Itemize

ver0.2.0

A typst package for users to easily customize and
format enumerations and lists.

Contents

1 Overview

The `itemize` package allows users to easily customize and format enumerations and lists. To use this package, include the following at the beginning of your document:

```typst
1  #import "@preview/itemize:0.2.0" as el
```

Use the method `default-enum-list` to override the native behaviour of `enum` and `list` by adding the following at the beginning of your document:

```typst
1  #show: el.default-enum-list
```

Now you can use `enum` and `list` as usual. Below is a comparison.

```typ
1   #let item-test = [
2     + one $vec(1, 1, 1)$
3       $
4         x^2 + y^2 = z^2
5       $
6     + #rect(height: 2em, width: 2em) #lorem(2)
7     + #block(stroke: 1pt)[two $vec(1, 1, 1,)$]
8     + $ (a + b)^2 = a^2 + 2a b + b^2 $
9     + + #lorem(2)
10    + - #lorem(2)
11      - #lorem(2)
12  ]
13  #table(
14    columns: (1fr, 1fr),
15    [native], [itemize],
16    [
17      #item-test
18    ],
19    [
20      #show: el.default-enum-list
21      #item-test
22    ],
23  )
```

native	itemize
1. one $\begin{pmatrix}1\\1\\1\end{pmatrix}$ $$x^2 + y^2 = z^2$$ 2. ☐ Lorem ipsum. 3. two $\begin{pmatrix}1\\1\\1\end{pmatrix}$ 4. $(a+b)^2 = a^2 + 2ab + b^2$ 5. 1. Lorem ipsum. 6. • Lorem ipsum. • Lorem ipsum.	1. one $\begin{pmatrix}1\\1\\1\end{pmatrix}$ $$x^2 + y^2 = z^2$$ 2. ☐ Lorem ipsum. 3. two $\begin{pmatrix}1\\1\\1\end{pmatrix}$ 4. $(a+b)^2 = a^2 + 2ab + b^2$ 5. 1. Lorem ipsum. 6. • Lorem ipsum. • Lorem ipsum.

2 Basic Features and Model Introduction

2.1 Features

The `itemize` package currently offers the following features:

1. Compatibility with native `enum` and `list` behaviors in most cases, along with fixes for certain native bugs (or providing alternative choices), such as `typst/issue#1204` and `typst/issue#529`.

2. Customization of `enum` and `list` *labels* and *bodies* by **level** and **item**:
   - Horizontal spacing settings: `indent`, `body-indent`, `label-indent`, `enum-margin` (`is-full-width`).
   - Vertical spacing settings: `item-spacing`, `enum-spacing`.
   - Label formatting settings: `..args(text-style)`, `label-align`, `label-baseline`, `label-width`.
     ▸ Customize labels in any way: `label-format`.
   - Alignment styles for labels between items: `auto-label-width`.
   - Body formatting settings: `hanging-indent`, `line-indent`.

> ‣ Set text and border styles for the body: `body-format`.

3. Enhanced `enum` features:
   - Reference functionality for `enum` numbering.
   - Resume functionality for `enum` numbering.

4. Enhanced `list` features:
   - Terms-like functionality: Temporarily change the marker of the current item using the `item` method.
   - Checklist (similar to the `cheq` package).

2.2 Main Methods

The package `itemize` primarily provides the following methods:

2.2.1 Two Styles of enum-list

- `default`-style (Typst's native style): `default-enum-list`, `default-enum`, `default-list`

- `paragraph`-style: `paragraph-enum-list`, `paragraph-enum`, `paragraph-list`

By default, the `default-*` methods indent paragraphs after the `label`, while the `paragraph-*` methods align paragraphs with the `label`. See the example below:

default-style	paragraph-style
```1 #set enum(numbering: "(A).(I).(i)") [typ]``` ```2 #show: el.default-enum-list```	```1 #set enum(numbering: "(A).(I).(i)") [typ]``` ```2 #show: el.paragraph-enum-list```
(A) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do. (I) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do. (II) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do. (B) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do. • Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do. ‣ Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do. (A) • (I) ‣ Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.	(A) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do. (I) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do. (II) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do. (B) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do. • Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do. ‣ Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do. (A) • (I) ‣ Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

The differences between `*-enum-list`, `*-enum`, and `*-list` are:

- `*-enum-list` can uniformly configure both `enum` and `list`, while allowing separate configuration through parameters:

- ▸ `enum-config` for `enum`
- ▸ `list-config` for `list`

- `*-enum` only configures `enum`, leaving nested `list` styles unchanged

- `*-list` only configures `list`, leaving nested `enum` styles unchanged

For details, see Section 3.1.

2.2.2 Enum Numbering References

To enable this feature, add the following at the beginning of your document:

```typst
1  #show: el.config.ref
```

In the `enum` items you want to reference, label them with `<some-label>`, and then use `@some-label` to reference the enum number of that item.

Example (taken from: https://github.com/typst/typst/issues/779#issuecomment-2702268234 with minor modification)

```typ
1   #show: el.config.ref.with(supplement:
    "Item")
2   #show link: set text(fill: orange)
3   #set enum(numbering: "(E1)", full: true)
4   #show: el.default-enum-list
5
6   Group axioms:
7   + Associativity <ax:ass>
8   + Existence of identity element <ax:id>
9   + Existence of inverse element <ax:inv>
10
11  #set enum(numbering: "1.a", full: true)
12  #set math.equation(numbering: "(1.1)")
13  Another important list:
14  + Newton's laws of motion are three physical laws
    that relate the motion of an object to the forces
    acting on it.
15    + A body remains at rest, or in motion at a
      constant speed in a straight line, unless it is
      acted upon by a force.
16    + The net force on a body is equal to the
      body's acceleration multiplied by its mass.
17    + If two bodies exert forces on each other,
      these forces have the same magnitude but
      opposite directions. <newton-third>
18  + Another important force is hooks law: <hook1>
19    $ arrow(F) = -k arrow(Delta x). $ <eq:hook>
      #el.elabel[hook2]
20  + $F = m a$ <eq:c> #el.elabel("eq:ma")
21
22  We covered the three group axioms @ax:ass, @ax:id
    and @ax:inv.
23
24  It is important to remember Newton's third law
    @newton-third[], and Hook's law @hook1. In @hook2
    we gave Hook's law in @eq:hook. Note that
    @eq:ma[Conclusion] is a simlified version.
```

Group axioms:
(E1) Associativity
(E2) Existence of identity element
(E3) Existence of inverse element

Another important list:
1 Newton's laws of motion are three physical laws that relate the motion of an object to the forces acting on it.
   1.a A body remains at rest, or in motion at a constant speed in a straight line, unless it is acted upon by a force.
   1.b The net force on a body is equal to the body's acceleration multiplied by its mass.
   1.c If two bodies exert forces on each other, these forces have the same magnitude but opposite directions.
2 Another important force is hooks law:

$$\vec{F} = -k\overrightarrow{\Delta x}. \tag{1}$$

3 $F = ma$

We covered the three group axioms Item (E1), Item (E2) and Item (E3).

It is important to remember Newton's third law 1.c, and Hook's law Item 2. In Item 2 we gave Hook's law in Equation 1. Note that Conclusion 3 is a simlified version.

➡ Note. For the label `<hook2>`, you cannot directly write `<hook2>`, as this would label `<hook2>` to the equation. Use the method `elabel` to label it. The same applies to `eq:c`.

• The method `elabel(<some-label>)` is equivalent to `elabel("some-label")`, and can sometimes be written as `elabel[some-label]` (provided the latter can be parsed as a string).

For more details, see Section 3.2.

2.2.3 Resuming Enum Numbering

• To enable this feature, set the `auto-resuming` parameter to `auto` in `*-enum-list` (or `*-enum`).
  ➡ Note: This feature cannot be nested. Do not set `auto-resuming` to `auto` within another.

• Use the method `resume()` to continue using the enum numbers from the previous enum at the same level.

```typ
1  #show: el.default-enum-list.with(auto-
   resuming: auto)
2  + #lorem(5)
3    + #lorem(5)
4    + #lorem(5)
5   #lorem(5)
6  + #lorem(5)
7   #el.resume() // -> 3
8    + #lorem(5)
9    + #lorem(5)
10 + #lorem(5)
11   + + #lorem(5)
```

1. Lorem ipsum dolor sit amet.
   1. Lorem ipsum dolor sit amet.
   2. Lorem ipsum dolor sit amet.

   Lorem ipsum dolor sit amet.
2. Lorem ipsum dolor sit amet.
   3. Lorem ipsum dolor sit amet.
   4. Lorem ipsum dolor sit amet.
3. Lorem ipsum dolor sit amet.
   1. 1. Lorem ipsum dolor sit amet.

• Alternatively, use `resume-label(<some-label>)` to label the enum to resume, then use `resume-list(<some-label>)` in the desired enum to continue numbering.
  ‣ If the following is added to the document:

```typst
1  #show: el.config.ref-resume
```

  You can use `@some-label` instead of `resume-list(<some-label>)`.

```typ
1  #show: el.config.ref-resume
2  #let auto-resume = el.default-enum-
   list.with(auto-resuming: auto)
3  #auto-resume[
4    + #lorem(5)
5      + #lorem(5)
6      + #lorem(5)
7      - #lorem(5)
8     #el.resume() // continue
9      + #lorem(5) #el.resume-label(<resume:demo>)
10   #lorem(5)
11   @resume:demo // resume the enum labelled with
      `resume:demo`
12   + #lorem(5)
13 ]
```

1. Lorem ipsum dolor sit amet.
   1. Lorem ipsum dolor sit amet.
   2. Lorem ipsum dolor sit amet.

   ‣ Lorem ipsum dolor sit amet.

   3. Lorem ipsum dolor sit amet.

Lorem ipsum dolor sit amet.
4. Lorem ipsum dolor sit amet.

- Alternatively, use `auto-resume-enum(auto-resuming: true)[...]` to ensure all `enum` items within `[...]` continue numbering from the previous items. For example:

```typ
1  #let resume-enum(doc) = el.default-enum-
   list(auto-resuming: auto)[
2    #el.auto-resume-enum(auto-resuming: true,
     doc)
3  ]
4  #resume-enum[
5    + #lorem(5)
6        + #lorem(5)
7        + #lorem(5)
8          + #lorem(5)
9        - #lorem(5)
10       + #lorem(5)
11         + #lorem(5)
12   #lorem(5)
13   + #lorem(5)
14 ]
```

1. Lorem ipsum dolor sit amet.
    1. Lorem ipsum dolor sit amet.
    2. Lorem ipsum dolor sit amet.
        1. Lorem ipsum dolor sit amet.

    ‣ Lorem ipsum dolor sit amet.

    3. Lorem ipsum dolor sit amet.
        2. Lorem ipsum dolor sit amet.

Lorem ipsum dolor sit amet.
2. Lorem ipsum dolor sit amet.

2.2.4 Terms-like Functionality

Now, you can use the `item` method within a `list` to temporarily change the marker of the current item. For example:

```typ
1  #show: el.default-enum-list
2  - #el.item[#sym.backslash.circle] #lorem(2)
3  - #el.item[①] #lorem(2)
```

⊘ Lorem ipsum.
① Lorem ipsum.

2.2.5 Checklist

- To enable this feature, set the `checklist` parameter to `true` in `*-enum-list` (or `*-list`), for example:

```typst
1  #show: el.default-enum-list.with(checklist: true)
```

Now you can use:

```typ
1  #show: el.default-enum-list.with(checklist:
   true)
2  - [x] checked #lorem(2)
3  - [ ] unchecked #lorem(2)
4  - [/] incomplete #lorem(2)
5  - [-] canceled #lorem(2)
```

☑ checked Lorem ipsum.
☐ unchecked Lorem ipsum.
◪ incomplete Lorem ipsum.
⊟ canceled Lorem ipsum.

- Alternatively, you can enable and configure checklist-related features using the `config.checklist` method:

```typst
1  #show: el.config.checklist
```

```typ
1  #show: el.config.checklist
2  #show: el.default-enum-list
3  - [x] checked #lorem(2)
4  - [ ] unchecked #lorem(2)
5  - [/] incomplete #lorem(2)
```

☑ checked Lorem ipsum.
☐ unchecked Lorem ipsum.
◪ incomplete Lorem ipsum.
⊟ canceled Lorem ipsum.

```
6  - [-] canceled #lorem(2)
```

For related configuration methods, see Section 3.5.

2.3 Model

The diagram below illustrates the model for `enum` and `list` in `itemize`.

- `enum` and `list` consist of multiple items.

- Each item is composed of a `label` and a `body`:
  - ‣ `label`: The enum number or list marker.
  - ‣ `body`: The content following the label.

- Level (`level`): Starts from 1, and increments if an item contains nested items.

- Item index (`n`): Starts from 1. The position of items in the same level.

Preceding Text

enum-spacing.above

label-width

indent

label

body-indent

Item 1: Paragraph 1

enum-margin

hanging-indent

par.spacing

line-indent

Item 1: Paragraph 2

item-spacing

label

Item 2: Paragraph

enum-spacing.below

Following Text
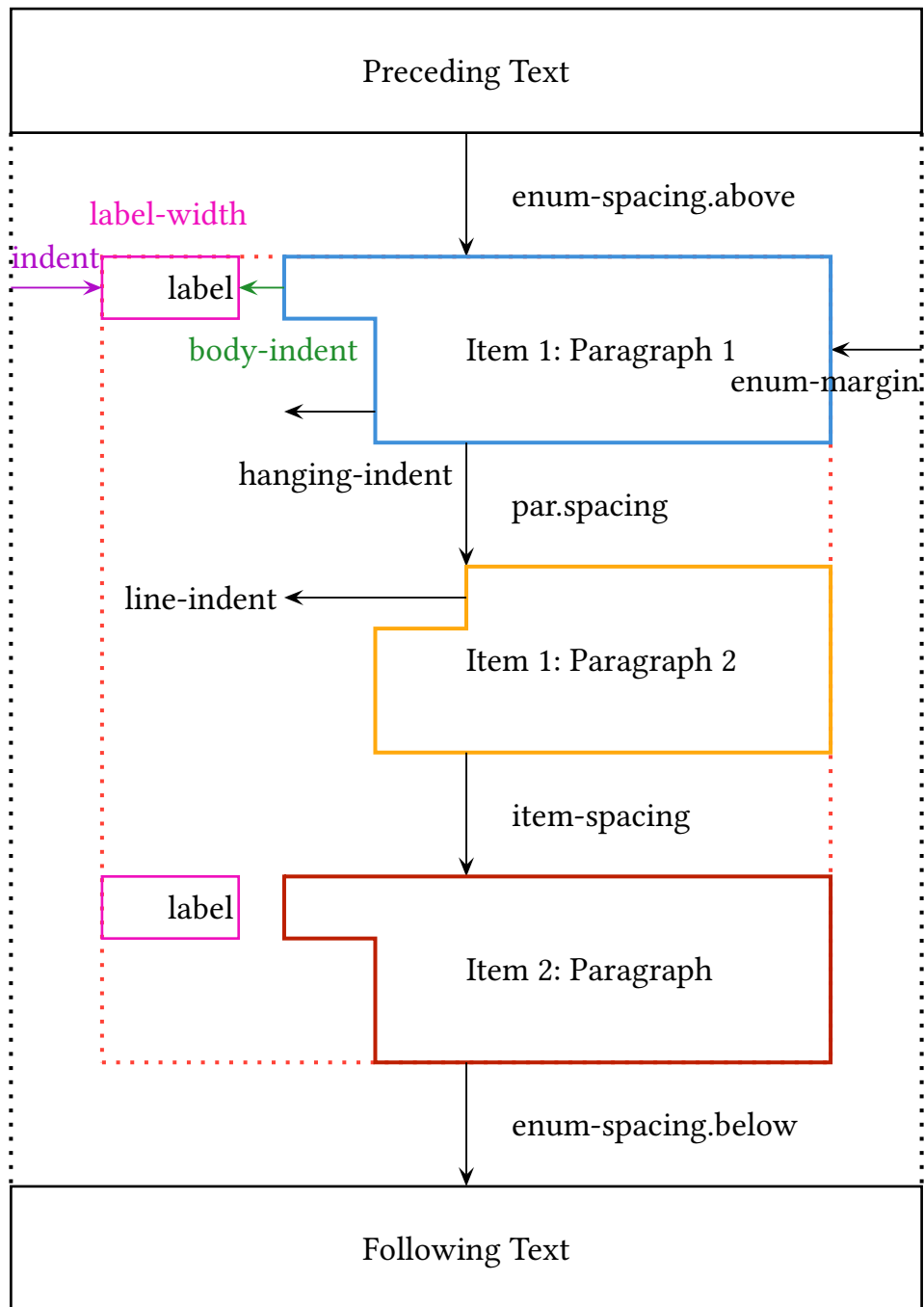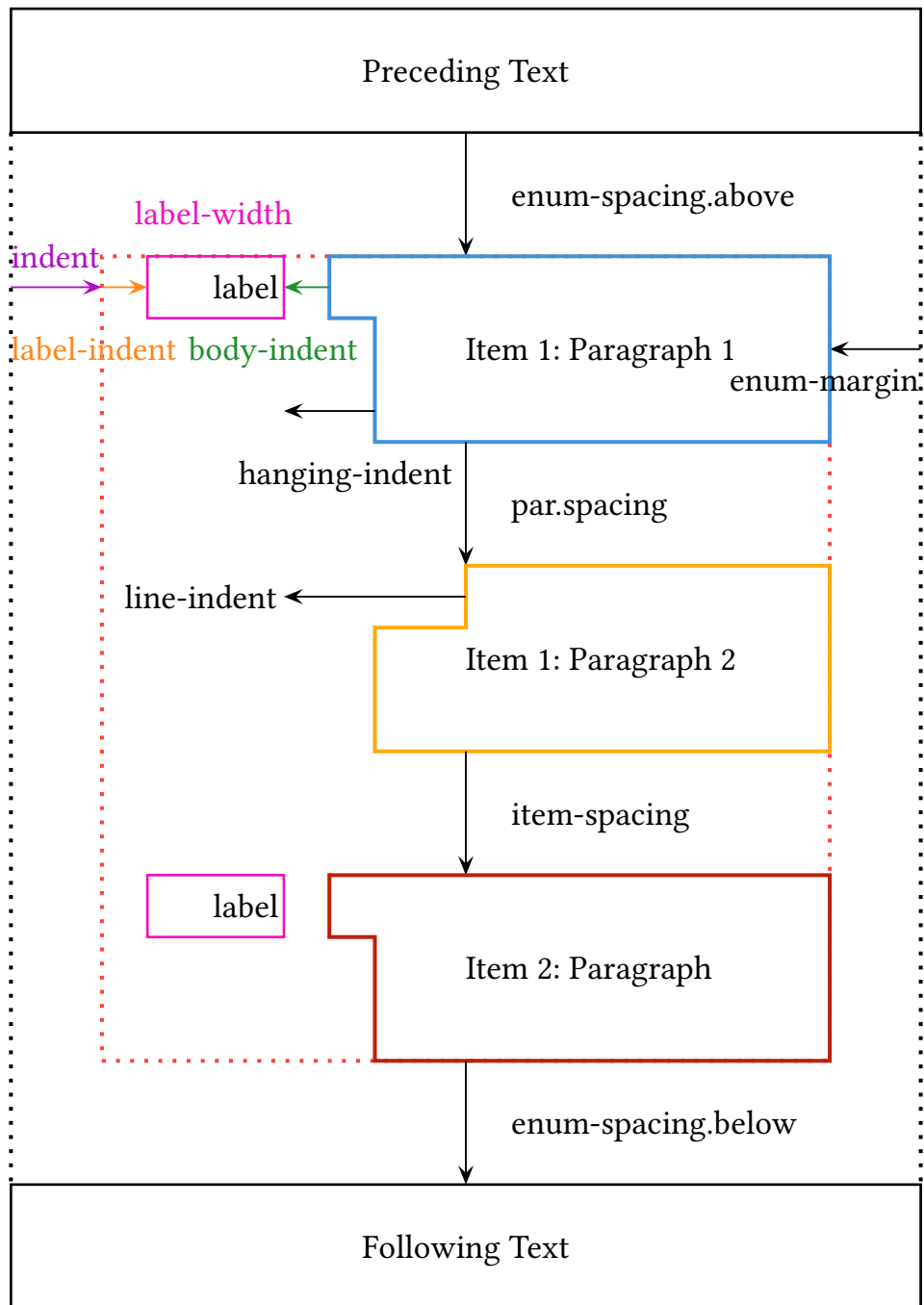
default-enum-list

The `label-indent` of default-enum-list

Code Example:

```typ
1   #set align(center)
2   #set par(justify: true)
3   #block(width: 60%, stroke: 1pt + red, inset: 5pt)[
4     #set align(left)
5     #show: el.default-enum-list.with(
6       indent: 1em,
7       label-align: right,
8       label-width: 3em,
9       body-indent: 1em,
10      hanging-indent: 2em,
11      line-indent: 4em,
12      is-full-width: false,
13      enum-margin: 4em,
14      enum-spacing: 3em,
15      item-spacing: 2.5em,
16    )
17    Preceding Text. #lorem(30)
18
19    1. Item 1: Paragraph 1. #lorem(30)
20
21      Paragraph 2. #lorem(30)
22
23    100. Item 2. #lorem(30)
24
25    Following Text. #lorem(30)
26  ]
```

Preceding Text. 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 animo, cum corpore dolemus, fieri.

1. Item 1: Paragraph 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 animo, cum corpore dolemus, fieri.

Paragraph 2. 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 animo, cum corpore dolemus, fieri.

100. Item 2. 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 animo, cum corpore dolemus, fieri.

Following Text. 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 animo, cum corpore dolemus, fieri.

3 Detailed Feature Introduction

3.1 `*-enum-list` Methods

The parameters for `default-enum-list` and `paragraph-enum-list` are similar. The following examples focus on `default-enum-list`.

```typst
1   #let default-enum-list(
2     doc: any,
3     // horizontal spacing
4     indent: array | auto | function | length = auto,
5     body-indent: array | auto | function | length = auto,
6     label-indent: array | auto | function | length = auto,
7     is-full-width: bool = true,
8     enum-margin: array | auto | function | length = auto,
9     // vertical spacing
10    enum-spacing: array | auto | dictionary | length = auto,
11    item-spacing: array | auto | function | length = auto,
12    // body style
13    hanging-indent: array | auto | function | length = auto,
14    line-indent: array | auto | function | length = auto,
15    body-format: dictionary | none = none,
16    // label style
17    ..args: arguments,
18    label-align: array | alignment | auto | function = auto,
19    label-baseline: auto | dictionary | function | length | "center" | "top" | "bottom" =
        auto,
20    label-width: array | auto | dictionary | function | length = auto,
21    label-format: array | function | none = none,
22
23    auto-base-level: bool = false,
24    checklist: array | bool = false,
25    auto-resuming: auto | bool | none = none,
26    auto-label-width: array | bool | "all" | "each" | "list" | "enum" | auto | none | =
        none,
27    // seperate setting of enum and list
28    enum-config: dictionary = (:),
29    list-config: dictionary = (:),
30  )
```

This method allows customizing the *labels* and *bodies* of enums and lists by **level** and **item**.

3.1.1 Horizontal Spacing Settings: `indent`, `body-indent`, `label-indent`, `enum-margin` (`is-full-width`)

- `indent`, `body-indent`: Control the **enum and list indentation** and **spacing** between the `label` and `body` for each level. Unless `auto` is used, these spacings cannot be changed via `#set enum` or `#set list`.

- `is-full-width`: Default is `true`, setting the `body` width to `100%`. This may temporarily fix the bug where block-level equations in the item are not center-aligned in some cases (not an ideal solution).

- `enum-margin`: Controls the **right margin** for each level of enum and list. For this parameter to take effect, `set` `is-full-width` `to` `false`. If `auto`, the item width for the current level is `auto` (native `enum` and `list` behavior).

3.1.2 Vertical Spacing Settings: `item-spacing`, `enum-spacing`

- `enum-spacing`: Controls the **spacing above and below** enum and list. Can be
  1. a `length` (same spacing above and below)
  2. or a `dictionary` with form `(above: len1, below: len2)`.

- `item-spacing`: Controls the **spacing between items** for each level.

- `label-indent`: Sets the **indentation** for the first line of an item. Refer to Section 2.3.

3.1.3 Label Formatting Settings: `..args(text-style)`, `label-align`, `label-baseline`, `label-width`, `label-format`

- `..args`: Allows passing any named arguments of `text` to format the text style of `label`. Unless `auto` is used, the text style of `label` cannot be changed via `#set text`.

```typ
1 #show: el.default-enum-list.with(
2   fill: (red, purple, auto), // level 1: red,
    level 2: purple, others: current text color
3   size: 1.2em,
4   weight: "bold",
5 )
6 #set text(fill: blue)
7 + #lorem(5)
8   + + #lorem(5)
9 + #lorem(5)
```

1. Lorem ipsum dolor sit amet.
   1. 1. Lorem ipsum dolor sit amet.
2. Lorem ipsum dolor sit amet.

- `label-align`: The `alignment` that enum numbers and list markers should have. Unless `auto` is used, it cannot be be changed via `#set enum(number-align: ...)`. For native `list`, it has no such property (default is `right`).

```typ
1 #show: el.default-enum-list.with(label-
    align: center)
2 + #lorem(5)
3 100. #lorem(5)
4   - #lorem(5)
5   - #el.item[—] #lorem(5)
6 + #lorem(5)
```
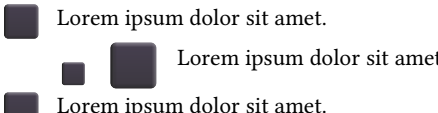
   1.   Lorem ipsum dolor sit amet.
 100.   Lorem ipsum dolor sit amet.
         ‣ Lorem ipsum dolor sit amet.
         — Lorem ipsum dolor sit amet.
 101.   Lorem ipsum dolor sit amet.

- `label-baseline`: An amount to shift the label baseline by. It can be taken
  1. `length`, `auto` or `"center"`, `"top"`, `"bottom"`
  2. or a `dictionary` with the keys:
     - ‣ `amount`: `length`, `auto` or `"center"`, `"top"`, `"bottom"`
     - ‣ `same-line-style`: `"center"`, `"top"`, `"bottom"`
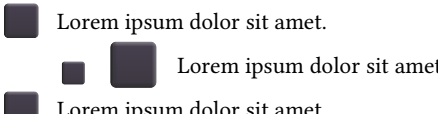
- ‣ `alone`: `bool`
‣ The first case is interpreted as `(amount: len, same-line-style: "bottom", alone: false)`
‣ When the label has a paragraph relationship with the first line of text in the current item, the label baseline will shift based on the value of `amount`.
  – For `"center"`, `"top"`, and `"bottom"`, the label will be aligned to the center, top, or bottom respectively.
    ➡ Note. We use the height of the first character (e.g., [A]) in the first line of the paragraph where the label is located. If you manually set the font style of the paragraph's text, this alignment may not be accurate, It is recommended to use the `style` parameter in `body-format` for adjustments.
  – When the value of `amount` is `auto`, set it to `0pt`.
  – If labels from different levels appear on the same line, their alignment is determined by `same-line-style`.
‣ If `alone` is `true`, it will not participate in the alignment of labels on the same line.

```typ
1 #set list(marker: [#emoji.square])
2 #show: el.default-enum-list.with(
3   size: (1.5em, auto, 2em),
4   label-baseline: "top",
5 )
6 - #lorem(5)
7   - - #lorem(5)
8 - #lorem(5)
```
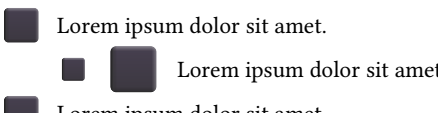
⬛ Lorem ipsum dolor sit amet.
  ⬛ ⬛ Lorem ipsum dolor sit amet.
⬛ Lorem ipsum dolor sit amet.

```typ
1 #set list(marker: [#emoji.square])
2 #show: el.default-enum-list.with(
3   size: (1.5em, auto, 2em),
4   label-baseline: "center",
5 )
6 - #lorem(5)
7   - - #lorem(5)
8 - #lorem(5)
```

⬛ Lorem ipsum dolor sit amet.
  ⬛ ⬛ Lorem ipsum dolor sit amet.
⬛ Lorem ipsum dolor sit amet.

```typ
1 #set list(marker: [#emoji.square])
2 #show: el.default-enum-list.with(
3   size: (1.5em, auto, 2em),
4   label-baseline: (amount: "center", same-line-
    style: "center"),
5 )
6 - #lorem(5)
7   - - #lorem(5)
8 - #lorem(5)
```

⬛ Lorem ipsum dolor sit amet.
  ⬛ ⬛ Lorem ipsum dolor sit amet.
⬛ Lorem ipsum dolor sit amet.

➡ Note. This is different for the text-style `baseline`.

➡ Note. These settings generally only apply when the label has a paragraph relationship with the first line of text in the current item, and not all cases satisfy this relationship. For example:

```typ
1  #set list(marker: [#emoji.square])
2  #show: el.default-enum-list.with(
3    size: (1.5em, auto, 2em),
4    label-baseline: 5pt,
5  )
6  - #lorem(5)
7    - - #rect(height: 3em)[#lorem(5)]
8  - #lorem(5)
```

■ Lorem ipsum dolor sit amet.

■ ■ Lorem ipsum dolor sit amet.

■ Lorem ipsum dolor sit amet.
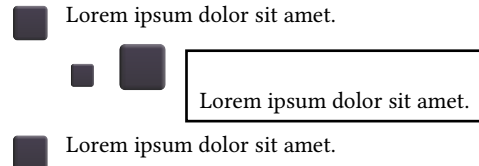
They only appear to be on the same line; in this case, the `amount` value in `label-baseline` usually does not take effect. However, if it is `"center"`, `"top"`, or `"bottom"`, the label will be aligned with the first paragraph of the current item (not the first line), For example:

```typ
1  #set list(marker: [#emoji.square])
2  #show: el.default-enum-list.with(
3    size: (1.5em, auto, 2em),
4    label-baseline: "center",
5  )
6  - #lorem(5)
7    - - #rect(height: 3em)[#lorem(5)]
8  - #lorem(5)
```

■ Lorem ipsum dolor sit amet.

■ ■ Lorem ipsum dolor sit amet.

■ Lorem ipsum dolor sit amet.
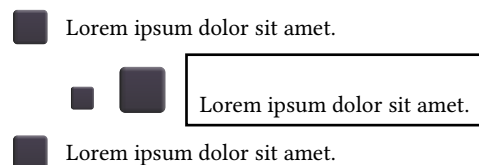
- `label-format` : Customize labels in any way. It takes
  ‣ `none` : Does not take effect
  ‣ `function`
    – The form is: `it => ...` , Access
      • `it.body` to get the label content,
      • `it.level` for the current label's level, and
      • `it.n` for the current label's index
      • `it.n-last` for the last index of the current level
    Example:

```typ
1   #import "@preview/numbly:0.1.0": numbly   typ
2   #set enum(
3     numbering: numbly(
4       "{1:A}",
5       "{2:1}.",
6       "{3:a})",
7     ),
8     full: true, // add this if use `numbly`
9   )
10  #set text(size: 10pt)
11  #show: el.default-enum-list.with(
12    label-format: it => {
13      if it.level == 1 {
14        let text-height = measure([A]).height
15        set text(size: 20pt, fill: blue,
          weight: "bold")
16        set align(center)
17        let number = box(stroke: 1pt + blue,
          inset: 5pt, width: 25pt)[#it.body]
18        let height = measure(number).height
19        move(dy: height)[#number]
20      } else {
21        it.body
22      }
23    },
24    enum-spacing: ((above: 5pt, below: 25pt),
        auto),
25    item-spacing: (-5pt, auto),
26    auto-base-level: true,
27  )
28  #lorem(10)
29  + #lorem(10)
30    - #lorem(10)
31    - #lorem(10)
32  + #lorem(15)
33    + #lorem(5)
34    + + #lorem(5)
35      + #lorem(5)
36    + #lorem(5)
37  + #lorem(15)
38  #lorem(10)
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

A Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
- Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
- Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

B Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.
1. Lorem ipsum dolor sit amet.
2. a) Lorem ipsum dolor sit amet.
   b) Lorem ipsum dolor sit amet.
3. Lorem ipsum dolor sit amet.

C Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

▸ `array`

– The (`level-1`)-th element of the array applies to the label at the `level`-th level.

– Each element in the array can be:

  1. A `function` with the form `body => ...`, which applies the label's content to this function.

  2. A `content`, which outputs this content directly.

  3. `auto` or `none`, which means no processing will be done.

  4. An `array`:
     - Its elements follow the meanings of 1, 2, and 3 above.
     - The (`n-1`)-th element of the array applies to the `n`-th item's label at the current level.

```
     #let emoji = (emoji.alien,
1    emoji.book.orange, emoji.butterfly,        typ
     emoji.cloud.storm, el.LOOP)
2    #show: el.default-enum-list.with(
3      label-format: (
4        emoji,
5        box.with(stroke: 1pt + blue, inset: 1pt),
6        auto,
7      ),
8    )
9    + #lorem(5)
10     + #lorem(5)
11       + #lorem(5)
12     + #lorem(5)
13   + #lorem(5)
14   + #lorem(5)
15   + #lorem(5)
16   + #lorem(5)
```

👽 Lorem ipsum dolor sit amet.
   1. Lorem ipsum dolor sit amet.
      1. Lorem ipsum dolor sit amet.
   2. Lorem ipsum dolor sit amet.
📙 Lorem ipsum dolor sit amet.
🦋 Lorem ipsum dolor sit amet.
🌩 Lorem ipsum dolor sit amet.
👽 Lorem ipsum dolor sit amet.

– This method can not only control the style of labels, but also control the content displayed by the current label. In the current version `0.2.x`, we recommend using `enum.numbering` (along with the `numbly` package) to control the output content of labels in `enum`, and `list.marker` to control the output content of labels in `list`.

➡ Note. We no longer recommend formatting labels using `enum.numbering` as in ver0.1.x, In complex cases, it may also cause "layout did not converge within 5 attempts".

• `label-width` : Sets the width of the label.

1. `auto` : Uses the native behavior.
2. `length` : The width of the label.
3. `dictionary` , with keys:
   ‣ `amount` : `length` , `auto` , or `"max"` .
   ‣ `style` : `"default"` , `"constant"` , `"auto"` , or `"native"` .

   The Case 2 is equivalent to `(amount: len, style: "default")` , where `len` is the specified width value; The Case 1 is equivalent to `(amount: max-width, style: "native")` , where `max-width` is the maximum width of labels at the current level.

Below we explain their differences. Let `max-width` be the maximum width of labels at the current level (also affected by `auto-label-width`). Here, `amount` represents the meaning shown in the diagram (i.e., for the default style, the hanging indent length = `amount` + `body-indent` ; for the paragraph style, the hanging indent length = `amount`).
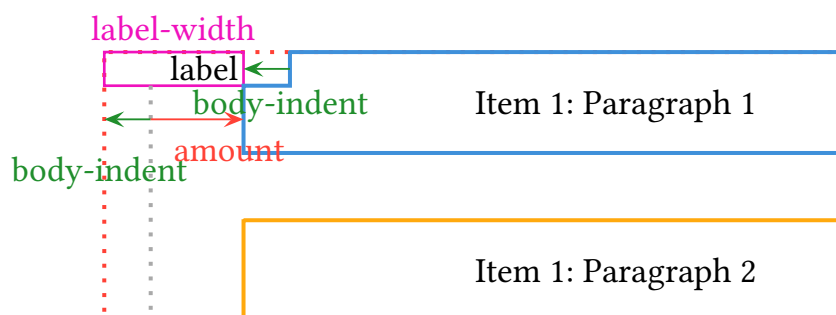
▸ When `style` is `"native"`, the label's width is rendered as `max-width`. For example:

```typ
#set par(justify: true)
#el.default-enum-list(
  label-width: (amount: 2em, style:
  "native"),
  label-format: (box.with(stroke: 1pt +
  red),box.with(stroke: 1pt + blue),),
)[
  1. #lorem(10)
  11. #lorem(10)

    #lorem(15)
    - #lorem(15)
    - #lorem(15)
  1111. #lorem(10)
  111111. #lorem(10)
]
```

> 1. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
> 11. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
>
>   Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.
>   ⊟ Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.
>   ⊟ Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.
> 1111. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
> 111111. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

▸ When `style` is `"default"`, if the actual width of the current label is less than `amount`, the label's width is rendered as `amount`; otherwise, the label's width is rendered as its actual width. For example:

```typ
#set par(justify: true)
#el.default-enum-list(
  label-width: (amount: 2em, style:
  "default"),
  label-format: (box.with(stroke: 1pt +
  red),box.with(stroke: 1pt + blue),),
)[
  1. #lorem(10)
  11. #lorem(10)

    #lorem(15)
    - #lorem(15)
    - #lorem(15)
  1111. #lorem(10)
  111111. #lorem(10)
]
```

> 1. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
> 11. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
>
>   Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.
>   ⊟ Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.
>   ⊟ Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.
> 1111. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
> 111111. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

This behavior is similar to MS-Word and can also serve as a solution for https://github.com/typst/typst/issues/4126.

▸ When `style` is `"constant"`, the label's width is rendered as `amount`. For example:

```typ
1  #set par(justify: true)
2  #el.default-enum-list(
3    label-width: (amount: 2em, style:
     "constant"),
4    label-format: (box.with(stroke: 1pt +
     red),box.with(stroke: 1pt + blue),)),
5  )[
6    1. #lorem(10)
7    11. #lorem(10)
8
9      #lorem(15)
10     - #lorem(15)
11     - #lorem(15)
12   1111. #lorem(10)
13   111111. #lorem(10)
14 ]
```

1. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
11. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

   Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.
   ⊟ Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.
   ⊟ Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.
1111. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
111111. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

▸ When `style` is `"auto"`, the label's width is rendered as its actual width; in this case, the `amount` value affects the hanging indent of the first line. For example:

```typ
1  #set par(justify: true)
2  #el.default-enum-list(
3    label-width: (amount: 2em, style: "auto"),
4    label-format: (box.with(stroke: 1pt +
     red),box.with(stroke: 1pt + blue),)),
5  )[
6    1. #lorem(10)
7    11. #lorem(10)
8
9      #lorem(15)
10     - #lorem(15)
11     - #lorem(15)
12   1111. #lorem(10)
13   111111. #lorem(10)
14 ]
```

1. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
11. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

   Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.
   ⊟ Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.
   ⊟ Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.
1111. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
111111. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

This behavior is like `terms`.

– Specifically, if `amount` is set to `-body-indent`, the next line of all paragraphs will be aligned to the left edge.

```typ
1  #set par(justify: true)
2  #el.default-enum-list(
3    body-indent: 1em,
4    label-width: (amount: -1em, style: "auto"),
5    label-format: (box.with(stroke: 1pt +
     red),box.with(stroke: 1pt + blue),)),
6  )[
7    1. #lorem(10)
8    11. #lorem(10)
9
10     #lorem(15)
11     - #lorem(15)
12     - #lorem(15)
13   1111. #lorem(10)
14   111111. #lorem(10)
15 ]
```

1. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
11. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.
⊟ Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.
⊟ Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.
1111. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
111111. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

- ‣ When `amount` is `auto`, the value of `amount` is calculated as the actual width of the current label; specifically:
    - – When `style` is `"default"`, `"constant"`, or `"auto"`, the effect is the same:

```typ
1  #set par(justify: true)
2  #el.default-enum-list(
3    label-width: (amount: auto, style: "auto"),
4    label-format: (box.with(stroke: 1pt +
       red),box.with(stroke: 1pt + blue),),
5  )[
6    1. #lorem(10)
7    11. #lorem(10)
8
9      #lorem(15)
10     - #lorem(15)
11     - #lorem(15)
12   1111. #lorem(10)
13   111111. #lorem(10)
14 ]
```

1. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
11. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.
  ⊟ Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.
  ⊟ Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.
1111. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
111111. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

    - – When `style` is `"native"`, the effect is:

```typ
1  #set par(justify: true)
2  #el.default-enum-list(
3    label-width: (amount: auto, style:
       "native"),
4    label-format: (box.with(stroke: 1pt +
       red),box.with(stroke: 1pt + blue),),
5  )[
6    1. #lorem(10)
7    11. #lorem(10)
8
9      #lorem(15)
10     - #lorem(15)
11     - #lorem(15)
12   1111. #lorem(10)
13   111111. #lorem(10)
14 ]
```

  1. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
  11. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.
  ⊟ Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.
  ⊟ Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.
1111. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
111111. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

- ‣ When `amount` is `"max"`, the value of `amount` is the maximum actual width of labels at the current level.
- ‣ Currently, the setting of `label-width` is also affected by the format of `label-format`, especially when `label-format` specifies width information. In this case, the actual width of the label will be determined by the width specified in `label-format` (usually set via constructs like `box.with(width: ...)`). It is recommended to set the container width in `label-format` to `auto` and then control it via `label-width`.

3.1.4 Body formatting settings: `hanging-indent`, `line-indent`, `body-format`

- `line-indent`, `hanging-indent`: Control the **first-line indentation** (excluding the first paragraph) and **hanging indentation** for paragraphs in each level of the enumerations and lists. Unless `auto` is used, paragraph indentation cannot be changed via `#set par`.

- `body-format`: Sets the **text style** and **border style** of the body. It is a dictionary containing the following keys:
  - ‣ `none`: Does not take effect.

- ▸ `style`: A dictionary that can include any named arguments of `text` to format the text style of `body`.
- ▸ `whole`, `outer`, `inner`: Dictionaries used to set the borders of the item.
  - – `whole`: Wraps the entire `enum` or `list`
  - – `outer`: Wraps the item (including the label)
  - – `inner`: Wraps the item (excluding the label)
  - – If `whole`, `outer`, or `inner` is omitted, the default is to set the border for `outer`.
  - – Supported border properties (consistent with `box` borders):
    - • `stroke`
    - • `radius`
    - • `outset`
    - • `fill`
    - • `inset`
    - • `clip`
  - ➡ Note. In ver0.2.x, `inset` with `relative` length is temporarily *not supported*!!!
- ▸ Each value in `style`, `whole`, `outer`, `inner` is also supported `array` and `function` types. See Section 3.1.6 and Section 3.1.7.

```
1  #set par(justify: true)                    typ
2  #el.default-enum-list(
3    body-format: (
4      whole: (
5        stroke: (1pt + purple, auto), // for
         level 1
6        inset: (5pt, auto), // for level 1
7      ),
8      outer: (
9        stroke: (1pt + green, auto), // for level
         1
10       inset: (5pt, auto),
11     ),
12     inner: (
13       stroke: (1pt + orange, auto), // for
         level 1
14       inset: (5pt, auto), // for level 1
15       fill: orange.lighten(90%) // for level 1
16     ),
17     style: (fill: (red, blue, black)), // level
         1: red, level 2: blue, others: black
18   )
19  )[
20     + #lorem(10)
21     + #lorem(10)
22      - #lorem(5)
23      - + #lorem(5)
24        + #lorem(5)
25     100. #lorem(10)
26      - #lorem(5)
27        - #lorem(5)
28        - #lorem(5)
29      - #lorem(5)
30  ]
```

1. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

2. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
   - ▸ Lorem ipsum dolor sit amet.
   - ▸ 1. Lorem ipsum dolor sit amet.
     2. Lorem ipsum dolor sit amet.

100. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

▸ Lorem ipsum dolor sit amet.
  - – Lorem ipsum dolor sit amet.
  - – Lorem ipsum dolor sit amet.

▸ Lorem ipsum dolor sit amet.

```typ
1   #set par(justify: true)
2   #el.default-enum-list(
3     body-format: (
4       outer: (
5         stroke: it => {
6           let r = calc.rem-euclid(it.n * 40 +
                it.level * 60, 255)
7           let g = calc.rem-euclid(it.n * 50 +
                it.level * 50, 255)
8           let b = calc.rem-euclid(it.n * 20 +
                it.level * 20, 255)
9           color.rgb(r, g, b)
10        },
11        inset: 5pt,
12      ),
13    )
14  )[
15      + #lorem(5)
16      + #lorem(5)
17        - #lorem(5)
18        - + #lorem(5)
19          + #lorem(5)
20      + #lorem(5)
21        - #lorem(5)
22  ]
```

1. Lorem ipsum dolor sit amet.

2. Lorem ipsum dolor sit amet.
   - Lorem ipsum dolor sit amet.
     1. Lorem ipsum dolor sit amet.
     2. Lorem ipsum dolor sit amet.

3. Lorem ipsum dolor sit amet.
   - Lorem ipsum dolor sit amet.

3.1.5 `auto-label-width`

Typically, the body indentation between different enums and lists cannot be aligned, mainly because the body indentation is related to the actual (maximum) width of the labels.

To ensure consistent first-line indentation of the body across different enums and lists, you can now set `auto-label-width` to `auto` and use the method `auto-label-item` to align the sublists within. For example:

```
1   #el.default-enum-list(                              typ
2     auto-label-width: auto,
3     body-format: (
4       whole: (
5         stroke: (1pt + red, 1pt + green, auto),
6         inset: (5pt, 5pt, auto),
7       ),
8       inner: (
9         stroke: (1pt + orange, 1pt + orange,
          auto,),
10        inset: (5pt, 5pt, auto),
11        fill: orange.lighten(90%)
12      ),
13      style: (fill: (red, blue, black)),
14    ),
15  )[
16    + #lorem(5) // 不参与
17      10. #lorem(5)
18    #el.auto-label-item(form: "all")[
19      // consider as a new enum-list
20      + #lorem(5)
21        + #lorem(10)
22        + #lorem(10)
23        - + #lorem(5)
24        - + #lorem(5)
25        100. #lorem(10)
26
27      #lorem(5)
28      10. #lorem(5)
29        + #lorem(5)
30        #lorem(5)
31        1000. #lorem(5)
32    ]
33    // 不参与
34    100. #lorem(5)
35      + #lorem(5)
36  ]
```


- Parameter `form` of the method `auto-label-item`, which can take the following values:
  ‣ `none` : No processing.
  ‣ `"each" == auto` : `Enum` and `list` are considered separately, i.e., align the label in enums and the label in lists independently.
  ‣ `"enum"` : Only align the label in `enum`.
  ‣ `"list"` : Only align the label in `list`.
  ‣ `"all"` : Align the label in both `enum` and `list`.
  ‣ `array` : Each element can be one of the above values, where the value of the `level-1`-th element represents the alignment method for the label at the `level`-th level.
  ➡ Note. The method `auto-label-item` cannot be nested.

```typ
1   #el.default-enum-list(                              typ
2       auto-label-width: auto,
3       body-format: (
4           whole: (
5               stroke: (1pt + red, 1pt + green, auto),
6               inset: (5pt, 5pt, auto),
7           ),
8           inner: (
9               stroke: (1pt + orange, 1pt + orange,
                auto,),
10              inset: (5pt, 5pt, auto),
11              fill: orange.lighten(90%)
12          ),
13          style: (fill: (red, blue, black)),
14      ),
15  )[
16      #el.auto-label-item(form: ("all", "each"))[
17          // consider as a new enum-list
18          + #lorem(5)
19            + #lorem(10)
20            + #lorem(10)
21            - + #lorem(5)
22            - + #lorem(5)
23            100. #lorem(10)
24          + #lorem(5)
25          #lorem(5)
26          - #lorem(5)
27          10. #lorem(5)
28      ]
29  ]
```

1. Lorem ipsum dolor sit amet.

   1. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
   2. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

   ‣ 1. Lorem ipsum dolor sit amet.
   ‣ 1. Lorem ipsum dolor sit amet.

   100. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

2. Lorem ipsum dolor sit amet.

Lorem ipsum dolor sit amet.

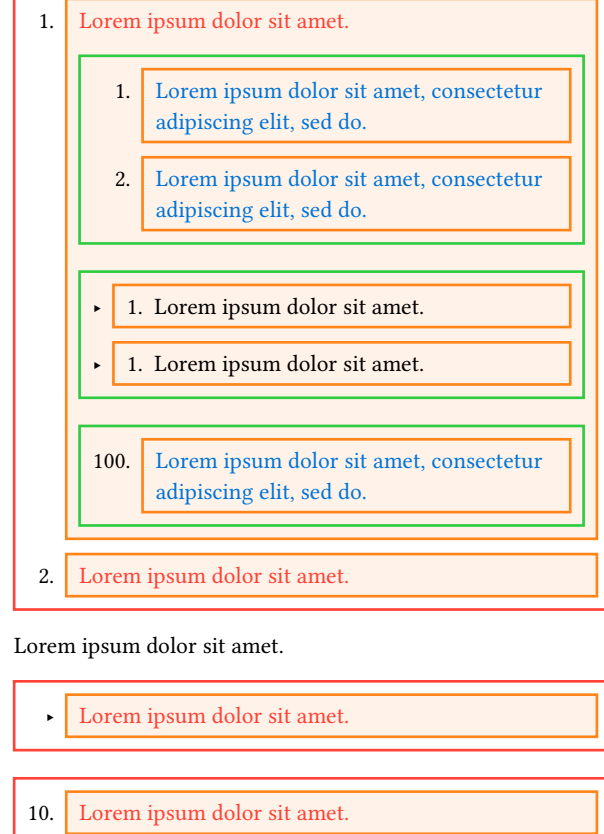   ‣ Lorem ipsum dolor sit amet.

   10. Lorem ipsum dolor sit amet.

- To align the body indentation throughout the entire document, simply add the following at the beginning of the document:

```typst
1   #show : el.default-enum-list.with(auto-label-width: "all")          typst
```

  The values and meanings of `auto-label-width` are the same as those of the `form` parameter in the `auto-label-item` method.

  ➡ Note. `auto-label-width` can be set once for non `auto` or `none`.

  ➡ Note. In a document, `auto-label-width` only retrieves the actual maximum width of labels at the current level of enums and lists.

3.1.6 Passing `array` to Parameters

If a method in the `itemize` package accepts an `array` type parameter, then:

- Each element controls the style for the corresponding level of the enum and list.

- The last element's value applies to subsequent levels.

- If the last element of the array is `LOOP`, the values in the array will be used cyclically.

- If the parameter applies to each item, each element in the array can also be an array, where each element applies to the corresponding item.

For example:

```typ
1  #show: el.default-enum-list.with(           [typ]
2    fill: (yellow, blue, auto, el.LOOP),
3    weight: "bold",
4    body-format: (
5      style: (
6        fill: (red, (purple, green, el.LOOP),
         black),
7      ),
8    ),
9  )
10 + Level1 One
11 + Level1 Two
12   + Level2-1
13     - Level3 #lorem(5)
14       - Level4 #lorem(5)
15     - Level3 #lorem(5)
16   + Level2-2
17     + Level3 #lorem(5)
18       + Level4 #lorem(5)
19       + Level4 #lorem(5)
20   + Level2-3
21   + Level2-4
22 + Level1 Three
```

1. Level1 One
2. Level1 Two
    1. Level2-1
        • Level3 Lorem ipsum dolor sit amet.
            ‣ Level4 Lorem ipsum dolor sit amet.
        • Level3 Lorem ipsum dolor sit amet.
    2. Level2-2
        1. Level3 Lorem ipsum dolor sit amet.
            1. Level4 Lorem ipsum dolor sit amet.
            2. Level4 Lorem ipsum dolor sit amet.
    3. Level2-3
    4. Level2-4
3. Level1 Three

The meaning is:

- Label styles:
    ‣ Levels 1-3: `yellow`, `blue`, `green`, the current text color (due to `auto`), then cyclically used (because the last element of the array is `LOOP`). Specifically, level 4: `yellow`.
    ‣ All levels: Bold weight.
    ‣ Level 1: `15pt` font size; subsequent levels: `12pt`.

- Body styles:
    ‣ Level 1: `red`.
    ‣ Level 2:
        – First item: `purple`.
        – Second item: `green`.
        – Then cyclically used (because the last element of the array is `LOOP`).
    ‣ Remaining levels: `black`

3.1.7 Passing `function` to Parameters

If a method in the `itemize` package accepts a `function` type parameter, its form is:

```typst
1  it => ...                                                                    [typst]
```

- The meaning is:
    ‣ The return value will be used for each level and item.
        – If the return value is an `array`, it will be used for each item.
        ➡ Note. So the parameter taking `array` and `it => array` has different meanings. The parameter taking `array` is used for each level, while the parameter taking `it => array` is used for each item.
    ‣ The parameter `it` in this method typically provides three properties:

– `level`: The current level;

– `n`: The current index;

– `n-last`: The index of the last item in the current level.

• For horizontal spacing parameters (`indent`, `body-indent`, `hanging-indent`, `line-indent`, `label-indent`), `it` also provides additional properties:

  ‣ `label-width`: Captures the label width for levels 1 to the current level.

    – Use `(it.label-width.get)(some-level)` to get the max label width at `some-level`.

    – Or `it.label-width.current` for the current level (equivalent to `(label-width.get)(level)`).

  ‣ `e`: Captures the construction (`enum` or `list`) for levels 1 to the current level. Use `(it.e.get)(some-level)` or `it.e.current`.

  ⚠ **Breaking Change**. In ver0.1.x, the `function` form here was:

```typst
1 (level, label-width, level-type) => length | auto
```

  This syntax is no longer supported in ver0.2.x. Now, the unified approach is to use `it.***` for access.

Here's an example using a `function` to align all `label`s to the left:

```typ
1  #let ex1 = [
2      + #lorem(10)
3      + #lorem(10)
4        + #lorem(10)
5          + #lorem(10)
6            + #lorem(10)
7          + #lorem(10)
8        + #lorem(10)
9  ]
10 #table(
11   rows: 2,
12   [
13     #set enum(numbering: "(A).(I).(i)", full:
         true, number-align: left)
14     #show: el.default-enum-list
15     #ex1
16   ],
17   [
18     #set enum(numbering: "(A).(I).(i)", full:
         true, number-align: left)
19     #let indent-f = it => {
20       if it.level >= 2 {
21         -(it.label-width.get)(it.level - 1) -
           (it.e.get)(it.level - 1).body-indent
22       } else {
23         auto
24       }
25     }
26     #show: el.default-enum-list.with(indent:
         indent-f)
27     #ex1
28   ],
29 )
```

(A) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
(B) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
  (B).(I) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
    (B).(I).(i) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
      (B).(I).(i).(i) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
    (B).(I).(ii) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
(C) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

(A) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
(B) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
(B).(I) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
(B).(I).(i) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
(B).(I).(i).(i) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
(B).(I).(ii) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
(C) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

Here is another example.

```
1   #let number = (a, b) => {                      [typ]
        (calc.rem-euclid(a * 100 + b * 50, 255),
2       calc.rem-euclid(a * 2 + b * 5, 255),
        calc.rem-euclid(a * 10 + b * 60, 255))
3   }
4   #show: el.default-enum-list.with(
5     size: it => {
6       (it.level + it.n)* 5pt
7     },
8     body-format: (
9       style: (
10        fill: it => color.rgb(..number(it.level,
          it.n)),
          size: it => (10pt, 12pt, 16pt) // The
11        each size of 1-4-th item's body is 10pt,
          12pt, 16pt, and others are 16pt.
12      ),
13    ),
14  )
15  + #lorem(10)
16  + #lorem(10)
17    + #lorem(10)
18      + #lorem(10)
19      + #lorem(10)
20    + #lorem(10)
21  + #lorem(10)
```

1. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
2. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
   1. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
      1. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
      2. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
   2. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
3. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

3.1.8 `enum-config` and `list-config`

When using the `*-enum-list` method, you can configure `enum` and `list` separately

- `enum-config`: Only applies to `enum`

- `list-config`: Only applies to `list`

- The parameter type is a dictionary, and the currently allowed properties (keys) are:
  ‣ `indent`,
  ‣ `body-indent`,
  ‣ `label-indent`,
  ‣ `is-full-width`,
  ‣ `item-spacing`,
  ‣ `enum-spacing`,
  ‣ `enum-margin`,
  ‣ `hanging-indent`,
  ‣ `line-indent`,
  ‣ `label-width`,
  ‣ `label-align`,
  ‣ `label-baseline`,
  ‣ `label-format`,
  ‣ `body-format`,
  ‣ any named arguments of the function of `text`

- Rules: If both `*-enum-list` and `enum-config` (`list-config`) have the same property set, the rules are:
  - ▹ The settings in `enum-config` (`list-config`) take precedence.
  - ▹ For properties of function type, a composite operation is used, where the inner function is provided by `enum-config` (`list-config`):
    - – `label-format`, `item-format`
  - ▹ For properties that are dictionaries composed of multiple attributes, these attributes are merged, and if the same attribute exists, the value from `enum-config` (`list-config`) is used.
    - – `body-format`

```
1   #set enum(numbering: "A.a.1")                    typ
2   #show: el.default-enum-list.with(
3     indent: (0.5em, 0em),
4     label-format: it => {
5       if it.level == 1 {
6           box(inset: 2pt, fill: blue.lighten(80%))
            [#it.body]
7       } else {
8           strong[#it.body]
9       }
10    },
11    body-format: (
12      style: (fill: (red, black)),
13    ),
14    enum-config: (
15      fill: (blue, red, purple, el.LOOP),
16      label-baseline: "center",
17      label-format: it => {
18        set align(center + horizon)
19        box(stroke: 1pt + blue, inset: 2pt, radius:
            1em, width: 1.2em, height: 1.2em)[#it.body]
20      },
21      body-format: (
22        outer: (
23          stroke: ((left: 2pt + blue), auto),
24        ),
25      ),
26    ),
27    list-config: (
28      fill: green,
29      label-align: center,
30    ),
31  )
32  + #lorem(5)
33    + #lorem(5)
34      + #lorem(5)
35      + #lorem(5)
36        - #lorem(5)
37        - #lorem(5)
38        - #el.item[Item] #lorem(5)
39        - #el.item[Item] #lorem(5)
40    + #lorem(5)
41    - #lorem(5)
42  + #lorem(5)
```

Ⓐ Lorem ipsum dolor sit amet.

   a Lorem ipsum dolor sit amet.

      1 Lorem ipsum dolor sit amet.

      2 Lorem ipsum dolor sit amet.

        • Lorem ipsum dolor sit amet.

        • Lorem ipsum dolor sit amet.

      Item Lorem ipsum dolor sit amet.

      Item Lorem ipsum dolor sit amet.

   b Lorem ipsum dolor sit amet.

   • Lorem ipsum dolor sit amet.

Ⓑ Lorem ipsum dolor sit amet.

3.1.9 `auto-base-level`

➡️ In ver0.1.x, we used absolute levels, meaning `*-enum-list`, `*-enum`, and `*-list` always used absolute levels for configuration, rather than treating the current level as 1. This was not intuitive.

⚠️ **Breaking change** Now in ver0.2.x, each time you use `*-enum-list`, `*-enum`, or `*-list` to configure `enum` and `list`, the current level is treated as 1.

```typ
#table(
  columns: (1fr, 1fr),
  [ver0.1.x], [ver0.2.x],
  [
    #import "@preview/itemize:0.1.2"
    #show: itemize.default-enum-list
    + $vec(1, 1, 1)$
      + #lorem(4)
        #show: itemize.default-enum-list.with(fill: (red, blue, yellow))
        // colored by yellow
        + #lorem(4)
          + #lorem(4)
        + #lorem(4)
      + #lorem(4)
    + #lorem(4)
  ],
  [
    #show: el.default-enum-list
    + $vec(1, 1, 1)$
      + #lorem(4)
        #show: el.default-enum-list.with(fill: (red, blue, yellow))
        // colored by red
        + #lorem(4)
          // colored by blue
          + #lorem(4)
        + #lorem(4)
      + #lorem(4)
    + #lorem(4)
  ],
)
```

ver0.1.x	ver0.2.x

1. $\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$
  1. Lorem ipsum dolor sit.
    1. Lorem ipsum dolor sit.
      1. Lorem ipsum dolor sit.
    2. Lorem ipsum dolor sit.
  2. Lorem ipsum dolor sit.
2. Lorem ipsum dolor sit.
	1. $\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$
	  1. Lorem ipsum dolor sit.
	    1. Lorem ipsum dolor sit.
	      1. Lorem ipsum dolor sit.
	    2. Lorem ipsum dolor sit.
	  2. Lorem ipsum dolor sit.
	2. Lorem ipsum dolor sit.

• To maintain compatibility with native behavior, the display of `numbering` and `marker` still uses absolute levels. This means even if you reconfigure `enum.numbering` and `list.marker` in sublists, the display of `numbering` and `marker` in sublists follows the absolute level rules.

  ‣ If `auto-base-level` is set to `true`, then it treats the current level as 1.

  ‣ Note the difference when `enum.full` is set to `true` (only affects the current sublist).

```
1   #table(
2     columns: (1fr, 1fr),
3     [native (`auto-base-level`: `false`)], [`auto-base-level`: `true`],
4     [
5       #show: el.default-enum-list
6       + $vec(1, 1, 1)$
7         + #lorem(4)
8           #set enum(numbering: "1.a.i.")
9           #show: el.default-enum-list.with(fill: (red, blue, yellow))
10          + #lorem(4)
11            + #lorem(4)
12          + #lorem(4)
13        + #lorem(4)
14      + #lorem(4)
15    ],
16    [
17      #show: el.default-enum-list
18      + $vec(1, 1, 1)$
19        + #lorem(4)
20          #set enum(numbering: "1.a.i.")
21          #show: el.default-enum-list.with(fill: (red, blue, yellow), auto-base-level: true)
22          + #lorem(4)
23            + #lorem(4)
24          + #lorem(4)
25        + #lorem(4)
26      + #lorem(4)
27    ],
28  )
```

native (auto-base-level : false)	auto-base-level : true

1. $\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$
  1. Lorem ipsum dolor sit.
    i. Lorem ipsum dolor sit.
      i. Lorem ipsum dolor sit.
    ii. Lorem ipsum dolor sit.
  2. Lorem ipsum dolor sit.
2. Lorem ipsum dolor sit.
	1. $\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$
	  1. Lorem ipsum dolor sit.
	    1. Lorem ipsum dolor sit.
	      a. Lorem ipsum dolor sit.
	    2. Lorem ipsum dolor sit.
	  2. Lorem ipsum dolor sit.
	2. Lorem ipsum dolor sit.

```
1   #table(
2     columns: (1fr, 1fr),
3     [native (`auto-base-level`: `false`, `full: true`)], [`auto-base-level`: `true`, `full: true`],
4     [
5       #show: el.default-enum-list
6       + $vec(1, 1, 1)$
7         + #lorem(4)
8           #set enum(numbering: "1.a.i.", full: true)
9           #show: el.default-enum-list.with(fill: (red, blue, yellow))
10          + #lorem(4)
11            + #lorem(4)
12          + #lorem(4)
13        + #lorem(4)
14      + #lorem(4)
15    ],
16    [
17      #show: el.default-enum-list
18      + $vec(1, 1, 1)$
```

```
19        + #lorem(4)
20          #set enum(numbering: "1.a.i.", full: true)
21          #show: el.default-enum-list.with(fill: (red, blue, yellow), auto-base-level: true)
22          + #lorem(4)
23            + #lorem(4)
24          + #lorem(4)
25        + #lorem(4)
26      + #lorem(4)
27    ],
28  )
```

native (`auto-base-level`: `false`, `full`: `true`)	`auto-base-level`: `true`, `full`: `true`

1. $\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$
    1. Lorem ipsum dolor sit.
      1.a.i. Lorem ipsum dolor sit.
        1.a.i.i. Lorem ipsum dolor sit.
      1.a.ii. Lorem ipsum dolor sit.
    2. Lorem ipsum dolor sit.
2. Lorem ipsum dolor sit.
	1. $\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$
	    1. Lorem ipsum dolor sit.
	      1. Lorem ipsum dolor sit.
	        1.a. Lorem ipsum dolor sit.
	      2. Lorem ipsum dolor sit.
	    2. Lorem ipsum dolor sit.
	2. Lorem ipsum dolor sit.

3.2 Referencing Enum Numbers

- To enable this feature, use the following at the beginning of the document:

  ```
  1 #show: el.config.ref                                                    typst
  ```

➡ Note. The method `ref-enum` in `itemize 0.1.x` will be deprecated in the furture; please use `config.ref` instead.

In the enum item you want to reference, label it with `<some-label>`, and then use `@<some-label>` to reference the enum number of that item.

➡ Note. If the enum number cannot be retrieved correctly, use the method `elabel(<some-label>)` to label the enum.

```
1 #show: el.config.ref                          typ
2 #set enum(numbering: "(1).(a)")
3 #show: el.default-enum-list
4 + #lorem(10)
5   + #lorem(10)
6   + $a^2+b^2 = c^2$ #el.elabel(<item:eq>)
7 + #lorem(10) <item:one>
8 The Conclusion @item:one and the equation in
  @item:eq[Item] are important.
```

(1) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
   (a) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
   (b) $a^2 + b^2 = c^2$
(2) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

The Conclusion (2) and the equation in Item (b) are important.

- Configuration Method: The `config.ref` Method
  - ▸ `full`: `auto | bool = auto`: Default is `auto`, using the `full` value from `enum`. `true` displays the full number (including parent levels); `false` displays only the current item's number.
  - ▸ `numbering`: `auto | function | str = auto`: Numbering pattern or formatter. Default is `auto`, using the `numbering` of the referenced `enum`. You can customize the style of the referenced item number.
  - ▸ `supplement`: `any | auto = auto`. Supplemental content for the reference.

```typ
1  #show: el.config.ref.with(numbering: "1.a",
   supplement: "Item", full: true)
2  #set enum(numbering: "(1).(a)")
3  #show: el.default-enum-list
4  + #lorem(10)
5    + #lorem(10)
6    + $a^2+b^2 = c^2$ #el.elabel(<item:eq2>)
7  + #lorem(10) <item:two>
8  The @item:two and the equation in
   @item:eq2[Item] are important.
```

(1) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
    (a) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
    (b) $a^2 + b^2 = c^2$
(2) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

The Item 2 and the equation in Item 1.b are important.

3.3 Resuming Enum

Related to this feature is the parameter `auto-resuming` in the methods `*-enum-list` (or `*-enum`), which can take the following values:

- `none`: Disables this feature.

- `auto`: Enables this feature.

  ▸ Use the method `resume()` to continue using the enum numbers from the previous enum at the same level.

```typ
1  #show: el.default-enum-list.with(auto-
   resuming: auto)
2  + #lorem(5)
3    + #lorem(5)
4    + #lorem(5)
5    #lorem(5)
6  + #lorem(5)
7    #el.resume() // -> 3
8    + #lorem(5)
9    + #lorem(5)
10 + #lorem(5)
11   + + #lorem(5)
```

1. Lorem ipsum dolor sit amet.
    1. Lorem ipsum dolor sit amet.
    2. Lorem ipsum dolor sit amet.

    Lorem ipsum dolor sit amet.
2. Lorem ipsum dolor sit amet.
    3. Lorem ipsum dolor sit amet.
    4. Lorem ipsum dolor sit amet.
3. Lorem ipsum dolor sit amet.
    1. 1. Lorem ipsum dolor sit amet.

  – You can also use `resume[...]` to explicitly continue using the enum numbers from the previous level (especially in ambiguous cases) and treat the `[...]` as a new `enum`. For example:

```typ
1  #show: el.default-enum-list.with(auto-
   resuming: auto)
2  + #lorem(5)
3    + #lorem(5)
4    + #lorem(5)
5    #lorem(5)
6  + #lorem(5)
7    #el.resume()[
8      // consider as a new enum
9      + #lorem(5)
10     + #lorem(5)
11   ] // -> 3
12   + #lorem(5)
13 + #lorem(5)
```

1. Lorem ipsum dolor sit amet.
    1. Lorem ipsum dolor sit amet.
    2. Lorem ipsum dolor sit amet.

    Lorem ipsum dolor sit amet.
2. Lorem ipsum dolor sit amet.
    3. Lorem ipsum dolor sit amet.
    4. Lorem ipsum dolor sit amet.

    1. Lorem ipsum dolor sit amet.
3. Lorem ipsum dolor sit amet.

▸ Alternatively, use the method `resume-label(<some-label>)` to label the enum you want to resume, and then use `resume-list(<some-label>)` in the desired enum to continue using the labelled enum numbers.

  – If you use the following in your document:

```typst
1  #show: el.config.ref-resume
```

  You can use `@some-label` instead of `resume-list(<some-label>)`.

```typ
1   #show: el.default-enum-list.with(auto-
    resuming: auto)
2   + #lorem(5)
3     + #lorem(5) #el.resume-label(<resume:a>)
4     + #lorem(5)
5     + #lorem(5)
6     #lorem(5)
7   + #lorem(5)
8   #lorem(5)
9   #el.resume-list(<resume:a>) // -> 4
10  + #lorem(5)
```

1. Lorem ipsum dolor sit amet.
   1. Lorem ipsum dolor sit amet.
   2. Lorem ipsum dolor sit amet.
   3. Lorem ipsum dolor sit amet.

   Lorem ipsum dolor sit amet.
2. Lorem ipsum dolor sit amet.

Lorem ipsum dolor sit amet.
4. Lorem ipsum dolor sit amet.

```typ
1   #show: el.config.ref-resume
2   #show: el.default-enum-list.with(auto-
    resuming: auto)
3   + #lorem(5) #el.resume-
    label(<resume:level-1>)
4     + #lorem(5)
5     + #lorem(5)
6     #lorem(5)
7   + #lorem(5)
8     @resume:level-1 // -> 4
9     + #lorem(5)
10  + #lorem(5)
```

1. Lorem ipsum dolor sit amet.
   1. Lorem ipsum dolor sit amet.
   2. Lorem ipsum dolor sit amet.

   Lorem ipsum dolor sit amet.
2. Lorem ipsum dolor sit amet.
   4. Lorem ipsum dolor sit amet.
3. Lorem ipsum dolor sit amet.

▸ Or use the method `auto-resume-enum(auto-resuming: true)[...]`, where all enum numbers within `[...]` will continue from the previous ones.

  – `auto-resuming` can also be an array, where each element indicates whether the enum numbers for the corresponding level should continue from the previous ones.

```typ
1   #show: el.default-enum-list.with(auto-
    resuming: auto)
2   + #lorem(5)
3     #el.auto-resume-enum(auto-resuming: (false,
    true))[
4       // resume sublist from the level 2
5       + #lorem(5)
6       + #lorem(5)
7         + #lorem(5)
8       #lorem(5)
9       + #lorem(5)
10        + #lorem(5)
11    ]
12  #lorem(5)
13  + #lorem(5)
14    + #lorem(5)
```

1. Lorem ipsum dolor sit amet.
   1. Lorem ipsum dolor sit amet.
   2. Lorem ipsum dolor sit amet.
      1. Lorem ipsum dolor sit amet.

   Lorem ipsum dolor sit amet.
   1. Lorem ipsum dolor sit amet.
      2. Lorem ipsum dolor sit amet.

Lorem ipsum dolor sit amet.
1. Lorem ipsum dolor sit amet.
   1. Lorem ipsum dolor sit amet.

- The method `isolated-resume-enum[...]` allows the `[...]` to be treated as a new enum with independent numbering, without affecting other enums.

```typ
 1  #table(
 2    columns: (1fr, 1fr),
 3    [Case 1], [Case 2],
 4    [
 5      #show: el.default-enum-list.with(auto-resuming: auto)
 6      + #lorem(5)
 7        + enumA #lorem(5)
 8          + enumB #lorem(5)
 9          + enumB #lorem(5)
10        + #lorem(5)
11        + #lorem(5)
12          + #lorem(5)
13          + #lorem(5)
14          + #lorem(5)
15      #lorem(5)
16      #el.resume() // -> 4
17        + #lorem(5)
18        #el.resume() // -> 4
19          + #lorem(5)
20      + #lorem(5)
21    ],
22    [
23      #show: el.default-enum-list.with(auto-resuming: auto)
24      + #lorem(5)
25        + enumA #lorem(5)
26          + enumB #lorem(5)
27          + enumB #lorem(5)
28        #el.isolated-resume-enum()[
29          // isolated sub-enum
30          + #lorem(5)
31          + #lorem(5)
32            + #lorem(5)
33            + #lorem(5)
34            + #lorem(5)
35        ]
36        #lorem(5)
37        // resume enumA
38        #el.resume() // -> 2
39        + #lorem(5)
40          // resume enumB
41          #el.resume() // -> 3
42          + #lorem(5)
43      + #lorem(5)
44    ],
45  )
```

<table>
<tr><td>Case 1</td><td>Case 2</td></tr>
<tr><td>

1. Lorem ipsum dolor sit amet.
    1. enumA Lorem ipsum dolor sit amet.
        1. enumB Lorem ipsum dolor sit amet.
        2. enumB Lorem ipsum dolor sit amet.
    2. Lorem ipsum dolor sit amet.
    3. Lorem ipsum dolor sit amet.
        1. Lorem ipsum dolor sit amet.
        2. Lorem ipsum dolor sit amet.
        3. Lorem ipsum dolor sit amet.

    Lorem ipsum dolor sit amet.
    4. Lorem ipsum dolor sit amet.
        4. Lorem ipsum dolor sit amet.
2. Lorem ipsum dolor sit amet.

</td><td>

1. Lorem ipsum dolor sit amet.
    1. enumA Lorem ipsum dolor sit amet.
        1. enumB Lorem ipsum dolor sit amet.
        2. enumB Lorem ipsum dolor sit amet.

    1. Lorem ipsum dolor sit amet.
    2. Lorem ipsum dolor sit amet.
        1. Lorem ipsum dolor sit amet.
        2. Lorem ipsum dolor sit amet.
        3. Lorem ipsum dolor sit amet.

    Lorem ipsum dolor sit amet.
    2. Lorem ipsum dolor sit amet.
        3. Lorem ipsum dolor sit amet.
2. Lorem ipsum dolor sit amet.

</td></tr>
</table>

- `bool`: If `auto-resuming` is set to `true`, all enum numbers will continue from the previous ones. It can also be set as an array, e.g., `(false, true)` means the first level does not enable the resuming feature, while subsequent levels do.

  ➡ Note. Methods with `auto-resuming` set to `true` cannot be nested within `*-enum-list` (or `*-enum`, `*-list`). A document can only have one such method.

  ‣ For small documents like exams, exercises, or CVs, if you need to resume enum numbers throughout the document, you can use the following at the beginning:

  ```typst
  1 #show : el.default-enum-list.with(auto-resuming: true)
  ```

  We recommend using this only at the document's start. Generally,
  – `el.default-enum-list.with(auto-resuming: true)` and
  – `el.default-enum-list.with(auto-resuming: auto)`

  may interfere with each other.

  ‣ For large documents like books or articles, we recommend not setting `auto-resuming` at the beginning (i.e., leave it as `none`). Instead, set this parameter to `auto` in the required sublists and use it with methods like `resume`, `resume-label`, `resume-list`, or `auto-resume-enum`.

3.4 Terms-like Functionality

In a `list`, you can use the method `item` to change the current list marker. For example:

```typ
1 #show: el.default-list
2 - #el.item[#sym.backslash.circle] #lorem(2)
```

⊘ Lorem ipsum.

```typ
1  #show: el.default-list.with(
2      size: (1.2em, auto),
3      fill: (yellow.darken(20%), auto),
4      // baseline: -0.2em,
5      label-width: (amount: 1em, style: "auto"),
6      item-spacing: 1.5em,
7      // enum-spacing: 1.2em,
8      body-format: (
9          whole: (inset: 5pt, fill: black),
10         style: (fill: white),
```

```
11        ),
12     )
13     - #el.item[为什么现在宣布《黑神话：钟馗》？] \
14        因为每年8月20日向玩家汇报我们的进展是我们的传统——今年也不例外。
15
16        然而，由于项目目前仅仅是一个空文件夹，几乎没有任何游戏画面可以分享。为了让所有人专注于开发，我们决定制作一个CG短片，让大家知道新
          项目已经启动。
17     - #el.item[为什么现在是钟馗？为什么不先做《黑神话：悟空》的续作？] \
18        感谢玩家们的坚定支持，第一部黑神话作品已经安全着陆。在完成与天选之人的旅程后，我们现在希望迈出尝试性的第一步——打造更独特的游戏体
          验，挑战更大胆的功能，为我们的世界观和叙事设计带来新鲜的想法。
19
20        钟馗是基于这种愿望和其他因素的自然选择。我们相信，在这个新项目中，我们可以做出令人耳目一新的改变，创造新事物，同时认真审视我们过
          去的缺陷和遗憾。
21
22     - #el.item[对于所有喜爱《黑神话：悟空》的朋友们：西行之旅不会在这里结束。] \
23        作为黑神话系列的第二部作品，《黑神话：钟馗》与《黑神话：悟空》相比如何？有什么相似之处和不同之处？
24        从名称上看，《黑神话：钟馗》与前作一样，都以中国古代神话和民间传说为基础。
25
26        在类型方面，它仍将是一款标准的单人ARPG，遵循与以前相同的商业模式。
27
28        然而——这次你不会扮演猴子角色。话虽如此，我们仍在探索和实验悟空与钟馗之间的具体差异。所以放轻松——让我们先让自己印象深刻，然后再
          呈现给你们。
```

> **为什么现在宣布《黑神话：钟馗》？**
> 因为每年 8 月 20 日向玩家汇报我们的进展是我们的传统——今年也不例外。
>
> 然而，由于项目目前仅仅是一个空文件夹，几乎没有任何游戏画面可以分享。为了让所有人专注于开发，我们决定制作一个 CG 短片，让大家知道新项目已经启动。
>
> **为什么现在是钟馗？为什么不先做《黑神话：悟空》的续作？**
> 感谢玩家们的坚定支持，第一部黑神话作品已经安全着陆。在完成与天选之人的旅程后，我们现在希望迈出尝试性的第一步——打造更独特的游戏体验，挑战更大胆的功能，为我们的世界观和叙事设计带来新鲜的想法。
>
> 钟馗是基于这种愿望和其他因素的自然选择。我们相信，在这个新项目中，我们可以做出令人耳目一新的改变，创造新事物，同时认真审视我们过去的缺陷和遗憾。
>
> **对于所有喜爱《黑神话：悟空》的朋友们：西行之旅不会在这里结束。**
> 作为黑神话系列的第二部作品，《黑神话：钟馗》与《黑神话：悟空》相比如何？有什么相似之处和不同之处？从名称上看，《黑神话：钟馗》与前作一样，都以中国古代神话和民间传说为基础。
>
> 在类型方面，它仍将是一款标准的单人 ARPG，遵循与以前相同的商业模式。
>
> 然而——这次你不会扮演猴子角色。话虽如此，我们仍在探索和实验悟空与钟馗之间的具体差异。所以放轻松——让我们先让自己印象深刻，然后再呈现给你们。

3.5 Checklist

This feature is essentially the same as the package `cheq` . However, in native `enum`, labels and bodies are top-aligned, while in `itemize`, they are aligned within the same paragraph. This causes misalignment when using `cheq` with `itemize`. Additionally, `cheq` uses `enum` to implement checklists, which means `enum` configurations affect the checklist. Therefore, we have added the `checklist` feature to `itemize` with minor enhancements and fixes.

- To enable this method, set the `checklist` parameter to `true` in `*-enum-list` (or `*-list`). For example:

  ```typst
  1 #show: el.default-enum-list.with(checklist: true)
  ```

  Now we can use:

```typ
1  #show: el.default-enum-list.with(checklist:
   true)                                                [typ]
2  - [x] checked #lorem(2)
3  - [ ] unchecked #lorem(2)
4  - [/] incomplete #lorem(2)
5  - [-] canceled #lorem(2)
```

☑ checked Lorem ipsum.
☐ unchecked Lorem ipsum.
◧ incomplete Lorem ipsum.
⊟ canceled Lorem ipsum.

The parameter can also accept an array to specify which levels of the list should enable the checklist feature.

- You can also enable and configure checklist-related features using the method `config.checklist`.

  ‣ You can also enable the checklist feature by using it in the required places along with `el.default-enum-list`.

    ```typst
    1  #show: el.config.checklist
    ```

  ‣ Parameters for `config.checklist`:

    – `baseline: array | auto | "center" | "top" | "bottom" = auto`: Alignment method for labels and bodies (same as `label-baseline` with `"center"`, `"top"`, `"bottom"`). Default is `auto`, controlled by `label-baseline`.

    ```typ
    1  #show:
       el.config.checklist.with(baseline:     [typ]
       "center")
    2  #show: el.default-enum-list.with(
    3    size: 1.5em,
    4  )
    5  - [x] #lorem(5)
    6  - [/] #lorem(5)
    7  - #el.item[#sym.times.circle.big] #lorem(5)
    ```

    ☑ Lorem ipsum dolor sit amet.
    ◧ Lorem ipsum dolor sit amet.
    ⊗ Lorem ipsum dolor sit amet.

    – `enable: array | bool = true`: Whether to enable the checklist feature.

    – `extras: array | bool = false`: Whether to enable additional commands, i.e., use the following:

    ```typc
    1          ">": "➡",                              [typc]
    2          "<": "📅",
    3          "?": "❓",
    4          "!": "❗",
    5          "*": "⭐",
    6          "\"": "❝",
    7          "l": "📍",
    8          "b": "🔖",
    9          "i": "ℹ",
    10         "S": "💰",
    11         "I": "💡",
    12         "p": "📌",
    13         "c": "👎",
    14         "f": "🔥",
    15         "k": "🔑",
    16         "w": "🏆",
    ```

```typ
17        "u": "🔼",
18        "d": "🔽",
```

```typ
1  #show: el.config.checklist.with( extras:
   true)
2  #show: el.default-enum-list
3  - [>] checked #lorem(2)
4  - [?] unchecked #lorem(2)
5  - [f] incomplete #lorem(2)
6  - [d] canceled #lorem(2)
```

➡ checked Lorem ipsum.
❓ unchecked Lorem ipsum.
🔥 incomplete Lorem ipsum.
🔽 canceled Lorem ipsum.

- `fill: array | auto | color = auto` : Border color. If set to `auto`, it uses the current label's style.

- `radius: array | length = 0.1em` : Border radius.

- `solid: array | none | color = none` : Solid color. Default is `none`.

```typ
1  #show: el.config.checklist.with(
2    fill: (blue, auto),
3    radius: (.5em, 0em),
4    solid: gray.lighten(50%),
5  )
6  #show: el.default-enum-list
7  - [x] checked #lorem(2)
8  - [ ] unchecked #lorem(2)
9    - [N] #lorem(2)
10 - [/] incomplete #lorem(2)
11 - [-] canceled #lorem(2)
```

⊘ checked Lorem ipsum.
◯ unchecked Lorem ipsum.
  Ⓝ Lorem ipsum.
◑ incomplete Lorem ipsum.
⊖ canceled Lorem ipsum.

- `enable-character: array | bool = true` : When set to `true`, if the character in `[...]` is not among `x, , - /` or the extras characters (if `extras` is `true`), the character in `[...]` will be displayed. For example:

```typ
1  #show: el.default-enum-list
2  #show: el.config.checklist
3  - [N] #lorem(5)
4  - [A] #lorem(5)
5
6  #show: el.config.checklist.with(enable-
   character: false)
7  - [N] #lorem(5)
8  - [A] #lorem(5)
```

Ⓝ Lorem ipsum dolor sit amet.
Ⓐ Lorem ipsum dolor sit amet.

- [N] Lorem ipsum dolor sit amet.
- [A] Lorem ipsum dolor sit amet.

- `symbol-map: dictionary | function = (:)` : Custom commands.
  - Can replace built-in commands or add new ones.
    ▸ The format is: `("some-character": content)`.
  - If a function is specified, its form is: `it => dictionary`, where `it` provides three properties: `fill`, `radius`, `solid`.

```typ
1  #show: el.default-enum-list.with(fill:
   blue)
2  #show: el.config.checklist.with(
3    symbol-map: (" ": emoji.checkmark, "A":
     emoji.a),
4  )
5  - [ ] #lorem(5)
6  - [A] #lorem(5)
7
8
9  #let box-sym(fill: white, stroke:
   rgb("#616161"), radius: .1em, body) = box(
10   stroke: .05em + stroke,
11   fill: fill,
12   height: 1em,
13   width: 1em,
14   radius: radius,
15 )[#body]
16 #show: el.config.checklist.with(
17   fill: blue,
18   solid: luma(95%),
19   symbol-map: it => (
20     " ": box-sym(fill: it.solid, stroke:
       it.fill, radius: .2em, []),
21     "x": box-sym(fill: it.fill, stroke:
       it.fill, radius: .2em, [#text(fill:
       white)[#emoji.checkmark]]),
22     "A": box-sym(fill: it.fill, stroke:
       it.fill, radius: .2em, [#text(fill:
       white)[A]]),
23   ),
24 )
25 - [ ] #lorem(5)
26 - [x] #lorem(5)
27 - [A] #lorem(5)
```

✓ Lorem ipsum dolor sit amet.
Ⓐ Lorem ipsum dolor sit amet.

☐ Lorem ipsum dolor sit amet.
☑ Lorem ipsum dolor sit amet.
Ⓐ Lorem ipsum dolor sit amet.

– `enable-format: array | bool = false` : Whether to enable formatting for the body, with the format content determined by `format-map`. The default formatting is for the character `"-"`, with the formatting function:

```typst
1  it => strike(text(fill: rgb("#888888"), it))
```

```typ
1  #show: el.default-enum-list
2  #show: el.config.checklist.with(enable-
   format: true)
3  - [ ] #lorem(3)
4  - [-] #lorem(3)
```

☐ Lorem ipsum dolor.
⊟ ~~Lorem ipsum dolor.~~

➡ Note. According to our design, the text-style of checklist labels cannot be formatted, but other text formatting will still affect the label.

➡ Note. If `[]` is followed by `list` or `enum`, the label formatting will be incorrect (because the label and the current body are treated as the same paragraph). For example:

Typically works as expected:

```typ
1  #show: el.default-enum-list                    typ
2  #show: el.config.checklist.with(enable-
   format: true)
3  - [ ] #lorem(3)
4  - [-]
5    + #lorem(3)
6    + #lorem(3)
7  - - [-] #lorem(3)
```

Incorrect formatting (uncommon usage):

```typ
1  #show: el.default-enum-list                    typ
2  #show: el.config.checklist.with(enable-
   format: true)
3  - [ ] #lorem(3)
4  + - [-]
5      + #lorem(3)
6      + #lorem(3)
```

One solution (which may impact performance) is to exclude body formatting from the label by using `item-format: it => it.body`.

```typ
1  #show: el.default-enum-list.with(item-         typ
   format: it => it.body)
2  #show: el.config.checklist.with(enable-
   format: true)
3  - [ ] #lorem(3)
4  + - [-]
5      + #lorem(3)
6      + #lorem(3)
```

- `format-map: dictionary = (:)`, Formatting for list items.
  - The format follows: `("some-character": some-function)`.
  - `some-function` takes the form `it => ...`, It applies the body of the current `some-character` item to this method.

```typ
1   #show: el.default-enum-list                   typ
2   #show: el.config.checklist.with(
3     enable-format: true,
4     format-map: (
5       "!": it => highlight(it),
        "-": it => strike(text(fill:
6       rgb("#888888"), emph(it))), // Overrides
        the format-function of "-"
7     ),
8   )
9   - [ ] #lorem(3)
10  - [-] #lorem(3)
11  - [!] #lorem(3)
```

3.6 Enhancements to `list.marker`

The `itemize` package improves the parsing of `list.marker` as a `function` type (supporting label control for each item). Now, the `function` can be defined as:

```typst
1  level => n => content
```

or

```typst
1  level => array
```

Here:

- `level`: The relative nesting level of the `list`, cyclically used.

  ⚠ **Breaking change**: For compatibility with the native `list.marker` behavior, unlike the methods provided by `itemize`, the level here starts from 0 (see `issue#3`).

  ▸ Also, the item'index starts from 0.

```typ
1   #let test-list = [
2      - #lorem(5)
3        - #lorem(5)
4        - #lorem(5)
5      - #lorem(5)
6   ]
7   #[
8      #show: el.default-enum-list.with(label-
       format: it => [#it.level - #it.n])
9      #test-list
10  ]
11  #[
12     #set list(marker: level => n => [#level -
       #n])
13     #show: el.default-enum-list.with(auto-base-
       level: true)
14     #test-list
15  ]
```

```
1 - 1  Lorem ipsum dolor sit amet.
       2 - 1  Lorem ipsum dolor sit amet.
       2 - 2  Lorem ipsum dolor sit amet.
1 - 2  Lorem ipsum dolor sit amet.

0 - 0  Lorem ipsum dolor sit amet.
       1 - 0  Lorem ipsum dolor sit amet.
       1 - 1  Lorem ipsum dolor sit amet.
0 - 1  Lorem ipsum dolor sit amet.
```

- `n => content` or `array`: The `label` for each `item` in the same level.

```typ
1   #show: el.default-enum-list
2   #let marker = level => {
3     if level == 0 { // stand for level 1!!!
4        ([#sym.suit.club.filled],
         [#sym.suit.spade.filled],
         [#sym.suit.heart.filled], el.LOOP)
5     } else if level == 1 { // stand for level 2!!!
         The first item's number starts from 0
6        n => rotate(24deg * n, box(rect(stroke: 1pt +
         blue, height: 1em, width: 1em)))
7     } else { // others
8        [#sym.ballot.check.heavy]
9     }
10  }
11  #set list(marker: marker)
12  - #lorem(5)
13  - #lorem(5)
14    - #lorem(5)
```

```
♣ Lorem ipsum dolor sit amet.
♠ Lorem ipsum dolor sit amet.
    □ Lorem ipsum dolor sit amet.
    ◇ Lorem ipsum dolor sit amet.
        ☑ Lorem ipsum dolor sit amet.
        ☑ Lorem ipsum dolor sit amet.
    ◇ Lorem ipsum dolor sit amet.
    ◇ Lorem ipsum dolor sit amet.
    □ Lorem ipsum dolor sit amet.
♥ Lorem ipsum dolor sit amet.
    □ Lorem ipsum dolor sit amet.
♣ Lorem ipsum dolor sit amet.
```

```
15    - #lorem(5)
16      - #lorem(5)
17      - #lorem(5)
18    - #lorem(5)
19    - #lorem(5)
20    - #lorem(5)
21  - #lorem(5)
22    - #lorem(5)
23  - #lorem(5)
```

3.7 `*-enum` and `*-list` Methods

These methods function similarly to `*-enum-list`. However,

- `*-enum` only applies to `enum` without affecting nested list styles.

- `*-list` only applies to `list` without affecting nested enum styles.

```
1   #show: el.default-enum.with(            [typ]
2     fill: (red, blue, green),
3     weight: "bold",
4   )
5   + #lorem(5)
6     + #lorem(5)
7     - #lorem(5)
8       - #lorem(5)
9       + - #lorem(5)
10      - + #lorem(5)
11    - #lorem(5)
12  + #lorem(5)
```

1. Lorem ipsum dolor sit amet.
   1. Lorem ipsum dolor sit amet.
   • Lorem ipsum dolor sit amet.
     ‣ Lorem ipsum dolor sit amet.
     1. ‣ Lorem ipsum dolor sit amet.
     ‣ 1. Lorem ipsum dolor sit amet.
   • Lorem ipsum dolor sit amet.
2. Lorem ipsum dolor sit amet.

Comparison of differences:

```
1   #show: el.default-enum-list.with(       [typ]
2     fill: (red, blue, green),
3     weight: "bold",
4   )
5   + #lorem(5)
6     + #lorem(5)
7     - #lorem(5)
8       - #lorem(5)
9       + - #lorem(5)
10      - + #lorem(5)
11    - #lorem(5)
12  + #lorem(5)
```

1. Lorem ipsum dolor sit amet.
   1. Lorem ipsum dolor sit amet.
   • Lorem ipsum dolor sit amet.
     ‣ Lorem ipsum dolor sit amet.
     1. ‣ Lorem ipsum dolor sit amet.
     ‣ 1. Lorem ipsum dolor sit amet.
   • Lorem ipsum dolor sit amet.
2. Lorem ipsum dolor sit amet.

➡ Note. The behavior in ver0.2.x differs slightly from ver0.1.x. To separately configure `enum` and `list` in the enum-list, use `enum-config` and `list-config` in `*-enum-list`. The following approach cannot separately configure enum and list:

```typ
1  // This show-rule cannot be applied
2  #show: el.default-enum.with(
3    fill: (red, blue, green),
4    weight: "bold",
5  )
6  // Only this show-rule is applied
7  #show: el.default-list.with(fill: (yellow,
   orange))
8  + #lorem(5)
9    + #lorem(5)
10 - #lorem(5)
11   - #lorem(5)
12   + - #lorem(5)
13   - + #lorem(5)
14 - #lorem(5)
15 + #lorem(5)
```

1. Lorem ipsum dolor sit amet.
   1. Lorem ipsum dolor sit amet.
   ‣ Lorem ipsum dolor sit amet.
     – Lorem ipsum dolor sit amet.
     1. – Lorem ipsum dolor sit amet.
     – 1. Lorem ipsum dolor sit amet.
   ‣ Lorem ipsum dolor sit amet.
2. Lorem ipsum dolor sit amet.

In ver0.2.x, we do not recommend the following approach (it does not meet expectations).

```typst
1  #show enum: el.default-enum.with(...)
2  #show list: el.default-list.with(...)
```

The correct approach is:

```typst
1  #show : el.default-enum-list.with(
2    enum-config: (
3      ...
4    )
5    list-config: (
6      ...
7    )
8  )
```

3.8 Set-Rule

In the current version of `typst` (0.14), we cannot use custom functions with `set rule`, which prevents us from retaining previously set property values when using `default-item-list`, `paragraph-item-list`, etc. Now, with the `elembic` package, we can achieve this functionality. To this end, we have repackaged `default-item-list` and `paragraph-item-list` into `set-default` and `set-paragraph` methods, which work the same as the original functions but retain previously set property values.

Usage instructions:

Include the following at the beginning of your document:

```typst
1  #show: el.set-default() // need ()!!!
```

➡ Note. The key difference is that `set-default` requires **parentheses**!

```typ
 1   #let item = [                                    typ
 2     + #lorem(5)
 3       + #lorem(5)
 4       + + #lorem(5)
 5       + - #lorem(5)
 6     + #lorem(5)
 7   ]
 8
 9   #show: el.set-default()
10   #set enum(numbering: "(1).(a).(i)")
11   #item
12
13   // change the label color
14   #show: el.set-default(fill: (red, blue, green,
      auto))
15   #item
16
17   // change the label size
18   #show: el.set-default(size: (20pt, 16pt, 14pt,
      auto))
19   #item
20
21   // change the body-indent and indent
22   #show: el.set-default(body-indent: (auto, 0.5em),
      indent: (auto, 0em, 1em, auto))
23   #item
24
25   // use the default style
26   #show: el.set-default()
27   #item
```

(1) Lorem ipsum dolor sit amet.
  (a) Lorem ipsum dolor sit amet.
  (b) (i) Lorem ipsum dolor sit amet.
  (c) • Lorem ipsum dolor sit amet.
(2) Lorem ipsum dolor sit amet.
(1) Lorem ipsum dolor sit amet.
  (a) Lorem ipsum dolor sit amet.
  (b) (i) Lorem ipsum dolor sit amet.
  (c) • Lorem ipsum dolor sit amet.
(2) Lorem ipsum dolor sit amet.

(1) Lorem ipsum dolor sit amet.
  (a) Lorem ipsum dolor sit amet.
  (b) (i) Lorem ipsum dolor sit amet.
  (c) • Lorem ipsum dolor sit amet.
(2) Lorem ipsum dolor sit amet.

(1) Lorem ipsum dolor sit amet.
  (a) Lorem ipsum dolor sit amet.
  (b)   (i) Lorem ipsum dolor sit amet.
  (c)   • Lorem ipsum dolor sit amet.
(2) Lorem ipsum dolor sit amet.

(1) Lorem ipsum dolor sit amet.
  (a) Lorem ipsum dolor sit amet.
  (b) (i) Lorem ipsum dolor sit amet.
  (c) • Lorem ipsum dolor sit amet.
(2) Lorem ipsum dolor sit amet.

3.9 Experimental Features

3.9.1 Clearing Recorded Information in resuming enum

When implementing the `resume-label` and `resume-list` features in `resuming enum`, the `itemize` package needs to record the sequence numbers of the marked enums. If these records are no longer needed in the document, you can call the `adv.reset-resume()` method to clear this information. One common use case is:

```typc
1    // New enum-before                                           typc
2
3    // el.adv.reset-resume()
4    // New enum-here:
5    // el.adv.reset-resume()
6
7    // New enum-after
```

➡ Improper use of `adv.reset-resume()` may break the `resuming enum` functionality.

3.9.2 Customizing Body Formatting

In `*-enum-list` (`*-enum` and `*-list`), you can freely customize the format of each item's body using the `item-format` property.

• It accepts
  1. `none`: Does not take effect.
  2. a `function` with form:

```typst
1  it => ...
```

which will apply to the body of each item'body. `it` has the following properties:

- ‣ `it.level`: The level of the current item.
- ‣ `it.body`: The body of the current item.
- ‣ `it.n`: The index of the current item in the current level.
- ‣ `it.n-last`: The index of the last item in the current level.

3. an `array`, whose elements are `function` with form `body => ...` which will be applied to the body of each item in turn.

4. or a `dictionary` with the following keys:

- ‣ `whole`: Wraps the entire `enum` or `list`
- ‣ `outer`: Wraps each item (including the label)
- ‣ `inner`: Wraps each item (excluding the label)
- ‣ If `whole`, `outer`, or `inner` is omitted, the default is `outer`.
- ‣ The values are taken as Case 2 and Case 3 above.

```typ
1   #set enum(numbering: "(a)(i)(1)")
2   #show: el.default-enum-list.with(
3     item-format: it => {
4       if it.level == 2 {
5         // Equivalent to: if it.level >= 2
6         show: highlight
7         it.body
8       } else {
9         it.body
10      }
11    },
12  )
13  + #lorem(10)
14    + #lorem(10)
15    + + #lorem(10)
16  + #lorem(10)
```

(a) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
   (i) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
   (ii) (1) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
(b) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

```typ
1   #set enum(numbering: "(a)(i)(1)")
2   #show: el.default-enum-list.with(
3     item-format: (
4       outer: (block.with(fill:
              orange.lighten(80%)),auto),
5       whole: it => {
6         if it.level == 2 {
7           show: strong
8           set text(fill: orange)
9           box(stroke: (left: 3pt +
              orange,rest: .5pt + orange), outset:
              (left: 3pt, top : 2pt), inset: (bottom:
              1pt))[
10            #place(right+bottom)
              [#sym.suit.club.filled]
11            #it.body]
12        } else {
13          it.body
14        }
15      },
```

(a) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.
   (i) **Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.**
   (ii) (1) **Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.** ♣
(b) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.
   (i) **Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.**
   (ii) **Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore.** ♣

```
16    ),
17  )
18 + #lorem(15)
19   + #lorem(15)
20   + + #lorem(15)
21 + #lorem(15)
22   + #lorem(15)
23   + #lorem(15)
```

However, this may disrupt the correct positioning of the list labels. (The main reason is that `itemize` needs to treat the label and the first line of the body as the same paragraph.). It is mainly reflected in:

- When using containers (such as `box` and `block`) to wrap the body content, using the `inset` property of these containers may cause incorrect label positioning.
  - ▸ Since `item-format` is specified as a `function`, we cannot access properties like `inset` (left), making it impossible to correctly handle label positioning.

- Using complex `layout` containers (such as `move`, `rotate`, or `place`) to wrap the body content may also lead to incorrect label layout.

- Although we can ensure that most text formatting (e.g., `highlight`) applied to the body does not affect the nested `label` format, achieving this effect requires a complete refactor of the body elements in each item (to distinguish between labels and bodies).
  - ▸ `lower`, `upper`, `sub`, and `super` are not supported.

    However, you can use `show el.adv.style-format-label` to apply these effects. For example:

```
1   #set enum(numbering: "(a)(i)(1)")                  typ
2   #show: el.default-enum-list.with(
3     item-format: it => {
4       if it.level == 2 {
5         // Equivalent to: if it.level >= 2
6         show el.adv.style-format-label: upper
7         show: highlight
8         it.body
9       } else {
10        it.body
11      }
12    },
13  )
14 + #lorem(10)
15   + #lorem(10)
16   + + #lorem(10)
17 + #lorem(10)
```

(a) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
  (i) LOREM IPSUM DOLOR SIT AMET, CONSECTETUR ADIPISCING ELIT, SED DO.
  (ii) (1) LOREM IPSUM DOLOR SIT AMET, CONSECTETUR ADIPISCING ELIT, SED DO.
(b) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

- Content wrapped in `context` may not be processed correctly. (You can use `layout(_ => {...})` as an alternative.) Similarly, content output in `ref(<...>)` may also not be processed correctly.

- You can use `adv.native-content[...]` to prevent text formatting from being applied to the body. Example:

```typ
1   #set enum(numbering: "(a)(i)(1)")              typ
2   #show: el.default-enum-list.with(
3     item-format: it => {
4       if it.level == 2 {
5         // Equivalent to: if it.level >= 2
6         show el.adv.style-format-label: upper
7         show: highlight
8         it.body
9       } else {
10        it.body
11      }
12    },
13  )
14  + #lorem(10)
15    + #lorem(10)
16
17      before #el.adv.native-content([#lorem(2)])
        after
18    + + #lorem(10)
19  + #lorem(10)
```

(a) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.
  (i) <mark>LOREM IPSUM DOLOR SIT AMET, CONSECTETUR ADIPISCING ELIT, SED DO.</mark>

  <mark>BEFORE</mark> Lorem ipsum. <mark>AFTER</mark>
  (ii) (1) <mark>LOREM IPSUM DOLOR SIT AMET, CONSECTETUR ADIPISCING ELIT, SED DO.</mark>
(b) Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

4 Changelog and All Features

Breaking Change: ⚠️

New Feature: 🆕 for ver0.2.x

If you were using ver0.1.x, please read this section carefully when upgrading to ver0.2.x, as we have made some changes to the configuration methods.

- Two styles of enum-list
  ‣ Default (native typst)
  ‣ Paragraph

- Customize labels and bodies for enums and lists by level and item

  🆕 Allow per-item configuration

  🆕 Enable loop usage of array element values: `LOOP`

  🆕 If a property can be set at both the level and item, it allows passing a function
    – The function format is now standardized as `it => ...`, where you can control it by accessing its properties, typically `it.level` and `it.n`
      • More properties may be added in the future
      ⚠️ **Breaking change**: Properties related to horizontal spacing now follow this format, and additional properties like `it.label-width` (stores the actual maximum width of the label) and `it.e` (represents the current level's construction object: enum or list) are provided
      ⚠️ **Breaking change**: Deprecated `level`, `level-count`, and `list-level`
        ‣ Using these methods to configure enum and list properties may sometimes cause "layout did not converge within 5 attempts"
        ‣ In ver0.1.x, per-level and per-item settings for `enum.numbering` can now be combined with the `numbly` package and the `label-format` method

- 🆕 When using the `*-enum-list` method, you can separately configure enums and lists: `enum-config`, `list-config`
  - ▸ Horizontal spacing settings: `indent`, `body-indent`, `label-indent`, `enum-margin` (`is-full-width`)
  - ▸ Vertical spacing settings: `item-spacing`, `enum-spacing`
  - ▸ Label customization
    - – `text-style` (..args)
    - 🆕 `label-align`
    - 🆕 `label-baseline`
    - 🆕 `label-format`
    - 🆕 `label-width`
  - 🆕 Label alignment between items at different levels: `auto-label-width`
  - ▸ `auto-base-level`
    - ⚠ **Breaking change**: When configuring enums and lists using `*-enum-list`, `*-enum`, or `*-list`, the current level is treated as 1
      - • Version 0.1.x used absolute levels
    - – To maintain compatibility with native behavior, the display of numbering and markers still follows absolute levels. Even if you reconfigure `enum.numbering` and `list.marker` in sublists, their display adheres to absolute level rules
      - • In this case, `auto-base-level` is set to `true`, treating the current level as 1
  - ▸ Body formatting: `hanging-indent`, `line-indent`
    - 🆕 `body-format`:
      - • `text-style`: `style`
      - • Border settings: `outer`, `inner`, `whole`: (`stroke`, `radius`, `outset`, `fill`, `inset`)
    - 📅 Experimental: `item-format`

- Enum numbering references: `elabel` and `config.ref`
  - ⚠ The `ref-enum` method from ver0.1.x will be deprecated in the future. Please use `config.ref` for configuration

- Resume enum (`issue#1`)
  - 🆕 `auto-resuming`
    - – `none`: Disable resume functionality
    - – `auto`: Using the following methods:
      - • `resume`
      - • `resume-label`, `resume-list`
      - • `auto-resume-enum`
      - • `isolated-resume-enum` (independent sublists)
    - – Globally use `true` or (`true`, `false`) etc. to enable functionality; `false` means the functionality does not apply to this level

- List enhancements

- ‣ Enhanced `list.marker`: Allows passing a `function` parameter in the form of `level => n => content` or `level => array`
- 🆕 Terms-like functionality: `item`
- 🆕 Checklist (from the `cheq` package, with minor enhancements and fixes for compatibility with `itemize`)
  - – Configure using `config.checklist`

- Add manual

- Fixes
  - ‣ Alignment of labels and bodies
  - 🆕 Block-level elements displayed on the same line (`issue#4`)
    - – Now better handles blank content in typst, ensuring correct alignment, e.g.,

      ```typ
      1  #show: el.default-enum-list
      2  + #state("").update(none)
      3    + 111
      ```

      ```
      1. 1. 111
      ```

      is now correctly processed
    - – Due to limitations in typst, content within `context` cannot be retrieved in pure typst, making it impossible to correctly process cases like:

      ```typ
      1  #show: el.default-enum-list
      2  + #context {block[#text.fill]}
      ```

      ```
      1.
         luma(0%)
      ```

      - • Temporary solution: Use `layout(_ => {...})` instead of `context {...}`, e.g.,

        ```typ
        1  #show: el.default-enum-list
        2  + #layout(_ => {block[#text.fill]})
        ```

        ```
        1. luma(0%)
        ```

    - – Similarly, output content in `ref(...)` cannot be processed
    - – For block-level elements, our handling is more refined compared to native behavior:
      - • If a block-level element behaves like a paragraph (`par`, `pad`, `block`, `repeat`, `layout`), it is treated as paragraph behavior, ensuring alignment of labels and bodies within the same paragraph
      - • If it is not a paragraph (`block-equation`, `block-raw` `rect`, `table`, `grid`, `stack`, `heading`, `figure`, etc.), it follows native behavior
      - ❓ `align` here is paragraph behavior but cannot be correctly implemented (currently follows native behavior)
  - ‣ Resolved some cases where "layout did not converge within 5 attempts" occurred
    - – However, this may lead to "maximum show rule depth exceeded" when nesting levels increase
    - – Mainly occurs with complex configurations of `label-format` and `body-format`
  - ‣ Other fixes:
  - 💰 Support for `enum.start`
  - ⚠️ **Breaking change**: For compatibility with the native `list.marker` behavior, unlike the methods provided by `itemize`, the level here starts from 0 (see `issue#3`).
  - ☑️ Fixed: Incorrect label display when mixing enums and lists with `*-enum` and `*-list` configurations

- Now, we rewrite all the behaviors of `enum` and `list`, but `*-enum` does not configure list formatting, and vice versa
- ☑ Compatibility with typst 0.14 behavior (`[#6609]` and `[#6242]`)