GROUP PROJECT 2.1, FLAVOR A

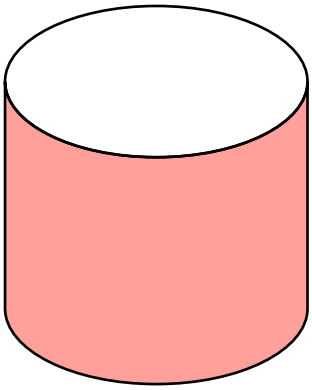
Some Group

Jane **Doe**
12345678

~~Jane~~John **Smith**
10010010

Alex **kʷikʷəƛ̓**
9999^{9999}

```
(
  "Jane Doe",
  "John Smith",
  "Alex kʷikʷəƛ\u{313}",
)
```



$12\,\mathrm{km}\ \overset{\text{wut}}{\lim_{x \to 0}}$

---

1. (3 points) one
   (a) (Extra, no) two
      i. (1 point) three

User Manual

Getting Started

"So, how do I even start using Typst?"

First thing first, it is all free.

You have 2 options: working online or offline. Since this is a "group project" template, you probably want to work online for collaboration. Here is a step-by-step guide to get you started.

1. Sign up for an account on the Typst web app.
2. Follow some guides and explore a bit.
3. (Optional) Assemble a team.
   1. Dashboard → (top left) Team → New Team.
   2. Team dashboard → (next to big team name) manage team → Add member.

Voilà! You are ready to start your math group project.

Initialize Projects

To start a math group project, simply import this package (you should have done it already) and use the `setup` function to define the project details. Here is an example:

```
#import "@preview/ubc-math-group-project:0.1.0": * // This version number may be outdated, use the latest.
#show: setup.with( // Use `.with()` instead of the full function.
  number: 1,
  flavor: "A",
  group: "A Cool Math Group",
  author("Jane", "Doe", 12345678),
  author("John", "Smith", 87654321),
)
```

Fortunately, you don't have to remember all the details. **Typst web app** can handle the initialization for you.

In the project dashboard, next to "Empty document", click on "Start from a template", search and select "ubc-math-group-project", enter your own project name, create, that easy!

In the project just initialized, you will see 2 files: `common.typ` and `project1.typ`.

If you are to reuse the template, create no new project, but add files to the existing one, like `project2typ`, `project3.typ`, etc.

`common.typ`

This file is for common content that can be shared across all projects. For instance, your group name and members.

```
#import "@local/ubc-math-group-project:0.1.0": author
// Modify as you please.
#let authors = (
  jane-doe: author("Jane", "Doe", "12345678"),
  alex-conquitlam: author(
    "Alex",
    "k\u{02b7}ik\u{02b7}\u{0259}\u{019b}\u{0313}",
    99999999,
    strname: "Alex Coquitlam"
  ),
)
#let group-name = [A Cool Group]
// Additional common content that you may add.
#let some-other-field = [Some other value]
#let some-function(some-arg) = { some-manipulation; some-output }
```

`project1.typ`

Here is where you write your project content.

```
#import "@local/ubc-math-group-project:0.1.0": *
#import "common.typ": * // Import the common content.
#show: setup.with(
  number: 1,
  flavor: [A],
  group: group-name,
  authors.jane-doe,
  // Say, Alex is absent for this project, so their entry is not included.
  // If you just want all authors, instead write:
  // ..authors.values(),
)
```

Below this `#show: setup.with(...)` is your project content.

Learn Typst

Yes, you do have to learn it, but it is simple (for our purpose).

For general techniques, consult the Typst documentation.

Other helps: `introduction`, `setup`, `author`, `caveats`, `drawing`.

User Manual

Drawing

As we are doing math, inevitably we will need to draw some graphs. Typst has some native drawing abilities, but they are very limited. There is an ad hoc Typst drawing library, a package actually, called “cetz”, with its graphing companion “cetz-plot”. Simply

```
#import drawing: *
```

to let the template import them for you.

For general drawing techniques, refer to the cetz documentation. For graphing, download and refer to the cetz-plot manual.

There are other drawing packages available, but not imported by this template, here is a brief list:

- fletcher: nodes & arrows;
- jlyfish: Julia integration;
- neoplot: Gnuplot integration.

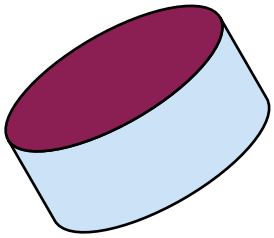
Find more visualization packages here.

Template Helpers

Besides importing the drawing packages, the `drawing` module also provides some helper functions.

For example, the `cylinder()` function draws an upright no-perspective cylinder.

```
#cetz.canvas({
  import cetz.draw: *
  group({
    rotate(30deg)
    cylinder(
      (0, 0), // Center
      (1.618, .6), // Radius: (x, y)
      2cm / 1.618, // Height
      fill-top: maroon.lighten(5%), // Top color
      fill-side: blue.transparentize(80%), // Side color
    )
  })
})
```



Other helps: `introduction`, `getting-started`, `setup`, `author`, `caveats`.

User Manual

Author

The `author()` function is to be used as an argument of the `setup()` function, providing an author dictionary. It takes the first name, last name, and student number as arguments. For example,

```
#show: setup.with(
  author("Jane", "Doe", 12345678),
  // ...
)
```

supplies

```
{
  name: {
    first: "Jane",
    last: "Doe",
  },
  id: 12345678,
}
```

And in the PDF metadata there will be a “Jane Doe” in the authors field, student number not included.

What if your last name is kʷikʷəƛ̓, that happens to type…

```
k\u{02b7}ik\u{02b7}\u{0259}\u{019b}\u{0313}
```

Well, you still call `author()`:

```
author("Alex", "k\u{02b7}ik\u{02b7}\u{0259}\u{019b}\u{0313}", 12345678)
```

If the PDF viewer gets lucky, the name will show up in file information as normal, but more often then not, it becomes something like `Alex kʷikʷəƛ\u{313}` due to incompatibility. You are recommended to provide a plain text version of the full name with argument `strname` that must be a `str`:

```
author(
  "Alex",
  "k\u{02b7}ik\u{02b7}\u{0259}\u{019b}\u{0313}",
  12345678,
  strname: "Alex Coquitlam" // This sure will have no display issue.
)
```

If `strname` is set, it will be used in the PDF metadata instead of the displayed name.

In some more extreme cases, `strname` would be a necessity, rather than a backup. Take name Ga***lli***$\overline{\ell eo}$ as an example. The name is so special that it cannot be converted to plain text. In this case, you must provide a `strname` to avoid incomprehensible PDF metadata.

```
author(
  [#underline(text(fill: purple)[Ga])#strike[*_lli_*]#overline($cal("leo")$)],
  "Smith",
  12345678,
  strname: "Gallileo Smith"
)
```

Other helps: `introduction`, `getting-started`, `setup`, `caveats`, `drawing`.