

Markdown themes in practice

Vít Starý Novotný

Abstract

The Markdown package for TEX supports themes that allow TEX ncians to tailor the presentation of Markdown and YAML content on the page. In this article, I will show the current state of Markdown themes using the example of $\text{L}\text{A}\text{T}\text{E}\text{X}$ templates that I developed for the International Software Testing Qualifications Board (ISTQB). Readers will leave with actionable steps to create or modify Markdown themes for $\text{L}\text{A}\text{T}\text{E}\text{X}$, and insights into extending these principles to other TEX engines.

Introduction

Although TEX has beautiful output, its input macro language is an acquired taste for many authors. The Markdown package for TEX allows authors to type familiar Markdown and YAML directly into a TEX document and receive a similarly beautiful output.

In my previous article, I introduced Markdown themes [2]. Much like CSS stylesheets, Markdown themes allow TEX ncians to tailor the presentation of Markdown and YAML content without complicating the document markup for authors. While that article used simple examples to explain the basic concepts behind Markdown themes, it did not demonstrate their application on a larger scale in real-world software projects.

In July 2023, I began working with the International Software Testing Qualifications Board (ISTQB) to help them typeset their certification study materials from Markdown and YAML sources. In this article, I discuss my work as a case study of using the Markdown package in a real-world software project.

Project overview

In my work, I developed a $\text{L}\text{A}\text{T}\text{E}\text{X}$ document class and six Markdown themes [1].

The $\text{L}\text{A}\text{T}\text{E}\text{X}$ document class is named `istqb` and it is stored in file `template/istqb.cls`. It implements the design of all ISTQB documents, defines the meaning of common Unicode characters, and defines $\text{L}\text{A}\text{T}\text{E}\text{X}$ markup such as `\istqbunnumberedsection`, `\istqblandscapebegin`, and `\istqblandscapeend`.

The Markdown themes are named `istqb/*` and stored in files `template/markdowntheme*.tex` and `*.sty`, see also Figure 1. Here is what they do:

- The theme `istqb/common` enables Markdown syntax extensions, implements the loading of YAML language definitions and document metadata into TEX macros, and defines the mapping

between Markdown elements and $\text{L}\text{A}\text{T}\text{E}\text{X}$ markup. The remaining themes are based on this theme and they implement support for specific types of ISTQB documents.

- The `istqb/body-of-knowledge` and `syllabus` themes are used in ISTQB Body of Knowledge and Syllabus documents. At the time of writing, the themes implement no extra functionality.
- The theme `istqb/sample-exam` implements the loading of YAML question definitions into TEX macros in ISTQB Sample Exam Questions and Answers documents. The following two themes are based on this theme.
- The theme `istqb/sample-exam/questions` implements the typesetting of questions in ISTQB Sample Exam Questions documents.
- The theme `istqb/sample-exam/answers` implements typesetting of answer keys and answers in ISTQB Sample Exam Answers documents.

In the rest of this article, I show the main concepts behind Markdown themes on the example of ISTQB Sample Exam Questions and Answers documents that use the themes `istqb/sample-exam/questions` and `/answers`.



With Markdown themes, your document can wear many different disguises, just like the wolf.

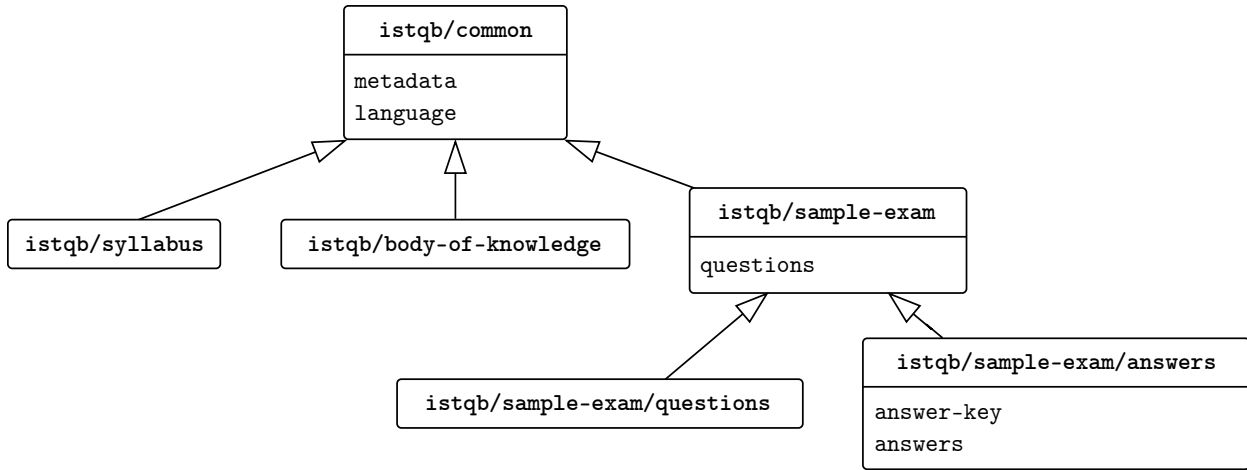


Figure 1: A class diagram of the six Markdown themes that I developed for the International Software Testing Qualifications Board (ISTQB). The snippets `metadata`, `language`, `questions`, `answer-key`, and `answers` specify the public interface of the themes and arrows specify inheritance.

Questions

Question #1 (1 Point)

What is the answer to life, the universe, and everything?

a) 24
b) 42
c) 64
d) 84

Select ONE option.

Question #5 (1 Point)

What's France's capital?

a) Berlin
b) Madrid
c) Paris
d) Rome

Select ONE option.

Question #6 (2 Points)

Which two of the following animals are classified as mammals?

a) Shark
b) Dolphin
c) Eagle
d) Whale
e) Crocodile

Select TWO options.

a

Answer key

Question Number (#)	Correct Answer	Learning Objective (LO)	K-Level	Number of Points
1	b	EXMPL1.2.3	K1	1
5	c	EXMPL4.5.6	K2	1
6	b, d	EXMPL7.8.9	K3	2

b

Answers

Question Number (#)	Correct Answer	Explanation / Rationale	Learning Objective (LO)	K-Level	Number of Points
1	b	The answer to life, the universe, and everything is a concept from Douglas Adams' science fiction series "The Hitchhiker's Guide to the Galaxy", where the supercomputer Deep Thought gives the answer 42.	EXMPL-1.2.3	K1	1
5	c	The capital of France is Paris, known for art, fashion, and culture.	EXMPL-4.5.6	K2	1
6	b, d	Dolphins and whales are classified as mammals because they are warm-blooded, breathe with lungs, and feed their young milk.	EXMPL-7.8.9	K3	2

c

Figure 2: Three different ways to typeset question definitions in ISTQB Sample Exam Questions and Answers documents: a) a list of questions, b) an answer key, and c) a list of answers.

1 Question definitions

As an example of question definitions, I use the following YAML file named `questions.yml`:

```
num-questions: 3
max-score: 4
pass-score: 50 # percent
duration: [10, 15] # minutes
questions:
  1:
    learning-objective: 1.2.3
    k-level: K1
    number-of-points: 1
    question: >
      What is the answer to life,
      the universe, and everything?
    answers: {a: 24, b: 42, c: 64, d: 84}
    correct: b
    explanation: >
      The answer to life, the universe,
      and everything is a concept from
      Douglas Adams' science fiction
      series "The Hitchhiker's Guide to
      the Galaxy", where the supercomputer
      Deep Thought gives the answer 42.
  5:
    learning-objective: 4.5.6
    k-level: K2
    number-of-points: 1
    question: What's France's capital?
    answers: {a: Berlin, b: Madrid,
              c: Paris, d: Rome}
    correct: c
    explanation: >
      The capital of France is Paris,
      known for art, fashion, and culture.
  6:
    learning-objective: 7.8.9
    k-level: K3
    number-of-points: 2
    question: >
      Which two of the following animals
      are classified as mammals?
    answers: {a: Shark, b: Dolphin,
              c: Eagle, d: Whale,
              e: Crocodile}
    correct: [b, d]
    explanation: >
      Dolphins and whales are classified
      as mammals because they are
      warm-blooded, breathe with lungs,
      and feed their young milk.
```

The file specifies three questions. For each question, it provides up to five possible answers.

2 User interface

In this section, I show how we can use themes `istqb/sample-exam/questions`, and `/answers` to typeset the question definitions from the previous section.

2.1 Typesetting questions

As an example of an ISTQB Sample Exam Questions document, I use the following \LaTeX file:

```
\documentclass{istqb}
\usepackage{markdown}
\markdownSetup {
  import = {
    istqb/sample-exam/questions =
    questions as qst
  }
}
\begin{document}
\istqbunnumberedsection{Questions}
\markdownInput[snippet=qst]{questions.yml}
\end{document}
```

The file imports the snippet `questions` from theme `istqb/sample-exam/questions` and uses it to:

1. Process question definitions in `questions.yml`.
2. Typeset the list of questions shown in Figure 2a.

2.2 Typesetting answer key and answers

As an example of an ISTQB Sample Exam Answers document, I use the following \LaTeX file:

```
\documentclass{istqb}
\usepackage{markdown}
\markdownSetup {
  import = {
    istqb/sample-exam/answers = {
      answer-key as key,
      answers as ans,
    },
  }
}
\begin{document}
\istqblandscapebegin
\istqbunnumberedsection{Answer key}
\markdownInput[snippet=key]{questions.yml}
\istqbunnumberedsection{Answers}
\markdownInput[snippet=ans]{questions.yml}
\istqblandscapeend
\end{document}
```

The file imports the snippet `answers` from theme `istqb/sample-exam/answers` and uses it to:

1. Process question definitions in `questions.yml`.
2. Typeset the answer key shown in Figure 2b.
3. Typeset the list of answers shown in Figure 2c.

3 Implementation

In this section, I show the implementation of ISTQB Sample Exam Questions and Answers documents. To make programming easier, I use the high-level `expl3` language in addition to plain `TeX` and `LATEX 2ε`.

3.1 Processing question definitions

Both the snippet questions from the theme `istqb/sample-exam/questions` and the snippet answers from the theme `/answers` process question definitions before typesetting them. For the processing, they use the snippet questions from the theme `istqb/sample-exam`, which I describe in this section.

First, I define a key-value `istqb/questions`:

```

1 \keys_define:nn
2 { istqb / questions }
3 { num-questions .int_gset:N =
4   \g_istqb_num_questions_int,
5   max-score .int_gset:N =
6   \g_istqb_max_score_int,
7   pass-score .int_gset:N =
8   \g_istqb_pass_score_int }
```

The key-value stores the values in top-level unstructured fields `num-questions`, `max-score`, and `pass-score` from question definitions to variables.

Next, I define a key-value `istqb/questions/duration`:

```

9 \keys_define:nn
10 { istqb / questions / duration }
11 { 1 .int_gset:N =
12   \g_istqb_duration_min_int,
13   2 .int_gset:N =
14   \g_istqb_duration_max_int }
```

The key-value stores the values in the top-level structured field `duration` to variables.

Then, I define the snippet `questions` itself:

```

15 \seq_new:N \g_istqb_questions_seq
16 \markdownSetupSnippet
17 { questions }
18 { jekyllData,
19   expectJekyllData,
20   renderers = {
21     jekyllDataBegin = {
22       \seq_gclear:N
23       \g_istqb_questions_seq },
24     jekyllData(String|Number) = {
25       \keys_set:nn
26       { istqb / questions }
27       { { #1 } = { #2 } }},
28     jekyllDataMappingBegin = ,
29     jekyllDataSequenceBegin = {
30       \str_case:nn
31       { #1 }
32       { { duration } {
33         \markdownSetup
```

```

        { code = \group_begin:,
          renderers = {
            jekyllData(String
              |Number) = {
              \keys_set:nn
                { istqb / questions /
                  duration }
                {{ #1 } = { #2 } }},
            jekyllDataSequenceEnd =
              \group_end: }}}}},
jekyllData(Mapping|Sequence)Begin += {
  \str_case:nn
    { #1 }
    { { questions } {
      \markdownSetup
        { code = \group_begin:,
          renderers = {
            jekyllData(Mapping
              |Sequence)End =
            },
            snippet = istqb
              / sample-exam / questions
              / list,
            renderers = {
              jekyllData(Mapping
                |Sequence)End
                += \group_end: }}}}}}
```

The snippet processes question definitions as follows:

1. Define an empty sequence that will store question numbers.
2. Pass unstructured top-level fields to the key-value `istqb/questions`.
3. Pass the structured top-level field `duration` to the key-value `istqb/questions/duration`.
4. Pass the structured top-level field `questions` to a snippet `questions/list`.

Next, I define the snippet `questions/list`:

```

49 \markdownSetupSnippet
50 { questions / list }
51 { renderers = {
52   jekyllDataMappingBegin = {
53     \group_begin:
54     \tl_set:Nn
55     \l_istqb_current_question_tl
56     { #1 }
57     \seq_gput_right:Nv
58     \g_istqb_questions_seq
59     \l_istqb_current_question_tl
60     \markdownSetup
61     { renderers = {
62       jekyllDataMappingEnd = },
63       snippet = istqb / sample-exam
64       / questions / *,
65       renderers = {
66         jekyllDataMappingEnd +=
67         \group_end: }}}}}}
```

The snippet processes each question as follows:

1. Store the current question number.
2. Pass all fields to a snippet `questions/*`.

Then, I define key-value `istqb/questions/*`:

```

80 \prop_new:N
81   \g_istqb_question_number_of_points_prop
82 \prop_new:N
83   \g_istqb_question_learning_objective_prop
84 \prop_new:N
85   \g_istqb_question_k_level_prop
86 \prop_new:N
87   \g_istqb_question_explanation_prop
88 \prop_new:N
89   \g_istqb_question_text_prop
90 \keys_define:nn
91   { istqb / questions / * }
92   { number-of-points .code:n = {
93     \prop_gput:cVn
94     { g_istqb_question_number_of_points
95       _prop }
96     \l_istqb_current_question_tl
97     { #1 } },
98   learning-objective .code:n = {
99     \prop_gput:cVn
100    { g_istqb_question_learning_objective
101      _prop }
102    \l_istqb_current_question_tl
103    { #1 } },
104   k-level .code:n = {
105     \prop_gput:NVn
106     \g_istqb_question_k_level_prop
107     \l_istqb_current_question_tl
108     { #1 } },
109   explanation .code:n = {
110     \prop_gput:NVn
111     \g_istqb_question_explanation_prop
112     \l_istqb_current_question_tl
113     { #1 } },
114   question .code:n = {
115     \prop_gput:NVn
116     \g_istqb_question_text_prop
117     \l_istqb_current_question_tl
118     { #1 } }}

```

The key-value stores the values in unstructured fields `number-of-points`, `learning-objective`, `k-level`, `explanation`, and `question` to dicts. The dicts use the current question number as the key.

Next, I define the snippet `questions/*`:

```

119 \markdownSetupSnippet
120 { questions / * }
121 { renderers = {
122   jekyllData(String|Number) = {
123     \keys_set:nn
124     { istqb / questions / * }
125     { { #1 } = { #2 } }},
126   jekyllDataSequenceBegin = {
127     \str_case:nn

```

```

{ #1 }
{ { correct } {
  \markdownSetup
  { code = \group_begin:,
    renderers = {
      jekyllDataSequenceEnd =
    },
    snippet = istqb
      / sample-exam / questions
      / * / correct,
    renderers = {
      jekyllDataSequenceEnd +=
        \group_end: }}}}},
jekyllDataMappingBegin = {
  \str_case:nn
  { #1 }
  { { answers } {
    \markdownSetup
    { code = \group_begin:,
      renderers = {
        jekyllDataMappingEnd = },
        snippet = istqb
          / sample-exam / questions
          / * / answers,
        renderers = {
          jekyllDataMappingEnd +=
            \group_end: }}}} }}

```

The snippet processes question definitions as follows:

1. Pass unstructured fields to the key-value `istqb/questions/*`.
2. Pass the structured field `correct` to a snippet `questions*/correct`.
3. Pass the structured field `answers` to a snippet `questions*/answers`.

Notice the design pattern on lines 44–60, 64–79, and 126–154 that locally applies a `<snippet>` to an `<element>`.¹ This pattern redefines the renderer `<element>Begin`, which is placed to the output when the `<element>` starts, as follows:

1. Open a \TeX group and apply the `<snippet>`.
2. Redefine the renderer `<element>End`, which is placed to the output when the `<element>` ends, so that it closes the \TeX group.

Finally, I define snippets `questions*/answers` and `/correct`:

```

155 \prop_new:N \g_istqb_answer_keys_prop
156 \prop_new:N \g_istqb_answers_prop
157 \seq_new:N \l_istqb_current_answer_keys_seq
158 \clist_new:N
159   \l_istqb_current_answer_keys_clist

```

¹ Such design patterns can be repetitive and difficult to understand without additional comments in the code. Markdown Enhancement Proposal (MEP) 445 [3] envisions support for higher-order snippets that would make it possible to hide such design patterns behind easy-to-read shorthands.

```

160 \markdownSetupSnippet
161 { questions / * / answers }
162 { renderers = {
163   jekyllData(String|Number) = {
164     \seq_put_right:Nn
165     \l_istqb_current_answer_keys_seq
166     { #1 }
167     \tl_set:NV
168     \l_tmpa_tl
169     \l_istqb_current_question_tl
170     \tl_put_right:Nn
171     \l_tmpa_tl
172     { / #1 }
173     \prop_gput:NVn
174     \g_istqb_answers_prop
175     \l_tmpa_tl
176     { #2 } },
177   jekyllDataMappingEnd += {
178     \clist_set_from_seq:NN
179     \l_istqb_current_answer_keys_clist
180     \l_istqb_current_answer_keys_seq
181     \prop_gput:NVv
182     \g_istqb_answer_keys_prop
183     \l_istqb_current_question_tl
184     { l_istqb_current_answer_keys
185       _clist } }}}
186 \prop_new:N
187 \g_istqb_answer_correct_keys_prop
188 \seq_new:N
189 \l_istqb_current_answer_correct_keys_seq
190 \clist_new:N
191 \l_istqb_current_answer_correct_keys_clist
192 \markdownSetupSnippet
193 { questions / * / correct }
194 { renderers = {
195   jekyllData(String|Number) = {
196     \seq_put_right:cn
197     { l_istqb_current_answer_correct
198       _keys_seq }
199     { #2 } },
200   jekyllDataSequenceEnd += {
201     \clist_set_from_seq:cc
202     { l_istqb_current_answer_correct
203       _keys_clist }
204     { l_istqb_current_answer_correct
205       _keys_seq }
206     \prop_gput:NVv
207     \g_istqb_answer_correct_keys_prop
208     \l_istqb_current_question_tl
209     { l_istqb_current_answer_correct
210       _keys_clist } }}}

```

The snippets accumulate potential and correct answer letters in a sequence, respectively. Then, they store the sequence as a comma-list to a dict that uses the current question number as the key.

Moreover, the snippet `questions/* / answers` stores potential answer texts to a dict that uses $\langle current\ question\ number \rangle / \langle answer\ letter \rangle$ as key.

Notice that I used no format-specific code in this section. Therefore, I can use the theme `istqb/sample-exam` with any format that supports `expl3` such as plain `TEX` and `ConTEXt`, not just with `LATEX`.

3.2 Typesetting questions

In this section, I describe the snippet questions from theme `istqb/sample-exam/questions`. This snippet typesets the list of questions in Figure 2a.

First, I import the theme `istqb/sample-exam` and I use the snippet `questions` from this theme to process question definitions:

```

1 \markdownSetup
2 { import = istqb / sample-exam }
3 \markdownSetupSnippet
4 { questions }
5 { snippet = istqb / sample-exam
6   / questions,

```

After the question definitions have been processed, I iterate over all question numbers. For each question number, I define a variable with code that typesets the corresponding question:

```

7   renderers = {
8     jekyllDataEnd = {
9       \seq_map_inline:Nn
10        \g_istqb_questions_seq
11        { \tl_set:Nn
12          \l_istqb_question_tl
13          {
14            \tl_set:Nn
15              \l_tmpa_tl
16              { Question~\# ##1~( }
17            \prop_get:cnN
18              { g_istqb_question_number
19                _of_points_prop }
20              { ##1 }
21            \l_tmpb_tl
22            \tl_put_right:NV
23              \l_tmpa_tl
24              \l_tmpb_tl
25            \tl_put_right:Nn
26              \l_tmpa_tl
27              { ~Point }
28            \int_compare:VNnF
29              \l_tmpb_tl = { 1 }
30            { \tl_put_right:Nn
31              \l_tmpa_tl
32              { s } }
33            \tl_put_right:Nn
34              \l_tmpa_tl
35              { ) }
36            \exp_args:NNV
37              \subsection *
38              \l_tmpa_tl
39            \exp_args:NVV

```

First, I add a section heading for the question:

```

40         \markboth
41         \l_tmpa_tl
42         \l_tmpa_tl
43     \exp_args:NnnV
44     \addcontentsline
45     { toc }
46     { subsection }
47     \l_tmpa_tl

```

Next, I add the question text and potential answers:

```

48     \prop_item:Nn
49     \g_istqb_question_text_prop
50     { ##1 }
51     \prop_get:NnN
52     \g_istqb_answer_keys_prop
53     { ##1 }
54     \l_tmpa_clist
55     \begin { enumerate }
56     \clist_map_inline:Nn
57     \l_tmpa_clist
58     { \item [ #####1 ) }
59         \prop_item:Nn
60         \g_istqb_answers_prop
61         { ##1 / #####1 } }
62     \end { enumerate }
63     \medskip

```

Lastly, I add the text “Select *⟨number of correct answers⟩* option(s).”:

```

64     \prop_get:cnN
65     { g_istqb_answer_correct
66       _keys_prop }
67     { ##1 }
68     \l_tmpa_clist
69     \int_set:Nn
70     \l_tmpa_int
71     { \clist_count:N
72       \l_tmpa_clist }
73     Select~\int_case:nn
74     { \l_tmpa_int }
75     { { 1 } { ONE~option }
76       { 2 } { TWO~options } }
77     }

```

Finally, I typeset the code from the variable at natural height and store the result to a vertical box:

```

78     \vbox_set:NV
79     \l_tmpa_box
80     \l_istqb_question_tl

```

For short questions, I insert the box to the current list for typesetting to prevent page breaks within the question. For longer questions, I place the content of the variable to the input stream, so that page breaks can occur naturally:

```

81     \dim_compare:nNnTF
82     { \box_ht:N \l_tmpa_box }
83     >
84     { 0.3 \paperheight }
85     { \tl_use:N

```

```

86         \l_istqb_question_tl }
87     { \box_use:N \l_tmpa_box }
88     \par }}}}

```

3.3 Typesetting answer key

In this section, I describe the snippet `answer-key` from the theme `istqb/sample-exam/answers`. This snippet typesets the answer key in Figure 2b.

First, I load packages `multicol` and `supertabular`:

```

1  \RequirePackage { multicol }
2  \RequirePackage { supertabular }
3  \RequirePackage { array }
4  \newcolumntype
5  { C }
6  [ 1 ]
7  { >{ \centering\arraybackslash } p { #1 } }

```

The packages allow me to typeset the answer key as a table in a two-column layout that automatically inserts column breaks.

Next, I import the theme `istqb/sample-exam` and I use the snippet `questions` from this theme to process question definitions:

```

8  \markdownSetup
9  { import = istqb / sample-exam }
10 \markdownSetupSnippet
11 { answer-key }
12 { snippet = istqb / sample-exam
13   / questions,

```

After the question definitions have been processed, I start a two-column layout:

```

14     renderers = {
15         jekyllDataEnd = {
16             \begin { multicol } { 2 }
17             \tablehead
18             { \hline
19               \textbf
20                 { Question~Number~(\#) } &
21               \textbf
22                 { Correct~Answer } &
23               \textbf
24                 { Learning~Objective~(LO) } &
25               \textbf
26                 { K~Level } &
27               \textbf
28                 { Number~of~Points } \\ \ }
29             \tabletail { \hline }
30             \tablelasttail { \hline }

```

Next, I define a variable that typesets the table:

```

31     \tl_set:Nn
32     \l_istqb_answer_key_table_tl
33     {

```

First, I start the table:

```

34         \begin
35         { supertabular }

```

```

36         { | C { 1.9cm } | C { 1.5cm }
37         | C { 2.4cm } | C { 1.4cm }
38         | C { 1.9cm } | } }

```

Next, I iterate over all question numbers:

```

39     \seq_map_inline:Nn
40     \g_istqb_questions_seq
41     {
42     \tl_put_right:Nn
43     \l_istqb_answer_key_table_tl
44     { \hline }

```

For each question, I add the question number:

```

45     \tl_put_right:Nn
46     \l_istqb_answer_key_table_tl
47     { \textbf { ##1 } & }

```

Next, I add the correct answer letters:

```

48     \prop_get:cnN
49     { g_istqb_answer_correct
50     _keys_prop }
51     { ##1 }
52     \l_tmpa_clist
53     \tl_put_right:Ne
54     \l_istqb_answer_key_table_tl
55     { \clist_use:Nn
56     \l_tmpa_clist
57     { ,~ } & }

```

Then, I add the learning objective:

```

58     \tl_put_right:NV
59     \l_istqb_answer_key_table_tl
60     \g_istqb_prefix_tl
61     \tl_put_right:Nn
62     \l_istqb_answer_key_table_tl
63     { - }
64     \prop_get:cnN
65     { g_istqb_question_learning
66     _objective_prop }
67     { ##1 }
68     \l_tmpa_tl
69     \tl_put_right:NV
70     \l_istqb_answer_key_table_tl
71     \l_tmpa_tl
72     \tl_put_right:Nn
73     \l_istqb_answer_key_table_tl
74     { & }

```

Next, I add the K-level:

```

75     \prop_get:NnN
76     \g_istqb_question_k_level_prop
77     { ##1 }
78     \l_tmpa_tl
79     \tl_put_right:NV
80     \l_istqb_answer_key_table_tl
81     \l_tmpa_tl
82     \tl_put_right:Nn
83     \l_istqb_answer_key_table_tl
84     { & }

```

Lastly, I add the number of points:

```

85     \prop_get:cnN
86     { g_istqb_question_number
87     _of_points_prop }
88     { ##1 }
89     \l_tmpa_tl
90     \tl_put_right:NV
91     \l_istqb_answer_key_table_tl
92     \l_tmpa_tl
93     \tl_put_right:Nn
94     \l_istqb_answer_key_table_tl
95     { \ }
96     }

```

After I have iterated over all question numbers, I end the table:

```

97     \tl_put_right:Nn
98     \l_istqb_answer_key_table_tl
99     { \end { supertabular } }

```

Then, I place the content of the variable to the input stream:

```

100     \tl_use:N
101     \l_istqb_answer_key_table_tl

```

Finally, I end the multicolumn layout:

```

102     \end { multicol } } }

```

3.4 Typesetting answers

In this section, I describe the snippet `answers` from the theme `istqb/sample-exam/answers`. This snippet typesets the list of answers in Figure 2c.

First, I load package `longtable`:

```

1 \RequirePackage { longtable }
2 \dim_const:Nn
3   \c_explanation_width_dim
4   { 11.15cm }

```

The package allows me to typeset the list of answers as a table that automatically inserts page breaks.

Next, I use snippet `questions` from theme `istqb/sample-exam` to process question definitions:

```

5 \markdownSetupSnippet
6   { answers }
7   { snippet = istqb / sample-exam
8     / questions,

```

After the question definitions have been processed, I define a variable that typesets the table:

```

9   renderers = {
10     jekyllDataEnd = {
11       \group_begin:
12       \tl_set:Nn
13         \l_istqb_answers_table_tl
14         {

```

First, I start the table and I set its heading:

```

15     \begin
16     { longtable }
17     { | C { 1.9cm } | C { 1.5cm }
18     | p

```



```

19         { \c_explanation_width_dim } 71         \l_tmpa_tl
20         | C { 2.4cm } | C { 1.4cm } 72         \tl_put_right:NV
21         | C { 1.9cm } | }           73         \l_istqb_answers_table_tl
22     \hline                          74         \l_tmpa_tl
23     \textbf                          75         \tl_put_right:Nn
24         { Question~Number~(\#) } & 76         \l_istqb_answers_table_tl
25     \textbf { Correct~Answer } &    77         { \end { minipage }
26     \multicolumn                      78         \medskip }
27         { 1 }                        79         \tl_put_right:Nn
28         { C                          80         \l_istqb_answers_table_tl
29         { \c_explanation_width_dim } 81         { & }
30         | }
31     { \textbf
32         { Explanation~/~Rationale } 82         \tl_put_right:NV
33     } &                               83         \l_istqb_answers_table_tl
34     \textbf                          84         \g_istqb_prefix_tl
35         { Learning~Objective~(LO) } & 85         \tl_put_right:Nn
36     \textbf { K-Level } &           86         \l_istqb_answers_table_tl
37     \textbf { Number-of~Points } \\ 87         { - }
38     \hline                          88         \prop_get:cnN
39     \endhead }                       89         { g_istqb_question_learning
90                                         _objective_prop }
91                                         { ##1 }
92                                         \l_tmpa_tl
93                                         \tl_put_right:NV
94                                         \l_istqb_answers_table_tl
95                                         \l_tmpa_tl
96                                         \tl_put_right:Nn
97                                         \l_istqb_answers_table_tl
98                                         { & }

Next, I iterate over all question numbers:
40     \seq_map_inline:Nn
41     \g_istqb_questions_seq
42     {
99                                         \tl_put_right:Nn
100                                        \l_istqb_answers_table_tl
101                                        { \textbf
102                                            { ##1 }
103                                        \addcontentsline
104                                        { toc }
105                                        { subsection }
106                                        { Question~\# ##1 } & }

For each question, I add the question number:
43     \tl_put_right:Nn
44     \l_istqb_answers_table_tl
45     { \textbf
46         { ##1 }
47     \addcontentsline
48     { toc }
49     { subsection }
50     { Question~\# ##1 } & }

Next, I add the correct answer letters:
51     \prop_get:cnN
52     { g_istqb_answer_correct
53     _keys_prop }
54     { ##1 }
55     \l_tmpa_clist
56     \tl_put_right:Ne
57     \l_istqb_answers_table_tl
58     { \clist_use:Nn
59     \l_tmpa_clist
60     { ,~ } & }

Then I add the explanation text:
61     \tl_put_right:Nn
62     \l_istqb_answers_table_tl
63     { \begin
64     { minipage }
65     [ t ]
66     \c_explanation_width_dim }
67     \prop_get:cnN
68     { g_istqb_question_explanation
69     _prop }
70     { ##1 }

99                                         \prop_get:NnN
100                                        \g_istqb_question_k_level_prop
101                                        { ##1 }
102                                        \l_tmpa_tl
103                                        \tl_put_right:NV
104                                        \l_istqb_answers_table_tl
105                                        \l_tmpa_tl
106                                        \tl_put_right:Nn
107                                        \l_istqb_answers_table_tl
108                                        { & }

Lastly, I add the K-level:
99                                         \prop_get:cnN
100                                        { g_istqb_question_number_of
101                                        _points_prop }
102                                        { ##1 }
103                                        \l_tmpa_tl
104                                        \tl_put_right:NV
105                                        \l_istqb_answers_table_tl
106                                        \l_tmpa_tl
107                                        \tl_put_right:Nn
108                                        \l_istqb_answers_table_tl
109                                        { & }

Lastly, I add the number of points:
109                                        \prop_get:cnN
110                                        { g_istqb_question_number_of
111                                        _points_prop }
112                                        { ##1 }
113                                        \l_tmpa_tl
114                                        \tl_put_right:NV
115                                        \l_istqb_answers_table_tl
116                                        \l_tmpa_tl
117                                        \tl_put_right:Nn
118                                        \l_istqb_answers_table_tl
119                                        { \\ \hline } }

After I have iterated over all question numbers, I
end the table:
120                                        \tl_put_right:Nn
121                                        \l_istqb_answers_table_tl
122                                        { \end { longtable } }

```

Then, I place the content of the variable to the input stream:

```

123     \t1_use:N
124     \l_istqb_answers_table_t1
125     \group_end: }}}
```

Conclusion

In this article, I have demonstrated the practical application of Markdown themes through a project that enabled the International Software Testing Qualifications Board (ISTQB) to produce their certification study materials from Markdown and YAML sources. While my previous article [2] focused on the underlying concepts of Markdown themes, this article provides concrete code used in a real-world software project. I hope this practical demonstration raises awareness of Markdown themes and illustrates how users can incorporate them into their own projects.

For ISTQB, the project for has yielded numerous benefits: Writing text in a structured format using Markdown and YAML, while generating visually appealing outputs with L^AT_EX, facilitates the separation of content from formatting. This ensures consistent application of the document’s visual style across all ISTQB content. Additionally, the structured text enables content verification against YAML schemas and ISTQB writing rules and allows for the creation of a complex knowledge base through automated processing. This enhances the quality of learning materials and reduces administrative overhead.

Moreover, the plain text formats of Markdown and YAML offer significant advantages over binary formats like Microsoft Office. They allow for efficient version control, better tracking of changes, collaborative editing, and fewer defects in the final products. The capability to produce various output formats, such as EPUB, HTML, and PDF with functional hyperlinks and cross-references, further amplifies the utility of this approach.

Acknowledgements

I wish to extend my special thanks to Tereza Vrabcová, Marei Peischl, Daniel Polan, and Petr Sojka for their invaluable insights and thorough review of my work. Their expertise and thoughtful feedback have been instrumental in shaping the final manuscript.

I would also like to thank Greg at fiverr.com/quickcartoon for their illustrations of the wolf mascot, which have provided an engaging visual identity of the Markdown package over the past four years.

References

[1] ISTQB.ORG. L^AT_EX+Markdown template, 2024. github.com/istqborg/istqb_product_base

[2] V. Novotný. Markdown 2.10.0: L^AT_EX themes & snippets, two flavors of comments, and lua_{meta}T_EX. *TUGboat* 42(2):186–193, 2021. doi.org/10.47397/tb/42-2/tb131novotny-markdown

[3] V. Starý Novotný. Parametric snippets, 2024. github.com/Witiko/markdown/discussions/445

◇ Vít Starý Novotný
 Studená 453/15
 Brno 63800, Czech Republic
[witiko \(at\) mail dot muni dot cz](mailto:witiko@mail.muni.cz)
github.com/witiko