Q-and-A

Typesetting Q&A-style conversation made easier

Jinwen XU

November 2023, in Paris

Corresponding to: Q-and-A 2023/11/11.

/ 1 / Introduction

Q: What is this?

A: Q-and-A is a LaTeX document class for you to typeset Q&A-style conversation. It turns a simple pure text Q&A dialog like this:

code

```
Q:
  What is `...`?

A:
  A `...` is a `...`. It has the following features:

  [Feature A]
    [*Aspect one] Firstly, ...
    [*Aspect two] Secondly, ...

  [Feature B]
    It also ...

Q:
  Can you provide some examples for `...`?

A:
  Certainly. There are many types of `...`, for example:
    [*] *emphasized*
    [*] **bold**
    [*] ***bold and emphasized***
```

into a carefully designed document like this:

> **Q1:** What is `...`?
>
> **A1:** A `...` is a `...`. It has the following features:
> 1. **Feature A:**
>   - **Aspect one:** Firstly, ...
>   - **Aspect two:** Secondly, ...
> 2. **Feature B:** It also ...
>
> **Q2:** Can you provide some examples for `...`?
>
> **A2:** Certainly. There are many types of `...`, for example:
> - *emphasized*
> - **bold**
> - ***bold and emphasized***

/ 2 / Preparation

Q: That is nice. How can I use it? Is there anything that needs to be prepared?

A: You should make sure that this document class is properly installed.

If you are using TeX Live 2024 or newer, or the most recent version of MikTeX, then this package should already be included, and you don't need to do anything.

Otherwise, you need to check for package update to see if you can receive it. In case not, you can always go to the CTAN page to download the `.zip` file with all related files included.

/ 3 / Usage

Q: Now that I have successfully installed it, could you propose an example of usage?

A: Of course. A typical document looks like this:

latex

```latex
\documentclass[%11pt,
  use theme = ⟨theme⟩,
  numbered question,
  numbered answer,
  % answer outside the box,
  scroll,
]{Q-and-A}

\UseLanguage{⟨language⟩} % For English this line can be omitted.

\begin{document}


% The content


\end{document}
```

The available class options include:

- Font size options such as `11pt`, `12pt`.
- `scroll`: turn the scroll mode on, which generates a single-page pdf similar to a long screenshot. It is recommended to use this option if your document contains some large piece of code.
- `use theme = ⟨theme⟩`: use the selected theme, available choices include: `default` (like the current document), `ChatGPT-light` and `ChatGPT-dark` (see the demo documents).
- `numbered question`, or `question number`: enable the numbering of questions.
- `numbered answer`, or `answer number`: enable the numbering of answers.
- `answer outside the box`, or `answer out of the box` (or even omit `the`): enable another implementation for printing the answer that does not involve putting it into a colored box. This would be useful if the answers contain many long pieces of code while you still want the pages to break normally.

Q: What about the main content?

A: You have already seen an example of the main content. As you might have noticed, there are several syntaxes. Let me explain.

1. **Questions (Q), Answers (A), and Narrations (N):**
   - A question begins with the prefix `Q:` or `?`.
   - An answer begins with the prefix `A:` or `:`.
   - A narration begins with the prefix `N:` or `"`.

   > Note that this depends on the current language. The prefixes `?`, `:` and `"` being universal, yet —
   > - for French, it is Q&R&N, thus the alphabetical prefixes become `Q:`, `R:` and `N:`;
   > - for German, it is F&A&E, thus the alphabetical prefixes become `F:`, `A:` and `E:`;
   > - for Italian, it is D&R&N, thus the alphabetical prefixes become `D:`, `R:` and `N:`;
   > - for Portuguese and Brazilian, it is P&R&N, thus the alphabetical prefixes become `P:`, `R:` and `N:`;
   > - for Russian, it is B&O&P, thus the alphabetical prefixes become `B:`, `O:` and `P:`;
   > - for Spanish, it is P&R&N, thus the alphabetical prefixes become `P:`, `R:` and `N:`;
   > - for simplified Chinese, it is also possible to use the prefix `问:` for questions, `答:` for answers, and `注:` for narrations; similarly for traditional Chinese;
   > - for Chinese or Japanese, it is also possible to use the prefix `？` for questions, `：` for answers, and `“`, `”` or `「` for narrations.
2. **Emphasize and Bold:**
   - Use `*⟨text⟩*` to emphasis `⟨text⟩`.
   - Use `**⟨text⟩**` to make `⟨text⟩` into boldface.
   - Use `***⟨text⟩***` to combine the previous effects.
3. **Enumerate and Itemize:**
   - **Emphasized version:**
     - An emphasized `enumerate` is marked by `[⟨text⟩]` *at the beginning*, where `⟨text⟩` is the text to be emphasized. The numbering is reset at the end of each answer.
       > If you wish to restart the numbering from 1, write an asterisk after the final bracket: `[⟨text⟩]*`.
     - An emphasized `itemize` is marked by `[*⟨text⟩]`, where `⟨text⟩` is the text to be emphasized.
   - **Normal version:**
     - When `⟨text⟩` is empty, you get the usual appearance of `enumerate` or `itemize`.

   > With this method of writing, each emphasized `enumerate` and `itemize` must be put into a separate paragraph, and the lists cannot be nested. The usual `enumerate` and `itemize` environments would still be useful if you need nested lists.
4. **Images:** Displayed images can be included via one of the following ways:
   - || <⟨width⟩> {⟨image file name⟩} or || {⟨image file name⟩} <⟨width⟩>: display in the center.
   - ((<⟨width⟩> {⟨image file name⟩} or (({⟨image file name⟩} <⟨width⟩>: display on the left.
   -)) <⟨width⟩> {⟨image file name⟩} or)) {⟨image file name⟩} <⟨width⟩>: display on the right.

   > The `⟨width⟩` is optional. Here, `⟨width⟩` is a number like `0.75`, the unit is `\linewidth`. When `⟨width⟩` is not given, the width would be full `\linewidth`.

   - For centered image, the caption can be written via `>> ⟨caption text⟩`.

   > Currently, the `⟨caption text⟩` shall be directly printed in the center. You may wish to add a pair of parentheses around it.
5. **Code:** Due to the current implementation of this document class, it is unfortunate that you cannot directly insert source code in your document. There are some workarounds, though.
   - For *inline* code, you may simply write it between two backticks `` `⟨code⟩` ``, similar to the Markdown syntax. However, be aware that special characters need to be escaped, for example, `\` should be written as `\textbackslash`, `{` should be written as `\{`, `%` should be written as `\%`, etc.
   - For *displayed* code, stored the code into a separate file, and then use `== {⟨file name⟩}` to print it. You may also use an optional argument like `== [⟨language⟩] {⟨file name⟩}` to select the language of your code.
     > If the answers contain many long pieces of code, you may consider using the class option `answer outside the box`.
6. **Sections:**
   - You may start a new (*unnumbered*) —
     - section, via `## {⟨section title⟩}`;
     - subsection, via `### {⟨subsection title⟩}`;
     - subsubsection, via `#### {⟨subsubsection title⟩}`;
   - If you wish to use the *numbered* version, write `##+`, `###+` and `####+` instead.
7. **Input/Include Files:**
   - Use `:: {⟨file name⟩}` to input a file.
   - Use `::: {⟨file name⟩}` to include a file.

And don't forget that you are still using LaTeX, so images, tables and lists can be written as usual.

> Currently, the default style for showing tables has not been settled (since the color configuration for tables with or without headers would be slightly different). As a temporary solution, it is recommended for you to consult the several nice answers of this question: https://tex.stackexchange.com/q/10224.

/ 4 / Cautions

Q: I see. Is there anything else for me to be careful about?

A: Glad that you asked. Here are several things that should be taken care of:

- A question, answer or narration should always begin in a new paragraph.
- An emphasized `enumerate` should also begin in a new paragraph. An emphasized `itemize` should usually begin in a new paragraph, with the exception that it can (partially) be used within an `enumerate`.
- Likewise, a `section` / `subsection` / `subsubsection` should be placed in a separate paragraph.
- Input or inclusion of files should also be operated in a separate paragraph.
- For emphasizing and bolding the text, it would be necessary to separate the asterisks with `{}` in some special cases: `**like**{}***this***{}*one*`.
- Due to the recognition of the syntax, you cannot directly use characters like `*` as usual, especially in math mode. You should consider using an alternative macro, such as `\ast` for `*` in math mode, or define a command version of the character in the preamble, such as `\textasterisk` for `*`. A list of predefined commands are:
  - `\textasterisk` for `*`;
  - `\textbacktick` for `` ` ``;
  - `\textcolon` for `:`;
  - `\textequalsign` for `=`;
  - `\textleftparen` for `(`;
  - `\textrightparen` for `)`;
  - `\textsharp` for `#`.
  - `\textvert` for `|`.

  There are also a few that already provided by LaTeX, such as:
  - `\textbackslash` for `\`;
  - `\textgreater` for `>`;
  - `\textless` for `<`.

/ 5 / Customization

Q: Now that we have learned the basic usage, I would like to know more. Can I customize the interface to suit my preferences, for instance?

A: Certainly. Apart from using themes via class option, you may also change the identifiers and the labels for each role in the conversation.

1. **Changing itentifiers:** Instead of the default identifiers, such as `?` for questions, you may also use your preferred one. This can be done via the use of `\QASetTypePrefix` in the preamble of your document.
   - Use `\QASetTypePrefix{Q}{⟨identifiers⟩}` to set the identifiers for questions.
   - Use `\QASetTypePrefix{A}{⟨identifiers⟩}` to set the identifiers for answers.
   - Use `\QASetTypePrefix{N}{⟨identifiers⟩}` to set the identifiers for narrations.

   Here, ⟨*identifier*⟩ is a comma list of your specified identifiers. For example, the default identifier for narrations is preset via `\QASetTypePrefix{N}{N:,", “,” , 「}`.

   > Note that, due to its implementation, the identifier cannot contain comma `,` in it. If you wish to use an identifier that contains a comma, you may use `\QAAddTypePrefix` instead, which only adds *one* identifier per use.

   > Note also that, the identifiers are *reset* upon changing of language. Thus, you need to put your setting into the corresponding language configuration, for example, via `\AddLanguageSetting[⟨language name⟩]{⟨settings⟩}`
2. **Changing labels:** You may also change the labels. For example, from the text "Q:" to a logo icon. This can be done via the use of `\SetLogoCode` in the preamble of your document.
   - Use `\SetLogoCode{Q}{⟨logo code⟩}` to set the labels for questions.
   - Use `\SetLogoCode{A}{⟨logo code⟩}` to set the labels for answers.

   Here, ⟨*logo code*⟩ is the actual code for displaying the corresponding label. For example, the default label for (unnumbered) questions is preset via `\SetLogoCode{Q}{\textbf{Q\,:}}`.

   > In the demo document, you can find an example on how to use this command to specify a logo for each role in the conversation.

   If you wish to include the question number or answer number, you may use the command `\QAShowCounter`:
   - Use `\QAShowCounter{Q}` to step the counter of question and print it.
   - Use `\QAShowCounter{A}` to step the counter of answer and print it.

   For example, the default label for numbered questions is preset by adding `\QAShowCounter{Q}` to the unnumbered version: `\SetLogoCode{Q}{\textbf{Q\QAShowCounter{Q}\,:}}`.

/ 6 / Known Issues

Q: Is there any known issue with this document class?

A: Unfortunately, yes `:(`.

Below is a list of known issues:

- Currently, the code highlight is done by the package `listings`. Due to its own limitations, the result is still far from satisfactory. Using `minted` instead could improve the situation, but this would require `-shell-escape` and some external tweaking, thus it would still take some effort to make it work with the current document class.
- Due to the current implementation, one cannot direct write code (either inline or displayed) within the text (alternative solution has been proposed above).
- Due to the current implementation, when displaying the code, one may not be able to write some of the language name like `C#` directly, but has to write an alternative name such as `csharp`.
- Due to the current implementation, one may not be able to directly write certain characters (such as `*`) in the text, this might be especially inconvenient in math mode (temporary alternative solution has been proposed above).
- Due to the current implementation, the method for inputting and including subfiles doesn't really work like `\input` and `\include`. In particular, the current type of conversation would be reset, this might cause some confusion in certain cases.
- Due to the current implementation, though it is already possible to automatically adopt the identifiers and labels for supported languages, you still need to use the identifiers `Q`, `A` and `N` when setting them.
- Currently, for typesetting questions and answers, the text is in fact being put into some sort of `description` list (upon reflection, why did I do this? I could have simply put the label into margin...). As a result, the level of lists might be slightly messed up, which could sometimes lead to issues with the list labels.

/ 7 / Get Support

Q: What should I do if I encounter any problem?

A: If you run into any issues or have ideas for improvement, feel free to discuss on:

https://github.com/Jinwen-XU/Q-and-A/issues

or email me via ProjLib@outlook.com.