenAI API and send us back a response:

然後，我們將問題發送到我們的伺服器，以便伺服器可以將其發送到OpenAI API並將回應發送回給我們：

```
let res = await fetch('<http://localhost:5000/chat>',
  {
    method: 'POST',
    headers: {
      "Content-Type": 'application/json'
    },
    body: JSON.stringify({
      question: inputText
    })
  }
)

const data = await res.json()
```

If the response has a `message` property, we add the message content to the chat area in the UI:

如果回應有一個`message`屬性，我們將訊息內容添加到UI中的聊天區：

```
if (data.message) {
  const answer = document.createElement('div')
  answer.innerHTML = data.message
  answer.classList.add("box", "answer")
  parentDiv.appendChild(answer)
}
```

Now the frontend is all set. Let's move our focus back to the backend.

現在前端設置完成。讓我們把焦點移回後端。

How to Send the API Response to the Client

Our backend will serve as the middleman between the frontend and OpenAI's API. Basically, we'll get the prompt from the client, send it to the API, and send the response back to the client.

在我們的後端將充當前端和OpenAI API之間的中介。基本上，我們將從客戶端獲取提示，將其發送到API，然後將回應發送回客戶端。

In **server.js**, import these at the top of the file:

在**server.js**中，文件頂部引入這些：

```
const { Configuration, OpenAIApi } = require("openai")
```

Next, create an instance of the `openai` connection using the API key you generated earlier:

接下來，使用您之前生成的API金鑰創建`openai`連接的實例：

```
const openai = new OpenAIApi(new Configuration({
    // 用 ChatGPT 中生成的 API 金鑰替換 'your-api-key'
    apiKey: 'your-api-key'
}))
```

Finally, create the route:

最後，創建路由：

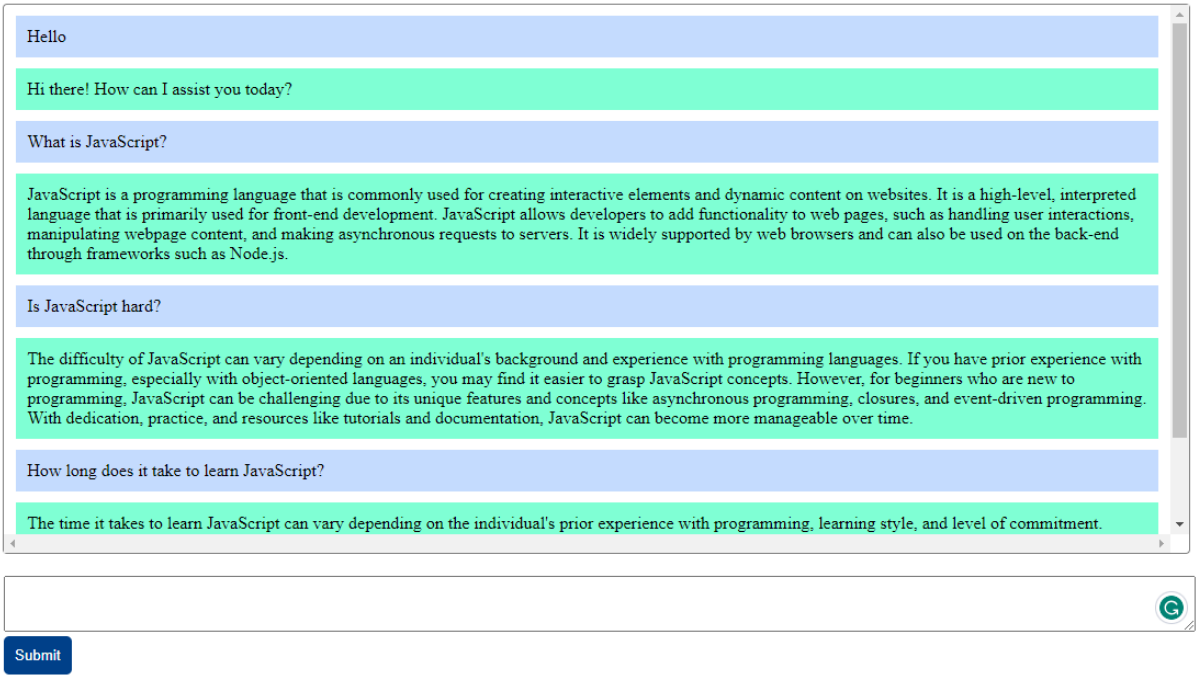
```
app.post('/chat', async (req, res)=> {
  try {
    const resp = await openai.createChatCompletion({
      model: "gpt-3.5-turbo",
        messages: [
          { role: "user", content: req.body.question}
        ]
    })

    res.status(200).json({message: resp.data.choices[0].message
  } catch(e) {
      res.status(400).json({message: e.message})
  }
})
```

Save the file changes, then go to your browser and submit a question. You should get back a response after a few seconds.

保存文件更改，然後轉到您的瀏覽器並提交一個問題。幾秒後，您應該會收到一個回應。

You can ask as many questions as you want, but you'll have to wait for a response to each question from the backend.

您可以提問任意多的問題，但您必須等待來自後端對每個問題的回應。



Screenshot of the questions and corresponding answers from ChatGPT

If any error is encountered, check the console in your browser to inspect the error message.

如果遇到任何錯誤，請檢查瀏覽器中的控制台以檢查錯誤訊息。

Conclusion

OpenAI's API offers you a way to include AI-powered chatbots in your application using JavaScript or even HTMX (if you're knowledgeable of HTML but not JavaScript).

Get my full-stack web development course if you're looking to build an awesome web application project from scratch and learn full stack web development.

OpenAI的API為您提供了一種使用JavaScript甚至HTMX（如果您熟悉HTML但不熟悉JavaScript）在應用程序中包含AI動力聊天機器人的方法。

如果您希望從頭開始構建一個令人驚嘆的Web應用程序項目並學習全棧Web開發，請參加我的全棧Web開發課程。