Alpha Release

ChatLib: ChatUI Javascript Library

LIBRARY OVERVIEW

ChatLib helps to create chat windows which can present forms and quizzes in a variety of unique ways, allowing websites to become more interactive and have more personality. This library provides an easy way for developers to integrate a chat bot/forms into their website and increase user interaction. What makes this library original from the others is the unique presentation style of questions and the variety of creative animations which have been provided for the developer to use.

- **USE CASE #1: Online Communities** can use this library to integrate a chatUI into their website to host polls or fan trivia questions much more easily, for their end-users to participate in.

- **USE CASE #1: Educational Websites** can use ChatLib to integrate lessons through chat-form using animations to highlight important parts and interactive quizzes can be added at the end of lessons,

- **USE CASE #1: Businesses** can use ChatLib to build their brand. The chatbot can act as a representative/mascot for the brand, helping customers with any frequently asked questions. Devs can also host surveys or receive feedback from customers through the chatUI .

More details for each of these use cases can be found in the ***ChatLib Proposal***.

ALPHA RELEASE DETAILS

An example of ChatLib being used can be found at : **https://quiet-dusk-02249.herokuapp.com/**. This website is an example of the first use case centered around online communities. The chatbot made with the library can be found on the lower right corner of the page.

As of now, the option for the developer to choose where to place the chatbot has been implemented however the ability to change the style and themes has not been pushed yet. Below more features will be discussed, followed up by future plans for further development.

Implemented Features

As of now ChatLib's alpha release includes the basis for all essential features for the library. These features include:

- **Chat Interface:** The library sets up the entire chat interface for the developer with one command. This window is customizable to fit the developer's needs and will hold all chat messages, questions and animations.
    - Minimize/ Expand: This chat interface can also be minimized and hidden away so the end-user is still able to see the main website the library is used for.

- **Text Input:** This allows the developer to prompt text input from the end-user and continue the conversation. It understands the key command 'Enter' as well as reacts to the send button.

- **Drag and Drop Questions:** This is a question format the developer can prompt the end-user to drag and drop an input to answer to a question.
    - This function also provides the foundation for other types of unique question formats such as Categorizing, Ordering/Rating and Voting.

- **Add Bot Chat Bubbles/Add User Chat Bubbles:** This allows the developer to create chat bubbles for both the bot and user in order to present the information on the chat window.

- **Set Bot Icon / Set User Icon :** This allows the developer to set the bot and user profile pictures as a way to further customize the chat window.

Future Features & Next Steps

The final product will include features such as animated reactions, animated timed or practice quizzes, presenting different types of questions such as: multiple choice, rating, ordering etc. in a variety of ways as well as providing new question formats such as categorization and matching. Each of these question types will allow the end-users to interact with the website through clicking, typing as well as drag and drop, making the website more interactive as a whole. More specific improvements and plans for future functions are listed below:

- **Chat Interface [*Improvements*]:**
    - Allow the developer to choose to have the chat window minimized or expanded when the site first loads

- **Text Input [*Improvements*]:**
    - Add a hide input function, if the developer wants to take away the input field at any time

- **Drag and Drop Questions [*Improvements*]:**
    - I chose to develop this feature first as it was one of the harder features to figure out as well as lays the foundation for other question formats
- **Ordering/Rating [*In Development*]:** Building off of the drag and drop function.
- **Multiple Choice [*In Development*]:** Also similar to the drag and drop function, will react to "on click" rather than the drop of the input.
- **Multi Select [*In Development*]:** Similar to both multiple choice and drag and drop as it will allow for multiple "on click" actions.
- **Matching Questions [*New*]:** This will be the next function that I plan to work on after finishing the multi select feature as they both share a similar base.
- **Reaction Animations [*New*]:** This I feel will be the next challenging part of developing this library as I have very little experience with animating with javascript. I plan to start developing these after finishing the main question format features.
- **Testing and Practice Modes :** I am unsure if I will be implementing this feature as it is something the developer can easily add on their end. It also does not need many DOM Manipulations when implemented. Instead I will try to implement more unique question formatting functions or animations.

LIBRARY STRUCTURE

Library Functions

Library functions the developer has access to in the Alpha Release:

setBotIcon/setUserIcon: These functions allow the user to set the bot's and user's profile picture shown in the chat.

createChatWindow: This initializes the window where the chatbot will reside. It has many optional variables which can allow the developer to customize the size, location and style of the window.

addBotMessage: This function allows the developer to add bot messages in the chat in order to provide end-users with information or prompt them to answer questions.

allowTextInput: This allows for the text input field to be shown and prompts the user to type their answer. When entered, it will take their input and add it to the chat window in the form of a chat bubble.

addDragDropQuestion: This allows for the developer to prompt a drag and drop question with variables to specify the number of categories and items to categorize, the categories and items themselves as well as.

addMultipleChoiceQuestion: This allows for the developer to prompt for a multiple choice question with variables to specify the number of choices, the choices themselves if there is a correct answer and if there is, which option it is.

addUserMessage: This function allows the developer to add user messages, however this will rarely be used as built-in questions have already accounted for the user's response. This is added in the library in the case they want to add their own type of question and need to create a userMessage to account for the response.

Storage Structure

Below is how the objects made with the library are stored and organized. I chose to store certain important divs to the library for ease of access as they are the most used (window, chatArea and inputField). I also stored the file paths for the bot and user icon's in strings to allow the developer to change them anytime (using the setIcon functions) and lastly I have three arrays, questions, botMessages and userMessages) which stores and keep track of the objects added in the chatArea.

```
this.window = null        // Assigned the window div which hold the
                          // entire chat UI

this.chatArea = null      // Assigned the conversation area div which
                          // holds all the chat bubbles shown

this.inputField = null    // Assigned the input field div

this.botIcon = ''         // file path to img src to bot icon file

this.userIcon = ''        // file path to img src to user icon file

this.questions = []       // array holding all questions asked asked
                          // and their corresponding answers

this.botMessages = []     // array holding all the the bubble message
                          // objects sent by the bot in the chat window

this.userMessages = []    // array holding all the bubble message
                          // objects sent by the user in the chat
                          // window
```

DOM Manipulation : An Example

One of the main objects that is manipulated throughout this library are the chat bubbles which are found in the chatArea div. The chat bubbles undergo manipulations to their size and style according to what the developer wants as well as when unique question formats are introduced to the chat window.

```
Bubble

{... ondragover = onDragOver(event)

ondrop = onDrop(event,nextFunction)

style = "border-radius: 15%;

background-color: #8fdddf; ...}
```

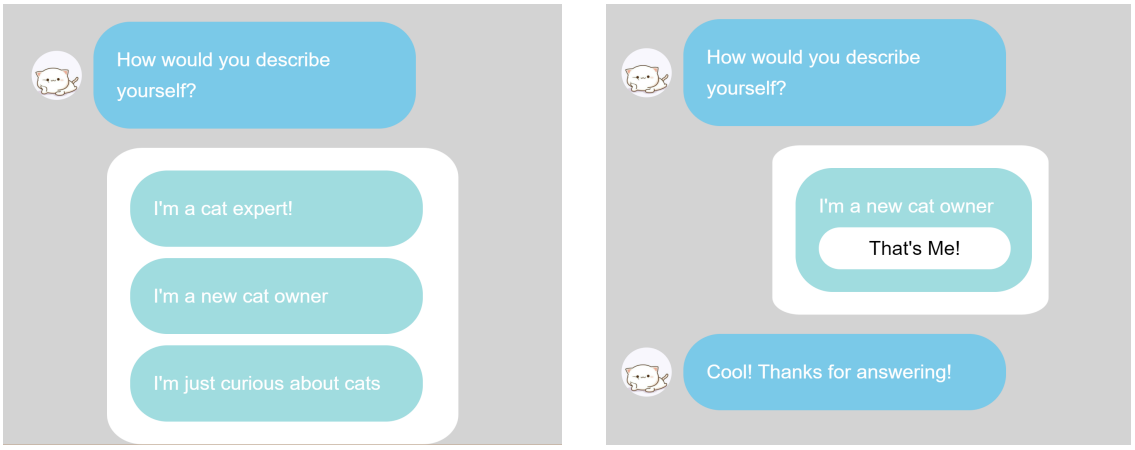
```
ChatArea {

// contains all the chat messages from both bot and user

// holds the drag and drop question prompt as well

}
```

For example, when the input is being dragged the input bubble changes colour. onDrop (when the input bubble is dropped into the selected div), it is removed from the previous inputArea div and then added to the new div within the chatArea, while it's text, size and colour are manipulated as well . Lastly, when the answer is submitted, the bubble containing the drag drop options ("im a cat expert", "Im a new cat owner" etc) is emptied except for the selected answer. This div is then moved to the user's side of the chat to indicate that they have submitted their choice. We see there are many DOM manipulations to the input and bubbles in the chatArea through in this one feature and a picture of this example occurring is shown below.



CHALLENGES

While making this library I came across a few challenges and have solved many issues. However, there are still a few issues I am working on which will be my next steps for this project.

As of now I am unable to call my library functions within other library functions, causing me to write duplicate code to do the same thing. Due to this, when trying to implement callback function parameters for each library function, the functions do not run. In order to solve this issue, I have used string parsing (which is why a large portion of the code is a string in example.js) and calling helper functions but it is very inefficient. I have implemented these temporary solutions in order to showcase features for the alpha release however I will be fixing them before implementing any features further.