1. Outline

1. Outline 1
2. API Reference 1
  2.1. Template 1
    2.1.1. notebook 1
  2.2. Entries 2
    2.2.1. create-entry 2
    2.2.2. create-frontmatter-entry 3
    2.2.3. create-body-entry 3
    2.2.4. create-appendix-entry 4
  2.3. Glossary 4
    2.3.1. add-term 4
  2.4. Additional Datatypes 4
    2.4.1. Theme 4
    2.4.2. Context 5
  2.5. Default Theme 5
    2.5.1. Components 5
  2.6. Radial Theme 7
    2.6.1. Components 8
  2.7. Linear Theme 15
    2.7.1. Components 16

2. API Reference

2.1. Template

2.1.1. notebook

The base notebook template. This function is meant to be applied to your entire document as a show rule.

Example Usage:

```
#import themes.default: default-theme

#show: notebook.with(
  theme: default-theme
)
```

2.1.1.1. Parameters

```
notebook(
  team-name: string,
  season: string,
  year: string,
  cover: content,
  theme: theme,
  body: content
) -> content
```

2.1.1.1.1. team-name `string`

The name of your team.

Default: `none`

2.1.1.1.2. season `string`

The name of the current season.

Default: `none`

2.1.1.1.3. year `string`

The years in which the notebook is being written.

Default: `none`

2.1.1.1.4. cover `content`

The title page of the notebook.

Default: `none`

2.1.1.1.5. theme `theme`

The theme that will be applied to all of the entries. If no theme is specified, it will fall back on the default theme.

Default: `(:)`

2.1.1.1.6. body `content`

The content of the notebook. This will be ignored. Use the create-entry functions instead.

2.2. Entries

2.2.1. create-entry

The generic entry creation function. This function is not meant to be called by the user. Instead, use the three entry variants, frontmatter, body, and appendix, to create entries.

2.2.1.1. Parameters

```
create-entry(
  section: string,
  title: string,
  type: string,
  date: datetime,
  author,
  witness,
  body: content
)
```

2.2.1.1.1. section `string`

The type of entry. Takes either “frontmatter”, “body”, or “appendix”.

Default: `none`

2.2.1.1.2. title `string`

The title of the entry.

Default: `""`

2.2.1.1.3. type `string`

The type of entry. The possible values for this are decided by the theme.

Default: `none`

2.2.1.1.4. date `datetime`

The date that the entry occured at.

Default: `none`

2.2.1.1.5. body `content`

The content of the entry.

2.2.2. create-frontmatter-entry

Variant of the `#create-entry()` function that creates a frontmatter entry.

Example Usage:

```
#create-frontmatter-entry(title: "Frontmatter")[
  #lorem(50)
]
```

2.2.3. create-body-entry

Variant of the `#create-entry()` function that creates a body entry.

Example Usage:

```
#create-body-entry(
  title: "Title",
  date: datetime(year: 2024, month: 1, day: 1),
  type: "identify", // Change this depending on what your theme allows
  author: "Bobert",
  witness: "Bobernius",
)[
  #lorem(50)
]
```

2.2.4. create-appendix-entry

Variant of the `#create-entry()` function that creates an appendix entry.

Example Usage:

```
#create-appendix-entry(title: "Appendix")[
  #lorem(50)
]
```

2.3. Glossary

2.3.1. add-term

Adds a term to the glossary.

Example Usage:

```
#glossary.add-term(
  "Word",
  "The definiton of the word."
)
```

2.3.1.1. Parameters

```
add-term(
  word: string,
  definition: string
)
```

2.3.1.1.1. word `string`

The word you're defining

2.3.1.1.2. definition `string`

The definition of the word

2.4. Additional Datatypes

2.4.1. Theme

Themes are stored as dictionaries with the following fields:

rules `<function>`

The show and set rules that will be applied to the document

cover `<function>`

A function that returns the cover of the notebook. Must take ctx as input.

frontmatter-entry `<function>`

A function that returns a frontmatter entry. Must take ctx and body as input.

body-entry `<function>`

A function that returns a body entry. Must take ctx and body as input.

appendix-entry `<function>`

A function that returns a appendix entry. Must take ctx and body as input.

2.4.2. Context

Provides information to a callback about how it's being called.

Context is stored as a dictionary with the following fields:

title `<string>`

The title of the entry

type `<string>` or `<none>`

The type of the entry. This value is used differently by different templates. Refer to the template level documentation to see what this means for your theme.

date `<datetime>`

The date at which the entry started.

page-number `<integer>` or `<none>`

The page number of the first page of the entry. Only available while using the `print-toc()` utility function.

2.5. Default Theme

The default theme contains all the basic components for an engineering notebook, including a pros and cons table and a decision matrix template.

Warning

This theme is very minimal, and is generally intended as a fallback for stuff that other themes don't implement.

2.5.1. Components

2.5.1.1. toc

Prints the table of contents.

Example Usage

```
#create-frontmatter-entry(title: "Table of Contents")[
  #components.toc()
]
```

2.5.1.1.1. Parameters

toc()

2.5.1.2. glossary

Prints out the glossary.

Example Usage

```
#create-appendix-entry(title: "Glossary")[
  #components.glossary()
]
```

2.5.1.2.1. Parameters

```
glossary()
```

2.5.1.3. decision-matrix

Prints a decision matrix table.

Example Usage

```
#components.decision-matrix(
   properties: (
     (name: "Category 1"),
     (name: "Category 2"),
     (name: "Category 3")
   ),
   ("Decision", 4, 3, 2),
   ("Matrix", 1, 2, 3),
 )
```

Example Usage

```
#components.decision-matrix(
  properties: (
    (name: "Flavor", weight: 2),
    (name: "Crunchiness"), // The weight defaults to 1
  ),
  ("Sweet Potato", 1, 2),
  ("Baked Potato", 2, 1)
)
```

2.5.1.3.1. Parameters

```
decision-matrix(
  properties: array,
  ..choices: array
) -> content
```

2.5.1.3.1.1. properties `array`

A list of the properties that each choice will be rated by and the weight of each property

Default: `()`

2.5.1.3.1.2. ..choices `array`

An array containing the name of the choices as its first member, and values for each of the properties at its following indices

2.5.1.4. pro-con

Prints a pros and cons table.

Example Usage

```
#components.pro-con(
  pros: "Pros",
  cons: "Cons"
)
```

Example Usage

```
#components.pro-con(
 pros: [
   #list(
     "Sweet potato",
     "Baked potato"
   )
 ],
 cons: [
   #list(
     "Fries",
     "Wedges"
   )
 ]
)
```

2.5.1.4.1. Parameters

```
pro-con(
  pros: content,
  cons: content
) -> content
```

2.5.1.4.1.1. pros `content`

The positive aspects

Default: `[]`

2.5.1.4.1.2. cons `content`

The negative aspects

Default: `[]`

2.6. Radial Theme

The Radial theme is a minimal theme focusing on nice, rounded corners.

You can change the look of body entries by changing their type. The following types are available:

- `"identify"`: For entries about the identify stage of the engineering design process.
- `"brainstorm"`: For entries about the brainstorm stage of the engineering design process.
- `"decide"`: For entries about the decide stage of the engineering design process.
- `"build"`: For entries about the build stage of the engineering design process.

- `"program"`: For entries about the programming stage of the engineering design process.
- `"test"`: For entries about the testing stage of the engineering design process.
- `"management"`: For entries about team management
- `"notebook"`: For entries about the notebook itself

Minimal starting point:

```
// Import the template and theme here

#show: notebook.with(theme: radial-theme)

#create-frontmatter-entry(title: "Table of Contents")[
  #components.toc()
]

#create-body-entry(
  title: "Sample Entry", type: "identify", start-date: datetime(year: 1984, month: 1,
day: 1),
)[

= Top Level heading

#lorem(20)

#components.admonition(type: "note")[
  #lorem(20)
]

#components.pro-con(pros: [
  #lorem(50)
], cons: [
  #lorem(20)
])

#components.decision-matrix(
  properties: ("Flavor", "Versatility", "Crunchiness"), ("Sweet Potato", 5, 3, 1),
("White Potato", 1, 2, 3), ("Purple Potato", 2, 2, 2),
)
]

#create-appendix-entry(title: "Glossary")[
  #components.glossary()
]
```

2.6.1. Components

2.6.1.1. toc

Print out the table of contents

Example Usage:

```
#create-frontmatter-entry(title: "Table of Contents")[
  #components.toc()
]
```

2.6.1.1.1. Parameters

toc()

2.6.1.2. glossary

Print out the glossary

Example Usage:

```
#create-frontmatter-entry(title: "Glossary")[
  #components.glossary()
]
```

2.6.1.2.1. Parameters

```
glossary()
```

2.6.1.3. admonition

A message in a colored box meant to draw the reader's attention.

2.6.1.3.1. Parameters

```
admonition(
  type: string,
  body: content
) -> content
```

2.6.1.3.1.1. type `string`

The type of admonition. Available types include:

- "note"
- "example"
- "quote"
- "equation"
- "decision"
- "build"

Default: `none`

2.6.1.3.1.2. body `content`

The content of the admonition

2.6.1.4. pro-con

A table displaying pros and cons.

2.6.1.4.1. Parameters

```
pro-con(
  pros: content,
  cons: content
) -> content
```

2.6.1.4.1.1. pros `content`

The positive aspects

Default: `[]`

2.6.1.4.1.2. cons `content`

The negative aspects

Default: `[]`

2.6.1.5. decision-matrix

A decision matrix table.

2.6.1.5.1. Parameters

```
decision-matrix(
  properties: array,
  ..choices: array
) -> content
```

2.6.1.5.1.1. properties `array`

A list of the properties that each choice will be rated by

Default: `none`

2.6.1.5.1.2. ..choices `array`

An array containing the name of the choices as its first member, and values for each of the properties at its following indices

2.6.1.6. tournament

A Series of tables displaying match data from a tournament. Useful for tournament analysis entries.

2.6.1.6.1. Parameters

```
tournament(..matches: dictionary) -> content
```

2.6.1.6.1.1. ..matches `dictionary`

A list of all of the matches at the tournament. Each dictionary must contain the following fields:
- match (string) The name of the match
- red-alliance <`dictionary`> The red alliance
  - teams <`array`>
  - score <`integer`>
- blue-alliance <`dictionary`> The blue alliance
  - teams <`array`>
  - score <`integer`>
- won <`boolean`> Whether you won the match
- auton <`boolean`> Whether you got the autonomous bonus
- awp <`boolean`> Whether you scored the autonomous win point
- notes <`content`> Any additional notes you have about the match

2.6.1.7. pie-chart

Creates a labeled pie chart.

Example Usage:

```
#pie-chart(
  (value: 8, color: green, name: "wins"),
  (value: 2, color:  red, name: "losses")
)
```

2.6.1.7.1. Parameters

pie-chart(..data: `dictionary`) -> `content`

2.6.1.7.1.1. ..data `dictionary`

Each dictionary must contain 3 fields.
- value: <`integer`> The value of the section
- color: <`color`> The value of the section
- name: <`string`> The name of the section

2.6.1.8. plot

Example Usage:

```
#plot(
  title: "My Epic Graph",
  (name: "thingy", data: ((1,2), (2,5), (3,5))),
  (name: "stuff", data: ((1,1), (2,7), (3,6))),
  (name: "potato", data: ((1,1), (2,3), (3,8))),
)
```

2.6.1.8.1. Parameters

```
plot(
  title: string,
  x-label: string,
  y-label: string,
  length,
  ..data: dictionary
) -> content
```

2.6.1.8.1.1. title `string`

The title of the graph

Default: `""`

2.6.1.8.1.2. x-label `string`

The label on the x axis

Default: `""`

2.6.1.8.1.3. y-label `string`

The label on the y axis

Default: `""`

2.6.1.9. gantt-chart

A gantt chart for task management

Example Usage:

```
#gantt-chart(
  start: datetime(year: 2024, month: 1, day: 27),
  end: datetime(year: 2024, month: 2, day: 3),
  tasks: (
    ("Build Robot", (0,4)),
    ("Code Robot", (3,6)),
    ("Drive Robot", (5,7)),
    ("Destroy Robot", (7,8)),
  ),
  goals: (
    ("Tournament", 4),
  )
)
```

2.6.1.9.1. Parameters

```
gantt-chart(
  start: datetime,
  end: datetime,
  date-interval: integer,
  date-format: string,
  tasks: array,
  goals: array
) -> content
```

2.6.1.9.1.1. start `datetime`

Start date using datetime object

- year: `<integer>`
- month: `<integer>`
- day: `<integer>`

Example usage: `datetime(year: 2024, month: 7, day: 16)`

Default: `datetime`

2.6.1.9.1.2. end `datetime`

End date using datetime object

- year: `<integer>`
- month: `<integer>`
- day: `<integer>`

Example usage: `datetime(year: 2024, month: 5, day: 2)`

Default: `datetime`

2.6.1.9.1.3. date-interval `integer`

The interval between dates, seven would make it weekly

Default: `1`

2.6.1.9.1.4. date-format `string`

The way the date is formated using the `<datetime.display()>` method

Default: `"[month]/[day]"`

2.6.1.9.1.5. tasks `array`

Specify tasks using an array of arrays that have three fields each
1. `<string>` or `<content>` The name of the task
2. `<array>`(`<integer>` or `<float>`, `<integer>` or `<float>`) The start and end point of the task
3. `<color>` The color of the task line (optional)

Example sub-array: `("Build Catapult", (1,5), red)`

Default: `()`

2.6.1.9.1.6. goals `array`

Add goal markers using an array of arrays that have three fields each
1. `<string>` or `<content>` The name of the goal
2. `<integer>` or `<float>` The position of the goal
3. `<color>` The color of the goal box (optional)
   - Default is grey, but put none for no box

Example sub-array: `("Worlds", 6, red)`

Default: `none`

2.6.1.10. team
Display information about your team.

Example Usage:

```
#team(
    (
      name: "Random Person",
      picture: image("./path-to-image.png", width: 90pt, height: 90pt),
      about: [
        Likes Coding
      ],
  ),
)
```

2.6.1.10.1. Parameters
team(..members: `dictionary`) -> `content`

2.6.1.10.1.1. ..members `dictionary`

A list of members in your team. Each dictionary must contain the following fields:
- name `<string>`: The name of the team member
- picture `<content>`: An image of the team member
- about `<content>`: About the team member

2.6.1.11. label

A label that corresponds with one of the entry types.

2.6.1.11.1. Parameters

```
label(
  type: string,
  size: size
) -> content
```

2.6.1.11.1.1. type `string`

Any of the radial entry types

2.6.1.11.1.2. size `size`

The size of the label

Default: `0.7em`

2.7. Linear Theme

The Linear theme is a template that uses straight lines and boxes.

You can change the look of body entries by changing their type. The following types are available:

- `"identify"`: For entries about the identify stage of the engineering design process.
- `"brainstorm"`: For entries about the brainstorm stage of the engineering design process.
- `"decide"`: For entries about the decide stage of the engineering design process/
- `"build"`: For entries about the build stage of the engineering design process.
- `"program"`: For entries about the programming stage of the engineering design process.
- `"test"`: For entries about the testing stage of the engineering design process

Minimal Starting Point

```
#create-frontmatter-entry(title: "Table of Contents")[
  #components.toc()
]

#create-body-entry(title: "Day 1", type: "identify", date: datetime(year: 1984,
month: 1, day: 1))[
  = Heading

  #lorem(50)

  #components.pro-con(
    pros: [
      #list(
        [Sweet potato],
        [Red potato],
        [Yellow potato]
      )
    ],
    cons: [
      #list(
        [Fries],
        [Wedges],
        [Mashed]
      )
    ]
  )
]

#create-body-entry(title: "Day 2", type: "identify", date: datetime(year: 1984,
month: 1, day: 2))[
  = Another Heading

  #lorem(50)

  #components.decision-matrix(
    properties: (
      (name: "Lorem", weight: 2),
      (name: "Ipsum")
    ),
    ("Dolor", 1, 3),
    ("Sit", 2, 2),
    ("Amet", 3, 1)
  )

  == A Smaller Heading

  #lorem(50)
]

#glossary.add-term("Term 1")[
  #lorem(10)
]

#glossary.add-term("Term 2")[
  #lorem(10)
]

#create-appendix-entry(title: "Glossary")[
  #components.glossary()
]
```

2.7.1. Components

2.7.1.1. decision-matrix

Prints a decision matrix table.

Example Usage

```
#decision-matrix(
  properties: (
    (name: "Versatility", weight: 2),
    (name: "Flavor", weight: 6),
    (name: "Crunchiness"), // Defaults to a weight of 1
  ),
  ("Sweet potato", 2, 5, 1),
  ("Red potato", 2, 1, 3),
  ("Yellow potato", 2, 2, 3),
)
```

2.7.1.1.1. Parameters

```
decision-matrix(
  properties: array,
  ..choices: array
) -> content
```

2.7.1.1.1.1. properties `array`

A list of the properties that each choice will be rated by

Default: `none`

2.7.1.1.1.2. ..choices `array`

An array containing the name of the choices as its first member, and values for each of the properties at its following indices

2.7.1.2. pro-con

Prints a pros and cons table.

Example Usage

```
#pro-con(
  pros: [
    #list(
      [Sweet potato],
      [Red potato],
      [Yellow potato]
    )
  ]
  cons: [
    #list(
      [Fries],
      [Wedges],
      [Mashed]
    )
  ]
)
```

2.7.1.2.1. Parameters

```
pro-con(
  pros: content,
  cons: content
) -> content
```

2.7.1.2.1.1. pros `content`

The positive aspects

Default: `[]`

2.7.1.2.1.2. cons `content`

The negative aspects

Default: `[]`

2.7.1.3. toc

Prints the table of contents.

Example Usage

```
#create-frontmatter-entry(title: "Table of Contents")[
  #components.toc()
]
```

2.7.1.3.1. Parameters

toc()