Complete the following JavaScript code by filling in the blanks with the appropriate code to initiate a chat with a GPT model:

```javascript
import OpenAI from "openai";

const client = new OpenAI ();

const response = await client. chat.completions .create({
    messages : [
      {
        role: "user",
        content: "Tell me how many seconds there are in a day in a Shakespearean tone?"
      }
    ],
    model: "gpt-3.5-turbo",
});

console.log(response.choices[0]. message .content);
```

👏 You got it!

What is the `system` message role used for when creating a prompt?

Providing the model with instructions, background information, or context on how to generate responses

👏 You got it!

Specifying the maximum token limit for the prompt

Specifying the penalty parameters for the model

Defining what model will be used in the chat.

Which of the following is **NOT** a message role type that can be used in a chat completion prompt?

observer

👏 Great job!

system

user

assistant

What is a few-shot prompt used for?

Creating a stream-like, chat-based interaction

Providing the model with a small amount of example data to guide its response

👏 Great job!

Initiating a conversation between multiple participants

To specify the model type to be used

What is the primary purpose of the `openai` package in JavaScript?

To automate data analysis and predictive modeling using traditional statistical methods.

To build and train new machine learning models from scratch within JavaScript applications.

To perform advanced image recognition and processing tasks using pre-built algorithms.

To offer developers a suite of tools for interacting with OpenAI's large language models, enabling the integration of generative AI into JavaScript applications.

👏 Correct!

In the following code, `savedMessages` is an array containing a series of `user` and `assistant` messages from a previous interaction with the OpenAI API. Which line of code would you use to add on the newly generated `assistant` message.

```
const response = await client.chat.completions.create({
  messages: savedMessages,
  model: "gpt-3.5-turbo",
});
```

savedMessages.push(response);

savedMessages.push(response);

savedMessages.push(messages);

savedMessages.push(response.choices[0].message.content);

👏 Correct!

savedMessages.push(response);