Kian's Homework Template

https://github.com/kdkasad/typst-homework-template

Kian Kasad

2024-12-13

Contents

1 Introduction 3
2 Using the template 3
2.1 Document setup 3
2.2 Typesetting problems 4
2.3 Typesetting multi-part problems 4
2.3.1 In the prompt 5
2.3.2 In the response 5

Examples

Example 1: Document setup code. 3
Example 2: Typesetting a problem using `problem()`. 4
Example 3: Problem with points value specified. 4
Example 4: Multi-part problem prompt. 5
Example 5: Multi-part problem response. 5
Example 6: Solution with parts and sub-parts. 6

1 Introduction

This project is a document template I created for homework assignments. It is licensed under the BSD-3-Clause license and is available at the URL on the cover page.

2 Using the template

To use the template, download the file `khw.typ` from this project's repository and place it in your project/document's working directory. If your project is a Git repository, you can use a Git submodule to include the template repository within your own.

2.1 Document setup

To set up your document, import the `khw()` function and apply it to all content using a show rule:

Markup

```
#import "khw.typ": khw
#show: khw.with(
  title: [My Homework Assignment],
  author: "Your Name",
)
```

Example 1: Document setup code.

The `title` and `author` fields are used to print a title block on the first page, as well as to set the document's metadata using `document()`.

The `khw()` function supports the following optional parameters:

`title` (`content` | `str` | `none`) Document title. Defaults to `none`. If `none`, no title block is printed and no document metadata is set.

`author` (`content` | `str` | `none`) Document author. Defaults to `none`.

`date` (`datetime`) Document creation date. Defaults to `datetime.today()`.

The following options do not take effect immediately, but are used to set default options for the `problem()` and `parts()` functions provided by this template.

`newpages` (`bool`) Whether to start each problem on a new page. Defaults to false.

`problem-prefix` (`str`) The word to place before each problem number. Defaults to "Problem".

`align-numbers` (`alignment`) How to align problem numbers within the problem header. Default is `center + horizon`.

`parts-numbering` (`str` | `function`) Numbering to use for parts of a problem. Takes a value which can be used as the argument to `numbering()`.

2.2 Typesetting problems

Use the `problem()` function to typeset a problem. The function takes a non-optional content argument which can be used to specify the problem prompt/question.

Markup

```
#import "khw.typ": problem
#problem[
  #lorem(25)
]
```

Result

Problem 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus.

Example 2: Typesetting a problem using `problem()`.

The `problem()` function takes the following optional arguments:

`number` (`auto` | `str` | `content`) Specifies the number of the problem. When `auto`, problems are automatically numbered sequentially starting from 1. Defaults to `auto`.

`points` (`none` | `str` | `int` | `float` | `content`) Specifies the point value of the problem. Defaults to `none`.

Markup

```
#problem(points: 5)[
  #lorem(25)
]
```

Result

Problem 2

(5 points) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus.

Example 3: Problem with points value specified.

`newpage` (`auto` | `bool`) Whether to insert a page break before the problem header. If `auto`, the value of the `newpages` argument to the `khw()` function is used. Defaults to `auto`.

`align-number` (`auto` | `alignment`) How to align the problem number. When `auto`, the value of the `align-numbers` argument to the `khw()` function is used.

Passing a value of `center + top` will center the problem number right below the problem prefix text, so that it always appears consistent no matter the height of the problem prompts.

`outlined` (`bool`) Whether this problem shows up in the outline. If `true`, an invisible `heading()` is created for this problem, making it act like a regular heading. Defaults to `true`.

2.3 Typesetting multi-part problems

For problems with multiple parts, there are two places you might want to typeset the parts: in the prompt and in the response/solution.

2.3.1 In the prompt

The content which makes up the prompt is displayed with a `set` rule that numbers regular lists using the format `"(a)"`. You can create a regular numbered list in the prompt to typeset a multi-part prompt.

Markup

```
#problem(points: 10)[
  Explain how virtual memory speeds up the following operations:
  + Allocating zero-initialized pages.
  + Spawning child processes.
]
```

Result

Problem 3 *(10 points)* Explain how virtual memory speeds up the following operations:
(a) Allocating zero-initialized pages.
(b) Spawning child processes.

Example 4: Multi-part problem prompt.

2.3.2 In the response

Since the solution is more likely to include regular numbered lists, I decided not to just use a `set` rule and to instead make a function for typesetting parts. This also makes it quite easy to split up multi-part problems into separate source files.

Use the `parts()` function to typeset multiple parts.

Sample markup:

Markup

```
#parts[
  Virtual memory allows for copy-on-write behavior, making
  page allocation faster.
][
  Only the page table needs to be copied when spawning a child
  process, rather than the entire space of mapped memory.
]
```

Result

(a) Virtual memory allows for copy-on-write behavior, making page allocation faster.

(b) Only the page table needs to be copied when spawning a child process, rather than the entire space of mapped memory.

Example 5: Multi-part problem response.

Sub-parts can be typeset using a normal numbered list and will be numbered with lowercase Roman numerals.

Markup	`#parts[`
	`  #lorem(5)`
	`  + #lorem(5)`
	`  + #lorem(5)`
	`]`

Result	(a) Lorem ipsum dolor sit amet. i. Lorem ipsum dolor sit amet. ii. Lorem ipsum dolor sit amet.

Example 6: Solution with parts and sub-parts.