`Beameri`kZ class manual

Michał Skrzypczak

University of Warsaw

Setup

Setup

1. Download the `beamerikz2.cls` file.

Setup

1. Download the `beamerikz2.cls` file.

2. Place it in the folder where your `*.tex` file is.

Setup

1. Download the `beamerikz2.cls` file.

2. Place it in the folder where your `*.tex` file is.

3. Prepare your slides in the `*.tex` file.

Setup

1. Download the `beamerikz2.cls` file.

2. Place it in the folder where your `*.tex` file is.

3. Prepare your slides in the `*.tex` file.

4. Compile them with `pdflatex` (optimally twice).

Setup

1. Download the `beamerikz2.cls` file.

2. Place it in the folder where your `*.tex` file is.

3. Prepare your slides in the `*.tex` file.

4. Compile them with `pdflatex` (optimally twice).

5. See the result.

Setup

1. Download the `beamerikz2.cls` file.

2. Place it in the folder where your `*.tex` file is.

3. Prepare your slides in the `*.tex` file.

4. Compile them with `pdflatex` (optimally twice).

5. See the result.

6. Be happy*.

Setup

1. Download the `beamerikz2.cls` file.

2. Place it in the folder where your `*.tex` file is.

3. Prepare your slides in the `*.tex` file.

4. Compile them with `pdflatex` (optimally twice).

5. See the result.

6. Be happy*.

* Item **6** is optional.

Minimal working example

```
\documentclass{beamerikz2}              % defines the document to be BeamerikZ

\begin{document}
\begin{bzFrame}                         % the only frame of that document
\bzOn{                                  % content to appear first
    \bzCenter{Welcome to \beamerikz!}  % a centred text
}
\end{bzFrame}
\end{document}
```

Minimal working example

```
\documentclass{beamerikz2}              % defines the document to be BeamerikZ

\begin{document}
\begin{bzFrame}                         % the only frame of that document
\bzOn{                                  % content to appear first
    \bzCenter{Welcome to \beamerikz!}   % a centred text
}
\bzOn{                                  % content to appear second
    \bzCenter{Welcome again, nice to see you!}
    \bzCenter{Hope that you'll stay longer :)}
}
\end{bzFrame}
\end{document}
```

To add more content, create a new `\bzOn{...}` entry, as above.

Minimal working example

```
\documentclass{beamerikz2}               % defines the document to be BeamerikZ

\begin{document}
\begin{bzFrame}                          % the first frame of that document
\bzOn{                                   % content to appear first
    \bzCenter{Welcome to \beamerikz!}    % a centred text
}
\bzOn{                                   % content to appear second
    \bzCenter{Welcome again, nice to see you!}
    \bzCenter{Hope that you'll stay longer :)}
}
\end{bzFrame}

\begin{bzFrame}                          % the second frame
\bzOn{                                   % content to appear first
    \bzCenter{Welcome again!}            % a centred text
}
\end{bzFrame}
\end{document}
```

To add more content, create a new `\bzOn{...}` entry, as above.

Or add a new frame.

Minimal working example

```
\documentclass{beamerikz2}              % defines the document to be BeamerikZ

\begin{document}
\begin{bzFrame}                         % the first frame of that document
\bzOn{                                  % content to appear first
    \bzCenter{Welcome to \beamerikz!}   % a centred text
}
\bzOn{                                  % content to appear second
    \bzCenter{Welcome again, nice to see you!}
    \bzCenter{Hope that you'll stay longer :)}
}
\end{bzFrame}

\begin{bzFrame}                         % the second frame
\bzOn{                                  % content to appear first
    \bzCenter{Welcome again!}           % a centred text
}
\end{bzFrame}
\end{document}
```

To add more content, create a new `\bzOn{...}` entry, as above.

Or add a new frame.

Hint: All your content (like `\bzCenter{...}`) needs to be put inside `\bzOn{...}` entries (or their variants).

Content

Content

Use the following commands to draw your content into the frame:

Content

Use the following commands to draw your content into the frame:

`\bzCenter{...}` for centred text

Content

Use the following commands to draw your content into the frame:

`\bzCenter{...}` for centred text

`\bzLeft{...}` for text aligned next to the left edge

Content

Use the following commands to draw your content into the frame:

`\bzCenter{...}` for centred text

`\bzLeft{...}` for text aligned next to the left edge

`\bzText{...}` for text written normally, from left, with an indent

Content

Use the following commands to draw your content into the frame:

`\bzCenter{...}` for centred text

`\bzLeft{...}` for text aligned next to the left edge

`\bzText{...}` for text written normally, from left, with an indent

`\bzRight{...}` for text aligned to the right

Content

Use the following commands to draw your content into the frame:

`\bzCenter{...}` for centred text

`\bzLeft{...}` for text aligned next to the left edge

`\bzText{...}` for text written normally, from left, with an indent

`\bzRight{...}` for text aligned to the right

`\bzBox{...}` for content that exceeds the standard width of the slide and should wrap to another line (other commands may also allow that, see `bzM` below...)

Item and list

Item and list

There are two ways to present lists of items:

Item and list

There are two ways to present lists of items:

- `\bzItem{...}` for each of the consecutive items

Item and list

There are two ways to present lists of items:

- `\bzItem{...}` for each of the consecutive items
- `\bzItem{...}` ...

Item and list

There are two ways to present lists of items:

- `\bzItem{...}` for each of the consecutive items
- `\bzItem{...}` ...

The big dot is drawn using `\bzItemDot` which you may:

```
\renewcommand{\bzItemDot}{$\circ$}
```

Item and list

There are two ways to present lists of items:

- `\bzItem{...}` for each of the consecutive items
- `\bzItem{...}` . . .

The big dot is drawn using `\bzItemDot` which you may:

```
\renewcommand{\bzItemDot}{$\circ$}
```

You can later reset it using:

```
\zBzItemDot
```

Item and list

There are two ways to present lists of items:

- `\bzItem{...}` for each of the consecutive items
- `\bzItem{...}` ...

The big dot is drawn using `\bzItemDot` which you may:

```
\renewcommand{\bzItemDot}{$\circ$}
```

You can later reset it using:

```
\zBzItemDot
```

Or, if you need a (non-nested) list (the number of last item is stored in `\bzL`):

Item and list

There are two ways to present lists of items:

- `\bzItem{...}` for each of the consecutive items
- `\bzItem{...}` ...

The big dot is drawn using `\bzItemDot` which you may:

```
\renewcommand{\bzItemDot}{$\circ$}
```

You can later reset it using:

```
\zBzItemDot
```

Or, if you need a (non-nested) list (the number of last item is stored in `\bzL`):

1. `\bzList{...}` one entry

Item and list

There are two ways to present lists of items:

- `\bzItem{...}` for each of the consecutive items
- `\bzItem{...}` ...

The big dot is drawn using `\bzItemDot` which you may:

```
\renewcommand{\bzItemDot}{$\circ$}
```

You can later reset it using:

```
\zBzItemDot
```

Or, if you need a (non-nested) list (the number of last item is stored in `\bzL`):

1. `\bzList{...}` one entry
2. `\bzList{...}` another

Item and list

There are two ways to present lists of items:

- `\bzItem{...}` for each of the consecutive items
- `\bzItem{...}` ...

The big dot is drawn using `\bzItemDot` which you may:

```
\renewcommand{\bzItemDot}{$\circ$}
```

You can later reset it using:

```
\zBzItemDot
```

Or, if you need a (non-nested) list (the number of last item is stored in `\bzL`):

1. `\bzList{...}` one entry
2. `\bzList{...}` another

To start your list again, use `\zBzL` (it zeroes `\bzL`), and then:

Item and list

There are two ways to present lists of items:

- `\bzItem{...}` for each of the consecutive items
- `\bzItem{...}` ...

The big dot is drawn using `\bzItemDot` which you may:

```
\renewcommand{\bzItemDot}{$\circ$}
```

You can later reset it using:

```
\zBzItemDot
```

Or, if you need a (non-nested) list (the number of last item is stored in `\bzL`):

1. `\bzList{...}` one entry

2. `\bzList{...}` another

To start your list again, use `\zBzL` (it zeroes `\bzL`), and then:

1. `\bzList{...}` new first entry

Item and list

There are two ways to present lists of items:

- `\bzItem{...}` for each of the consecutive items
- `\bzItem{...}` . . .

The big dot is drawn using `\bzItemDot` which you may:

```
\renewcommand{\bzItemDot}{$\circ$}
```

You can later reset it using:

```
\zBzItemDot
```

Or, if you need a (non-nested) list (the number of last item is stored in `\bzL`):

1. `\bzList{...}` one entry

2. `\bzList{...}` another

To start your list again, use `\zBzL` (it zeroes `\bzL`), and then:

1. `\bzList{...}` new first entry

2. `\bzList{...}` second. . .

Item and list

There are two ways to present lists of items:

- `\bzItem{...}` for each of the consecutive items
- `\bzItem{...}` ...

The big dot is drawn using `\bzItemDot` which you may:

```
\renewcommand{\bzItemDot}{$\circ$}
```

You can later reset it using:

```
\zBzItemDot
```

Or, if you need a (non-nested) list (the number of last item is stored in `\bzL`):

1. `\bzList{...}` one entry
2. `\bzList{...}` another

To start your list again, use `\zBzL` (it zeroes `\bzL`), and then:

1. `\bzList{...}` new first entry
2. `\bzList{...}` second...

Hint: `\bzEvalInt` later in the manual will help you manage `\bzL`

More content. . .

More content...

Theorem (Authors)

`\bzTheorem{authors}{statement}` to state a theorem (the statement can take multiple lines if needed)

More content...

Theorem (Authors)

`\bzTheorem{authors}{statement}` to state a theorem (the statement can take multiple lines if needed)

Hint: you can leave the authors empty if you don't want to specify them.

More content...

Theorem (Authors)

`\bzTheorem{authors}{statement}` to state a theorem (the statement can take multiple lines if needed)

Hint: you can leave the authors empty if you don't want to specify them.

The same syntax works for: **Lemma**, **Fact**, **Conjecture**, **Definition**, **Question**, **Problem**, **Proposition**, **Corollary**, **Exercise**, and **Example**.

More content...

Theorem (Authors)

`\bzTheorem{authors}{statement}` to state a theorem (the statement can take multiple lines if needed)

Hint: you can leave the authors empty if you don't want to specify them.

The same syntax works for: **Lemma**, **Fact**, **Conjecture**, **Definition**, **Question**, **Problem**, **Proposition**, **Corollary**, **Exercise**, and **Example**.

Proof

`\bzProof{...}` to provide a proof (again, it can be spread across multiple lines if needed)

More content...

Theorem (Authors)

`\bzTheorem{authors}{statement}` to state a theorem (the statement can take multiple lines if needed)

Hint: you can leave the authors empty if you don't want to specify them.

The same syntax works for: **Lemma**, **Fact**, **Conjecture**, **Definition**, **Question**, **Problem**, **Proposition**, **Corollary**, **Exercise**, and **Example**.

Proof

`\bzProof{...}` to provide a proof (again, it can be spread across multiple lines if needed)

Finally use `\bzQed` to conclude a proof: ■

More content...

Theorem (Authors)

`\bzTheorem{authors}{statement}` to state a theorem (the statement can take multiple lines if needed)

Hint: you can leave the authors empty if you don't want to specify them.

The same syntax works for: **Lemma**, **Fact**, **Conjecture**, **Definition**, **Question**, **Problem**, **Proposition**, **Corollary**, **Exercise**, and **Example**.

Proof

`\bzProof{...}` to provide a proof (again, it can be spread across multiple lines if needed)

Finally use `\bzQed` to conclude a proof: ■

`\bzLine` draws a line:

Text style

Text style

Use the following commands to emphasise some text:

Text style

Use the following commands to emphasise some text:

- `\bo{...}` to only **boldface** something (it works only with text, no math!)

Text style

Use the following commands to emphasise some text:

- `\bo{...}` to only **boldface** something (it works only with text, no math!)
- `\hl{...}` to highlight some text (including math: $a + b + c$)

Text style

Use the following commands to emphasise some text:

- `\bo{...}` to only **boldface** something (it works only with text, no math!)
- `\hl{...}` to highlight some text (including math: $a + b + c$)
- `\hb{...}` to **highlight&boldface** some text (again, no math!)

Text style

Use the following commands to emphasise some text:

- `\bo{...}` to only **boldface** something (it works only with text, no math!)
- `\hl{...}` to highlight some text (including math: $a + b + c$)
- `\hb{...}` to **highlight&boldface** some text (again, no math!)
- `\el{...}` to emphasize some text (including math: $a + b + c$)

Text style

Use the following commands to emphasise some text:

- `\bo{...}` to only **boldface** something (it works only with text, no math!)
- `\hl{...}` to highlight some text (including math: $a + b + c$)
- `\hb{...}` to **highlight&boldface** some text (again, no math!)
- `\el{...}` to emphasize some text (including math: $a + b + c$)
- `\eb{...}` to **emphasize&boldface** some text (again, no math!)

Text style

Use the following commands to emphasise some text:

- `\bo{...}` to only **boldface** something (it works only with text, no math!)
- `\hl{...}` to highlight some text (including math: $a + b + c$)
- `\hb{...}` to **highlight&boldface** some text (again, no math!)
- `\el{...}` to emphasize some text (including math: $a + b + c$)
- `\eb{...}` to **emphasize&boldface** some text (again, no math!)

The same suffixes with `bz` can be used for the standard content, e.g.:

Text style

Use the following commands to emphasise some text:

- `\bo{...}` to only **boldface** something (it works only with text, no math!)
- `\hl{...}` to highlight some text (including math: $a + b + c$)
- `\hb{...}` to **highlight&boldface** some text (again, no math!)
- `\el{...}` to emphasize some text (including math: $a + b + c$)
- `\eb{...}` to **emphasize&boldface** some text (again, no math!)

The same suffixes with `bz` can be used for the standard content, e.g.:

`\bzLeft[bzHL]{...}` gives a highlighted left-aligned text

Text style

Use the following commands to emphasise some text:

- `\bo{...}` to only **boldface** something (it works only with text, no math!)
- `\hl{...}` to highlight some text (including math: $a + b + c$)
- `\hb{...}` to **highlight&boldface** some text (again, no math!)
- `\el{...}` to emphasize some text (including math: $a + b + c$)
- `\eb{...}` to **emphasize&boldface** some text (again, no math!)

The same suffixes with `bz` can be used for the standard content, e.g.:

`\bzLeft[bzHL]{...}` gives a highlighted left-aligned text

`\bzCenter[bzHB]{...}`

gives a highlighted&boldfaced centred text

Text style

Use the following commands to emphasise some text:

- `\bo{...}` to only **boldface** something (it works only with text, no math!)
- `\hl{...}` to highlight some text (including math: $a + b + c$)
- `\hb{...}` to **highlight&boldface** some text (again, no math!)
- `\el{...}` to emphasize some text (including math: $a + b + c$)
- `\eb{...}` to **emphasize&boldface** some text (again, no math!)

The same suffixes with `bz` can be used for the standard content, e.g.:

`\bzLeft[bzHL]{...}` gives a highlighted left-aligned text

`\bzCenter[bzHB]{...}`

gives a highlighted&boldfaced centred text

`\bzRight[bzBO]{...}` **gives a boldfaced right-aligned text**

Text style

Use the following commands to emphasise some text:

- `\bo{...}` to only **boldface** something (it works only with text, no math!)
- `\hl{...}` to highlight some text (including math: $a + b + c$)
- `\hb{...}` to **highlight&boldface** some text (again, no math!)
- `\el{...}` to emphasize some text (including math: $a + b + c$)
- `\eb{...}` to **emphasize&boldface** some text (again, no math!)

The same suffixes with `bz` can be used for the standard content, e.g.:

`\bzLeft[bzHL]{...}` gives a highlighted left-aligned text

`\bzCenter[bzHB]{...}`

gives a highlighted&boldfaced centred text

`\bzRight[bzBO]{...}` **gives a boldfaced right-aligned text**

Hint: styles also apply to `\bzBox{...}`, `\bzTheorem{...}{...}`, `\bzProof{...}`, etc

Ti*k*Z style

Ti*k*Z style

Each of the following commands (some are explained later):

`\bzLeft{...}`, `\bzText{...}`, `\bzItem{...}`, `\bzList{...}`, `\bzRight{...}`, `\bzCenter{...}`, `\bzNext{...}`,

`\bzPrev{...}`, `\bzBox{...}`, `\bzTheorem{...}{...}`, `\bzProof{...}`

accepts an additional optional first parameter `\bz____[...]{...}`.

Ti*k*Z style

Each of the following commands (some are explained later):

`\bzLeft{...}`, `\bzText{...}`, `\bzItem{...}`, `\bzList{...}`, `\bzRight{...}`, `\bzCenter{...}`, `\bzNext{...}`,

`\bzPrev{...}`, `\bzBox{...}`, `\bzTheorem{...}{...}`, `\bzProof{...}`

accepts an additional optional first parameter `\bz____[...]{...}`.

You can put any Ti*k*Z style (applicable to a node) there.

TikZ style

Each of the following commands (some are explained later):

`\bzLeft{...}`, `\bzText{...}`, `\bzItem{...}`, `\bzList{...}`, `\bzRight{...}`, `\bzCenter{...}`, `\bzNext{...}`,

`\bzPrev{...}`, `\bzBox{...}`, `\bzTheorem{...}{...}`, `\bzProof{...}`

accepts an additional optional first parameter `\bz____[...]{...}`.

You can put any TikZ style (applicable to a node) there.

The styles **`bzBO`**, **`bzHL`**, etc from the previous slide are examples of such styles.

Ti*k*Z style

Each of the following commands (some are explained later):

`\bzLeft{...}`, `\bzText{...}`, `\bzItem{...}`, `\bzList{...}`, `\bzRight{...}`, `\bzCenter{...}`, `\bzNext{...}`,

`\bzPrev{...}`, `\bzBox{...}`, `\bzTheorem{...}{...}`, `\bzProof{...}`

accepts an additional optional first parameter `\bz____[...]{...}`.

You can put any Ti*k*Z style (applicable to a node) there.

The styles **`bzBO`**, **`bzHL`**, etc from the previous slide are examples of such styles.

For instance

```
\bzCenter[scale=1.5, draw, inner sep=1mm, ellipse]{egg}
```

Ti*k*Z style

Each of the following commands (some are explained later):

`\bzLeft{...}`, `\bzText{...}`, `\bzItem{...}`, `\bzList{...}`, `\bzRight{...}`, `\bzCenter{...}`, `\bzNext{...}`,

`\bzPrev{...}`, `\bzBox{...}`, `\bzTheorem{...}{...}`, `\bzProof{...}`

accepts an additional optional first parameter `\bz____[...]{...}`.

You can put any Ti*k*Z style (applicable to a node) there.

The styles **`bzBO`**, **`bzHL`**, etc from the previous slide are examples of such styles.

For instance

```
\bzCenter[scale=1.5, draw, inner sep=1mm, ellipse]{egg}
```

gives:

egg

Titles, authors, references

Titles, authors, references

Control the footline of your frames using these macros (in the preamble):

```
\documentclass{beamerikz}              % defines the document to be BeamerikZ

\sBzTitle{My title}                    % sets the title of the presentation
\sBzAuthor{Me Mysef}                   % sets the author (or authors)

\begin{document}
```

Titles, authors, references

Control the footline of your frames using these macros (in the preamble):

```
\documentclass{beamerikz}                % defines the document to be BeamerikZ

\sBzTitle{My title}                      % sets the title of the presentation
\sBzAuthor{Me Mysef}                     % sets the author (or authors)

\begin{document}
```

You can access these values using `\bzTitle` and `\bzAuthor`.

Titles, authors, references

Control the footline of your frames using these macros (in the preamble):

```
\documentclass{beamerikz}                % defines the document to be BeamerikZ

\sBzTitle{My title}                      % sets the title of the presentation
\sBzAuthor{Me Mysef}                     % sets the author (or authors)

\begin{document}
```

You can access these values using `\bzTitle` and `\bzAuthor`.

You may boldface your title: `\sBzTitle{\bf The Title}`

Titles, authors, references

Control the footline of your frames using these macros (in the preamble):

```
\documentclass{beamerikz}               % defines the document to be BeamerikZ

\sBzTitle{My title}                     % sets the title of the presentation
\sBzAuthor{Me Mysef}                    % sets the author (or authors)

\begin{document}
```

You can access these values using `\bzTitle` and `\bzAuthor`.

You may boldface your title: `\sBzTitle{\bf The Title}`

Or underline the speaker: `\sBzAuthor{A. Guy, \underline{My Self}}`

Titles, authors, references

Control the footline of your frames using these macros (in the preamble):

```
\documentclass{beamerikz}                % defines the document to be BeamerikZ

\sBzTitle{My title}                      % sets the title of the presentation
\sBzAuthor{Me Mysef}                     % sets the author (or authors)

\begin{document}
```

You can access these values using `\bzTitle` and `\bzAuthor`.

You may boldface your title: `\sBzTitle{\bf The Title}`

Or underline the speaker: `\sBzAuthor{A. Guy, \underline{My Self}}`

To refer to others' work, you may use:

```
\bzNames{Reference, Relevance [2013]}
```

(Reference, Relevance [2013])

Colours

Colours

The layout is based on the following colours

- `\bzCtext` used for text and drawings (here it's **bzBlack**)

Colours

The layout is based on the following colours

- `\bzCtext` used for text and drawings (here it's **bzBlack**)
- `\bzCemph` used for titles (here it's **bzDBlue**)

Colours

The layout is based on the following colours

- `\bzCtext` used for text and drawings (here it's **bzBlack**)
- `\bzCemph` used for titles (here it's **bzDBlue**)
- `\bzChigh` used for highlighting (here it's **bzPurple**)

Colours

The layout is based on the following colours

- `\bzCtext` used for text and drawings (here it's **bzBlack**)
- `\bzCemph` used for titles (here it's **bzDBlue**)
- `\bzChigh` used for highlighting (here it's **bzPurple**)
- `\bzCname` used for references (here it's **bzOrange**)

Colours

The layout is based on the following colours

- `\bzCtext` used for text and drawings (here it's **bzBlack**)
- `\bzCemph` used for titles (here it's **bzDBlue**)
- `\bzChigh` used for highlighting (here it's **bzPurple**)
- `\bzCname` used for references (here it's **bzOrange**)
- `\bzCback` used for background (here it's **bzWhite**)

Colours

The layout is based on the following colours

- `\bzCtext` used for text and drawings (here it's **bzBlack**)
- `\bzCemph` used for titles (here it's **bzDBlue**)
- `\bzChigh` used for highlighting (here it's **bzPurple**)
- `\bzCname` used for references (here it's **bzOrange**)
- `\bzCback` used for background (here it's **bzWhite**)

You can change each of these colours, by using (in the preamble):

```
\definecolor{mynewcolor}{RGB}{123,255,0}    % the RGB coordinates are between 0 and 255
\renewcommand{\bzCtext}{mynewcolor}
```

Colours

The layout is based on the following colours

- `\bzCtext` used for text and drawings (here it's **bzBlack**)
- `\bzCemph` used for titles (here it's **bzDBlue**)
- `\bzChigh` used for highlighting (here it's **bzPurple**)
- `\bzCname` used for references (here it's **bzOrange**)
- `\bzCback` used for background (here it's **bzWhite**)

You can change each of these colours, by using (in the preamble):

```
\definecolor{mynewcolor}{RGB}{123,255,0}    % the RGB coordinates are between 0 and 255
\renewcommand{\bzCtext}{mynewcolor}
```

To make your presentation look uniform, use these colours consequently:

```
\bzCenter{\textcolor{\bzCname}{this is also coloured}}
```

this is also coloured

Colours

The layout is based on the following colours

- `\bzCtext` used for text and drawings (here it's **bzBlack**)
- `\bzCemph` used for titles (here it's **bzDBlue**)
- `\bzChigh` used for highlighting (here it's **bzPurple**)
- `\bzCname` used for references (here it's **bzOrange**)
- `\bzCback` used for background (here it's **bzWhite**)

You can change each of these colours, by using (in the preamble):

```
\definecolor{mynewcolor}{RGB}{123,255,0}    % the RGB coordinates are between 0 and 255
\renewcommand{\bzCtext}{mynewcolor}
```

To make your presentation look uniform, use these colours consequently:

```
\bzCenter{\textcolor{\bzCname}{this is also coloured}}
```

this is also coloured

Hint: To choose your colours consistently, I recommend using:

https://coolors.co/

Plain frames

Plain frames

Some frames (like the title one) need to be plain.

Plain frames

Some frames (like the title one) need to be plain.

They go without the footline and are not counted in frame counter.

Plain frames

Some frames (like the title one) need to be plain.

They go without the footline and are not counted in frame counter.

$\Big($like this one...$\Big)$

Plain frames

Some frames (like the title one) need to be plain.

They go without the footline and are not counted in frame counter.

(like this one. . .)

To create such a frame, use

```
\begin{bzPlainFrame}
...
\end{bzPlainFrame}
```

Plain frames

Some frames (like the title one) need to be plain.

They go without the footline and are not counted in frame counter.

(like this one...)

To create such a frame, use

```
\begin{bzPlainFrame}
...
\end{bzPlainFrame}
```

The content of such a frame is generated in the same way as for other frames:

```
\begin{bzPlainFrame}
\bzOn{
    \bzCenter{\textcolor{\bzCtitle}{\bzTitle}}

    \bzCenter{\bzAuthor}
}

\bzOn{
    \bzCenter{\bf READY?}
}
\end{bzPlainFrame}
```

Coordinates

Coordinates

Each slide has the same coordinate system, with $(0,0)$ here:

Coordinates

Each slide has the same coordinate system, with $(0, 0)$ here:

Horizontal axis ranges from -10 here:

Coordinates

Each slide has the same coordinate system, with $(0, 0)$ here:

Horizontal axis ranges from -10 here: to $+10$ here:

Coordinates

Each slide has the same coordinate system, with $(0, 0)$ here:

Horizontal axis ranges from -10 here: to $+10$ here:

Vertical axis decreases down to -14.5 here:

Coordinates

Each slide has the same coordinate system, with $(0, 0)$ here:

Horizontal axis ranges from -10 here: to $+10$ here:

Vertical axis decreases down to -14.5 here:

That means that the point $(4, -6)$ is located here:

Coordinates

Each slide has the same coordinate system, with $(0, 0)$ here:

Horizontal axis ranges from -10 here: to $+10$ here:

Vertical axis decreases down to -14.5 here:

That means that the point $(4, -6)$ is located here:

The content is automatically clipped to the visible area, like those circles.

Coordinates

Each slide has the same coordinate system, with $(0, 0)$ here:

Horizontal axis ranges from -10 here: to $+10$ here:

Vertical axis decreases down to -14.5 here:

That means that the point $(4, -6)$ is located here:

The content is automatically clipped to the visible area, like those circles.

Hint: 1cm = 1unit of this coordinate system :)

Coordinates

Each slide has the same coordinate system, with $(0,0)$ here:

Horizontal axis ranges from -10 here: to $+10$ here:

Vertical axis decreases down to -14.5 here:

That means that the point $(4,-6)$ is located here:

The content is automatically clipped to the visible area, like those circles.

Hint: 1cm = 1unit of this coordinate system :)

Hint: `\bzGrid` shows the coordinate system

Nodes

Nodes

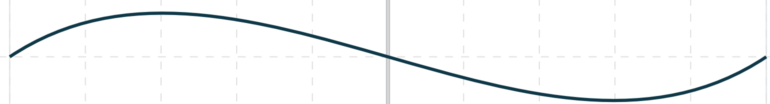
You can use all the Ti*k*Z machinery you like:

```
\bzOn{
    \draw (-5, -3) .. controls ++(+3, +2) and ++(-3, -2) .. (+5, -3);
}
```

Nodes

You can use all the Ti*k*Z machinery you like:

```
\bzOn{
    \draw (-5, -3) .. controls ++(+3, +2) and ++(-3, -2) .. (+5, -3);
}
```



Nodes

You can use all the Ti*k*Z machinery you like:

```
\bzOn{
    \draw (-5, -3) .. controls ++(+3, +2) and ++(-3, -2) .. (+5, -3);
}
```

Nodes

You can use all the Ti*k*Z machinery you like:

```
\bzOn{
    \draw (-5, -3) .. controls ++(+3, +2) and ++(-3, -2) .. (+5, -3);
}
```

However, all this code, including `\node`, `\draw`, `\coordinate`, ...

needs to be contained in `\bzOn{...}`, `\bzIn{from}{to}{...}`, ...!!!

Nodes

You can use all the Ti*k*Z machinery you like:

```
\bzOn{
    \draw (-5, -3) .. controls ++(+3, +2) and ++(-3, -2) .. (+5, -3);
}
```

However, all this code, including `\node`, `\draw`, `\coordinate`, ...

needs to be contained in `\bzOn{...}`, `\bzIn{from}{to}{...}`, ...!!!

To write text, you may use styles **`bzR`**, **`bzC`**, and **`bzL`**:

```
\bzOn{
    \node[bzL, draw] at (-8, -9) {aligns to left};
    \node[bzC, draw] at (+0, -9) {aligns to centre};
    \node[bzR, draw] at (+8, -9) {aligns to right};
}
```

Nodes

You can use all the Ti*k*Z machinery you like:

```
\bzOn{
    \draw (-5, -3) .. controls ++(+3, +2) and ++(-3, -2) .. (+5, -3);
}
```

However, all this code, including `\node`, `\draw`, `\coordinate`, . . .

needs to be contained in `\bzOn{...}`, `\bzIn{from}{to}{...}`, . . . !!!

To write text, you may use styles **`bzR`**, **`bzC`**, and **`bzL`**:

```
\bzOn{
    \node[bzL, draw] at (-8, -9) {aligns to left};
    \node[bzC, draw] at (+0, -9) {aligns to centre};
    \node[bzR, draw] at (+8, -9) {aligns to right};
}
```

aligns to left   lies centrally   aligns to right

Hint: in fact `\bzLeft{...}`, `\bzCenter{...}`, . . . are based on these styles!

Nodes

You can use all the Ti*k*Z machinery you like:

```
\bzOn{
    \draw (-5, -3) .. controls ++(+3, +2) and ++(-3, -2) .. (+5, -3);
}
```

However, all this code, including `\node`, `\draw`, `\coordinate`, ...

needs to be contained in `\bzOn{...}`, `\bzIn{from}{to}{...}`, ...!!!

To write text, you may use styles **`bzR`**, **`bzC`**, and **`bzL`**:

```
\bzOn{
    \node[bzL, draw] at (-8, -9) {aligns to left};
    \node[bzC, draw] at (+0, -9) {aligns to centre};
    \node[bzR, draw] at (+8, -9) {aligns to right};
}
```

aligns to left   lies centrally   aligns to right

Nodes

You can use all the Ti*k*Z machinery you like:

```
\bzOn{
    \draw (-5, -3) .. controls ++(+3, +2) and ++(-3, -2) .. (+5, -3);
}
```

However, all this code, including `\node`, `\draw`, `\coordinate`, . . .

needs to be contained in `\bzOn{...}`, `\bzIn{from}{to}{...}`, . . . !!!

To write text, you may use styles **`bzR`**, **`bzC`**, and **`bzL`**:

```
\bzOn{
    \node[bzL, draw] at (-8, -9) {aligns to left};
    \node[bzC, draw] at (+0, -9) {aligns to centre};
    \node[bzR, draw] at (+8, -9) {aligns to right};
}
```

aligns to left   lies centrally   aligns to right

What is important, is that the baseline of text is kept:

```
\bzOn{
    \node[bzL, draw] at (-8.5, -14) {car};
    \node[bzL, draw] at (-7.0, -14) {trunk};
    \node[bzL, draw] at (-5.0, -14) {yummy};
    \node[bzL, draw] at (-2.5, -14) {Tommy};
}
```

car   trunk   yummy   Tommy

Z-index

Z-index

You can place some content under or over the existing content.

Z-index

You can place some content under or over the existing content.

The first optional argument of `\bzOn[z]{...}`, `\bzIn[z]{}{}{...}`, ... controls the z-index:

Z-index

You can place some content under or over the existing content.

The first optional argument of `\bzOn[z]{...}`, `\bzIn[z]{}{}{...}`, ... controls the z-index:

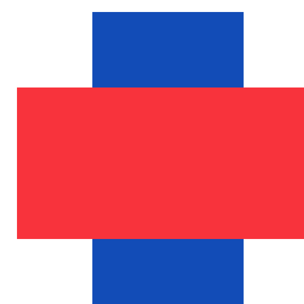
```
\bzOn{
    \path[fill=bzRed] (5, -10) rectangle (9, -12);
}

\bzOn[-1]{
    \path[fill=bzBlue] (6, -9) rectangle (8, -13);
}
```

Z-index

You can place some content under or over the existing content.

The first optional argument of `\bzOn[z]{...}`, `\bzIn[z]{}{}{...}`, ... controls the z-index:

```
\bzOn{
    \path[fill=bzRed] (5, -10) rectangle (9, -12);
}

\bzOn[-1]{
    \path[fill=bzBlue] (6, -9) rectangle (8, -13);
}
```



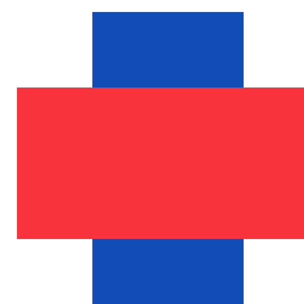
Z-index

You can place some content under or over the existing content.

The first optional argument of `\bzOn[z]{...}`, `\bzIn[z]{}{}{...}`, ... controls the z-index:

```
\bzOn{
    \path[fill=bzRed] (5, -10) rectangle (9, -12);
}

\bzOn[-1]{
    \path[fill=bzBlue] (6, -9) rectangle (8, -13);
}
```

Hint: The possible values are 1, 0, and -1, with 0 default.



Vertical spacing

Vertical spacing

Each consecutive command like `\bzLeft{...}` spreads text vertically…

…like these two.

Vertical spacing

Each consecutive command like `\bzLeft{...}` spreads text vertically...

...like these two.

The vertical step is stored in the counter `\bzHStep`.

By default, its value is 1.0.

Vertical spacing

Each consecutive command like `\bzLeft{...}` spreads text vertically...

...like these two.

The vertical step is stored in the counter `\bzHStep`.

By default, its value is 1.0.

To control the vertical spacing between the lines of your text, use:

```
\sBzHStep{2.0}
```

Vertical spacing

Each consecutive command like `\bzLeft{...}` spreads text vertically...

...like these two.

The vertical step is stored in the counter `\bzHStep`.

By default, its value is 1.0.

To control the vertical spacing between the lines of your text, use:

```
\sBzHStep{2.0}
```

Which changes the spacing

as seen

here!

Vertical spacing

Each consecutive command like `\bzLeft{...}` spreads text vertically…

…like these two.

The vertical step is stored in the counter `\bzHStep`.

By default, its value is 1.0.

To control the vertical spacing between the lines of your text, use:

```
\sBzHStep{2.0}
```

Which changes the spacing

as seen

here!

`\zBzHStep` is equivalent to `\sBzHStep{1.0}`, i.e. it restores the value to 1.0.

Vertical alignment

Vertical alignment

There is a counter called `\bzH`, controlling the *height* of the content.

Vertical alignment

There is a counter called `\bzH`, controlling the *height* of the content.

Its value is now -2.0

Vertical alignment

There is a counter called `\bzH`, controlling the *height* of the content.

Its value is now -2.0

and now -3.0

Vertical alignment

There is a counter called `\bzH`, controlling the *height* of the content.

Its value is now -2.0

and now -3.0

Each new line decreases it by `\bzHStep` (usually equal 1.0) so that now it's -4.0.

Vertical alignment

There is a counter called `\bzH`, controlling the *height* of the content.

Its value is now -2.0

and now -3.0

Each new line decreases it by `\bzHStep` (usually equal 1.0) so that now it's -4.0.

You should place **all** your content relatively to `\bzH`(!):

```
\draw (-4, \bzH) rectangle (+4, \bzH-1);
```

`\bzH` $= -7.0$

Vertical alignment

There is a counter called `\bzH`, controlling the *height* of the content.

Its value is now -2.0

and now -3.0

Each new line decreases it by `\bzHStep` (usually equal 1.0) so that now it's -4.0.

You should place **all** your content relatively to `\bzH`(!):

```
\draw (-4, \bzH) rectangle (+4, \bzH-1);
```

`\bzH` $= -7.0$

To get some vertical space, use:

`\bzH` $= -9.0$

`\iBzH`

`\bzH` $= -10.0$

Vertical alignment

There is a counter called `\bzH`, controlling the *height* of the content.

Its value is now -2.0

and now -3.0

Each new line decreases it by `\bzHStep` (usually equal 1.0) so that now it's -4.0.

You should place **all** your content relatively to `\bzH`(!):

```
\draw (-4, \bzH) rectangle (+4, \bzH-1);
```

`\bzH` $= -7.0$

To get some vertical space, use: `\bzH` $= -9.0$

`\iBzH` `\bzH` $= -10.0$

That *increases* (in fact decreases. . .) `\bzH` by `\bzHStep` `\bzH` $= -11.0$

Vertical alignment

There is a counter called `\bzH`, controlling the *height* of the content.

Its value is now -2.0

and now -3.0

Each new line decreases it by `\bzHStep` (usually equal 1.0) so that now it's -4.0.

You should place **all** your content relatively to `\bzH`(!):

```
\draw (-4, \bzH) rectangle (+4, \bzH-1);
```

`\bzH` $= -7.0$

To get some vertical space, use: `\bzH` $= -9.0$

`\iBzH` `\bzH` $= -10.0$

That *increases* (in fact decreases. . .) `\bzH` by `\bzHStep` `\bzH` $= -11.0$

You can control the amount of space by an optional parameter: `\bzH` $= -12.0$

`\iBzH[0.8]` `\bzH` $= -13.0$

Vertical alignment

There is a counter called `\bzH`, controlling the *height* of the content.

Its value is now -2.0

and now -3.0

Each new line decreases it by `\bzHStep` (usually equal 1.0) so that now it's -4.0.

You should place **all** your content relatively to `\bzH`(!):

```
\draw (-4, \bzH) rectangle (+4, \bzH-1);
```

`\bzH` $= -7.0$

To get some vertical space, use: `\bzH` $= -9.0$

`\iBzH` `\bzH` $= -10.0$

That *increases* (in fact decreases. . .) `\bzH` by `\bzHStep` `\bzH` $= -11.0$

You can control the amount of space by an optional parameter: `\bzH` $= -12.0$

`\iBzH[0.8]` `\bzH` $= -13.0$

Which *increases* (in fact decreases. . .) `\bzH` by 0.8 `\bzH` $= -13.8$

Vertical alignment — summary

Vertical alignment — summary

Available commands:

Vertical alignment — summary

Available commands:

- `\bzH` — the counter

Vertical alignment — summary

Available commands:

- `\bzH` — the counter
- `\sBzH{4.0}` — set the value of the counter

Vertical alignment — summary

Available commands:

- `\bzH` — the counter
- `\sBzH{4.0}` — set the value of the counter
- `\iBzH` — go down one line (i.e. decrease `\bzH` by `\bzHStep`, 1.0 by default)

Vertical alignment — summary

Available commands:

- `\bzH` — the counter
- `\sBzH{4.0}` — set the value of the counter
- `\iBzH` — go down one line (i.e. decrease `\bzH` by `\bzHStep`, 1.0 by default)
- `\iBzH[0.8]` — go down by 0.8

Vertical alignment — summary

Available commands:

- `\bzH` — the counter
- `\sBzH{4.0}` — set the value of the counter
- `\iBzH` — go down one line (i.e. decrease `\bzH` by `\bzHStep`, 1.0 by default)
- `\iBzH[0.8]` — go down by 0.8
- `\dBzH` — go up one line (i.e. *increase* `\bzH` by `\bzHStep`)

Vertical alignment — summary

Available commands:

- `\bzH` — the counter
- `\sBzH{4.0}` — set the value of the counter
- `\iBzH` — go down one line (i.e. decrease `\bzH` by `\bzHStep`, 1.0 by default)
- `\iBzH[0.8]` — go down by 0.8
- `\dBzH` — go up one line (i.e. *increase* `\bzH` by `\bzHStep`)
- `\dBzH[1.3]` — go up 1.3 of a line

Vertical alignment — summary

Available commands:

- `\bzH` — the counter
- `\sBzH{4.0}` — set the value of the counter
- `\iBzH` — go down one line (i.e. decrease `\bzH` by `\bzHStep`, 1.0 by default)
- `\iBzH[0.8]` — go down by 0.8
- `\dBzH` — go up one line (i.e. *increase* `\bzH` by `\bzHStep`)
- `\dBzH[1.3]` — go up 1.3 of a line
- `\zBzH` — reset `\bzH` to 0 (i.e. *zero* `\bzH`)

Vertical alignment — summary

Available commands:

- `\bzH` — the counter
- `\sBzH{4.0}` — set the value of the counter
- `\iBzH` — go down one line (i.e. decrease `\bzH` by `\bzHStep`, 1.0 by default)
- `\iBzH[0.8]` — go down by 0.8
- `\dBzH` — go up one line (i.e. *increase* `\bzH` by `\bzHStep`)
- `\dBzH[1.3]` — go up 1.3 of a line
- `\zBzH` — reset `\bzH` to 0 (i.e. *zero* `\bzH`)

All these commands accept arithmetic, like:

```
\newcommand{\x}{1.6}
\iBzH[(\x+1)*0.5+2]
```

Vertical alignment — summary

Available commands:

- `\bzH` — the counter
- `\sBzH{4.0}` — set the value of the counter
- `\iBzH` — go down one line (i.e. decrease `\bzH` by `\bzHStep`, 1.0 by default)
- `\iBzH[0.8]` — go down by 0.8
- `\dBzH` — go up one line (i.e. *increase* `\bzH` by `\bzHStep`)
- `\dBzH[1.3]` — go up 1.3 of a line
- `\zBzH` — reset `\bzH` to 0 (i.e. *zero* `\bzH`)

All these commands accept arithmetic, like:

```
\newcommand{\x}{1.6}
\iBzH[(\x+1)*0.5+2]
```

Hint: you may put such commands both **inside** and **outside** of `\bzOn{...}`

(all the commands above the given one do affect it)

Horizontal alignment

Horizontal alignment

There is a counter called `\bzI`, controlling the *indentation* of the content.

Horizontal alignment

There is a counter called `\bzI`, controlling the *indentation* of the content.

Using `\bzLeft{...}` when `\bzI` $= -10.0$

Horizontal alignment

There is a counter called `\bzI`, controlling the *indentation* of the content.

Using `\bzLeft{...}` when `\bzI` $= -10.0$

now `\bzLeft{...}` again but `\bzI` $= -5.0$

Horizontal alignment

There is a counter called `\bzI`, controlling the *indentation* of the content.

Using `\bzLeft{...}` when `\bzI` $= -10.0$

now `\bzLeft{...}` again but `\bzI` $= -5.0$

and now `\bzI` $= 3.0$

Horizontal alignment

There is a counter called `\bzI`, controlling the *indentation* of the content.

Using `\bzLeft{...}` when `\bzI` $= -10.0$

now `\bzLeft{...}` again but `\bzI` $= -5.0$

and now `\bzI` $= 3.0$

`\bzText{...}` is always 1 to the right (now `\bzI` $= -10$ again)

Horizontal alignment

There is a counter called `\bzI`, controlling the *indentation* of the content.

Using `\bzLeft{...}` when `\bzI` $= -10.0$

now `\bzLeft{...}` again but `\bzI` $= -5.0$

and now `\bzI` $= 3.0$

`\bzText{...}` is always 1 to the right (now `\bzI` $= -10$ again)

`\bzCenter{...}` is not affected

Horizontal alignment

There is a counter called `\bzI`, controlling the *indentation* of the content.

Using `\bzLeft{...}` when `\bzI` $= -10.0$

now `\bzLeft{...}` again but `\bzI` $= -5.0$

and now `\bzI` $= 3.0$

`\bzText{...}` is always 1 to the right (now `\bzI` $= -10$ again)

`\bzCenter{...}` is not affected

`\bzRight{...}` is affected by $-$`\bzI` (now `\bzI` $= -5.0$)

Horizontal alignment

There is a counter called `\bzI`, controlling the *indentation* of the content.

Using `\bzLeft{...}` when `\bzI` $= -10.0$

now `\bzLeft{...}` again but `\bzI` $= -5.0$

and now `\bzI` $= 3.0$

`\bzText{...}` is always 1 to the right (now `\bzI` $= -10$ again)

`\bzCenter{...}` is not affected

`\bzRight{...}` is affected by $-$`\bzI` (now `\bzI` $= -5.0$)

Available commands:

Horizontal alignment

There is a counter called `\bzI`, controlling the *indentation* of the content.

Using `\bzLeft{...}` when `\bzI` $= -10.0$

now `\bzLeft{...}` again but `\bzI` $= -5.0$

and now `\bzI` $= 3.0$

`\bzText{...}` is always 1 to the right (now `\bzI` $= -10$ again)

`\bzCenter{...}` is not affected

`\bzRight{...}` is affected by $-$`\bzI` (now `\bzI` $= -5.0$)

Available commands:

- `\sBzI{5.0}` — set the value of the counter

Horizontal alignment

There is a counter called `\bzI`, controlling the *indentation* of the content.

Using `\bzLeft{...}` when `\bzI` $= -10.0$

now `\bzLeft{...}` again but `\bzI` $= -5.0$

and now `\bzI` $= 3.0$

`\bzText{...}` is always 1 to the right (now `\bzI` $= -10$ again)

`\bzCenter{...}` is not affected

`\bzRight{...}` is affected by $-$`\bzI` (now `\bzI` $= -5.0$)

Available commands:

- `\sBzI{5.0}` — set the value of the counter
- `\iBzI` — increase indent by 1 (+parametrised `\iBzI[2.8]`)

Horizontal alignment

There is a counter called `\bzI`, controlling the *indentation* of the content.

Using `\bzLeft{...}` when `\bzI` $= -10.0$

now `\bzLeft{...}` again but `\bzI` $= -5.0$

and now `\bzI` $= 3.0$

`\bzText{...}` is always 1 to the right (now `\bzI` $= -10$ again)

`\bzCenter{...}` is not affected

`\bzRight{...}` is affected by $-$`\bzI` (now `\bzI` $= -5.0$)

Available commands:

- `\sBzI{5.0}` — set the value of the counter
- `\iBzI` — increase indent by 1 (+parametrised `\iBzI[2.8]`)
- `\dBzI` — decrease indent by 1 (+parametrised `\dBzI[1.3]`)

Horizontal alignment

There is a counter called `\bzI`, controlling the *indentation* of the content.

Using `\bzLeft{...}` when `\bzI` $= -10.0$

now `\bzLeft{...}` again but `\bzI` $= -5.0$

and now `\bzI` $= 3.0$

`\bzText{...}` is always 1 to the right (now `\bzI` $= -10$ again)

`\bzCenter{...}` is not affected

`\bzRight{...}` is affected by $-$`\bzI` (now `\bzI` $= -5.0$)

Available commands:

- `\sBzI{5.0}` — set the value of the counter
- `\iBzI` — increase indent by 1 (+parametrised `\iBzI[2.8]`)
- `\dBzI` — decrease indent by 1 (+parametrised `\dBzI[1.3]`)
- `\zBzI` — reset `\bzI` to 0

Horizontal alignment

There is a counter called `\bzI`, controlling the *indentation* of the content.

Using `\bzLeft{...}` when `\bzI` $= -10.0$

now `\bzLeft{...}` again but `\bzI` $= -5.0$

and now `\bzI` $= 3.0$

`\bzText{...}` is always 1 to the right (now `\bzI` $= -10$ again)

`\bzCenter{...}` is not affected

`\bzRight{...}` is affected by $-$`\bzI` (now `\bzI` $= -5.0$)

Available commands:

- `\sBzI{5.0}` — set the value of the counter
- `\iBzI` — increase indent by 1 (+parametrised `\iBzI[2.8]`)
- `\dBzI` — decrease indent by 1 (+parametrised `\dBzI[1.3]`)
- `\zBzI` — reset `\bzI` to 0

Hint: you may modify `\bzI` **outside** `\bzOn{...}`

Multiline nodes

Multiline nodes

Standard `\bzLeft{...}`, etc commands do not break lines automatically, as here where

Multiline nodes

Standard `\bzLeft{...}`, etc commands do not break lines automatically, as here where

You can manually break lines using the newline syntax `\\`
however, the node is then aligned vertically to the **last** line of text

Multiline nodes

Standard `\bzLeft{...}`, etc commands do not break lines automatically, as here where

You can manually break lines using the newline syntax `\\`
however, the node is then aligned vertically to the **last** line of text

To allow automatic line breaks, use the multiline style **`bzM`**:

```
\iBzI[3.5]
\bzLeft[bzM]{Text that is longer than a line and the style bzM brakes it automatically}
```

Text which is longer than a line and the style **`bzM`** brakes it automatically

Multiline nodes

Standard `\bzLeft{...}`, etc commands do not break lines automatically, as here where

You can manually break lines using the newline syntax `\\`
however, the node is then aligned vertically to the **last** line of text

To allow automatic line breaks, use the multiline style **`bzM`**:

```
\iBzI[3.5]
\bzLeft[bzM]{Text that is longer than a line and the style bzM brakes it automatically}
```

Text which is longer than a line and the style **`bzM`** brakes it automatically

The same works for other nodes, like `\bzRight{...}`, `\bzCenter{...}`, etc

Multiline nodes

Standard `\bzLeft{...}`, etc commands do not break lines automatically, as here where

You can manually break lines using the newline syntax `\\`
however, the node is then aligned vertically to the **last** line of text

To allow automatic line breaks, use the multiline style **`bzM`**:

```
\iBzI[3.5]
\bzLeft[bzM]{Text that is longer than a line and the style bzM brakes it automatically}
```

Text which is longer than a line and the style **`bzM`** brakes it automatically

The same works for other nodes, like `\bzRight{...}`, `\bzCenter{...}`, etc

Hint: The vertical spacing between the lines is controlled by `\bzHStep`!

Multiline nodes

Standard `\bzLeft{...}`, etc commands do not break lines automatically, as here where

You can manually break lines using the newline syntax `\\`
however, the node is then aligned vertically to the **last** line of text

To allow automatic line breaks, use the multiline style **`bzM`**:

```
\iBzI[3.5]
\bzLeft[bzM]{Text that is longer than a line and the style bzM brakes it automatically}
```

Text which is longer than a line and the style **`bzM`** brakes it automatically

The same works for other nodes, like `\bzRight{...}`, `\bzCenter{...}`, etc

Hint: The vertical spacing between the lines is controlled by `\bzHStep`!

Hint: BeamerikZ is not aware of the actual height of the node, so increase `\bzH` manually!

Multiline nodes

Standard `\bzLeft{...}`, etc commands do not break lines automatically, as here where

You can manually break lines using the newline syntax `\\`
however, the node is then aligned vertically to the **last** line of text

To allow automatic line breaks, use the multiline style **`bzM`**:

```
\iBzI[3.5]
\bzLeft[bzM]{Text that is longer than a line and the style bzM brakes it automatically}
```

Text which is longer than a line and the style **`bzM`** brakes it automatically

The same works for other nodes, like `\bzRight{...}`, `\bzCenter{...}`, etc

Hint: The vertical spacing between the lines is controlled by `\bzHStep`!

Hint: `Beamer`i*k*Z is not aware of the actual height of the node, so increase `\bzH` manually!

Hint: **`bzM`** aligns vertically to the first line of text and allows explicit newlines `\\`

Multiline nodes — width

Multiline nodes — width

By default, the width of the text is adjusted so that it spreads until the end of the working area $(-10, \ldots, +10)$ (or $(-13.0, \ldots, +13.0)$ in `wide` mode).

In `\bzCenter{...}`, margins of width given by `\bzI` are left on both sides.

Multiline nodes — width

By default, the width of the text is adjusted so that it spreads until the end of the working area $(-10, \dots, +10)$ (or $(-13.0, \dots, +13.0)$ in `wide` mode).

In `\bzCenter{...}`, margins of width given by `\bzI` are left on both sides.

Text aligned to left by `\bzLeft{...}` and being broken to a new line.

Text aligned to right by `\bzRight{...}` and being broken to a new line.

Text centered via `\bzCenter{...}` and broken to new lines.

Multiline nodes — width

By default, the width of the text is adjusted so that it spreads until the end of the working area $(-10, \ldots, +10)$ (or $(-13.0, \ldots, +13.0)$ in `wide` mode).

In `\bzCenter{...}`, margins of width given by `\bzI` are left on both sides.

Text aligned to left by `\bzLeft{...}` and being broken to a new line.

Text aligned to right by `\bzRight{...}` and being broken to a new line.

Text centered via `\bzCenter{...}` and broken to new lines.

You can set your own width of text, by using `bzM=<num>` like here

```
\bzLeft[bzM=5.5, draw]{Text of width 5.5 with automatic line breaks}
```

Text of width 5.5 with automatic line breaks

Mathematics

Mathematics

- Standard math mode `$...$` is allowed everywhere, like here $x = 1$.

Mathematics

- Standard math mode `$...$` is allowed everywhere, like here $x = 1$.
- To use displayed mathematics `\[...\]`, the option `bzM` is required:

```
\bzCenter[bzM]{\[\sum_{i=1}^n i=\frac{n\cdot(n+1)}{2}\]}
```

$$\sum_{i=1}^{n} i = \frac{n \cdot (n+1)}{2}$$

…but `\[...\]` introduces additional vertical space. **Better avoid it.**

Similarly `align*` is also not suggested…

Mathematics

- Standard math mode `$...$` is allowed everywhere, like here $x = 1$.
- To use displayed mathematics `\[...\]`, the option `bzM` is required:

```
\bzCenter[bzM]{\[\sum_{i=1}^n i=\frac{n\cdot(n+1)}{2}\]}
```

$$\sum_{i=1}^{n} i = \frac{n \cdot (n+1)}{2}$$

…but `\[...\]` introduces additional vertical space. **Better avoid it.**

Similarly `align*` is also not suggested…

- Instead, use `\bzEq{...}` as below:

```
\bzRight{$\bzEq{\sum_{i=1}^n i=\frac{n\cdot(n+1)}{2}}$}
```

$$\sum_{i=1}^{n} i = \frac{n \cdot (n+1)}{2}$$

Mathematics

- Standard math mode `$...$` is allowed everywhere, like here $x = 1$.
- To use displayed mathematics `\[...\]`, the option `bzM` is required:

```
\bzCenter[bzM]{\[\sum_{i=1}^n i=\frac{n\cdot(n+1)}{2}\]}
```

$$\sum_{i=1}^{n} i = \frac{n \cdot (n+1)}{2}$$

...but `\[...\]` introduces additional vertical space. **Better avoid it.**

Similarly `align*` is also not suggested...

- Instead, use `\bzEq{...}` as below:

```
\bzRight{$\bzEq{\sum_{i=1}^n i=\frac{n\cdot(n+1)}{2}}$}
```

$$\sum_{i=1}^{n} i = \frac{n \cdot (n+1)}{2}$$

- It allows formatting as in `align*`:

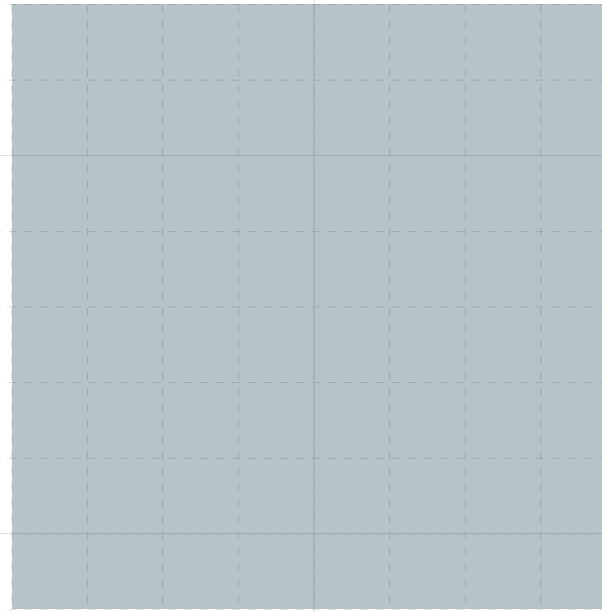
```
\bzCenter{$\bzEq{
  x_0 &= x_1 = 0 &\quad\text{(1)}\\
  x_{n+2} &= x_{n+1}+x_n &\quad\text{(2)}
}$}
```

$$\begin{aligned} x_0 &= x_1 = 0 &\quad\text{(1)}\\ x_{n+2} &= x_{n+1}+x_n &\quad\text{(2)} \end{aligned}$$

Graphics

Graphics

To include graphics, you may use:



Graphics

To include graphics, you may use:

```
\node[bzG] at (5,-7) {\includegraphics[width=6cm]{logo.png}};
```

Graphics

To include graphics, you may use:

```
\node[bzG] at (5,-7) {\includegraphics[width=6cm]{logo.png}};
```

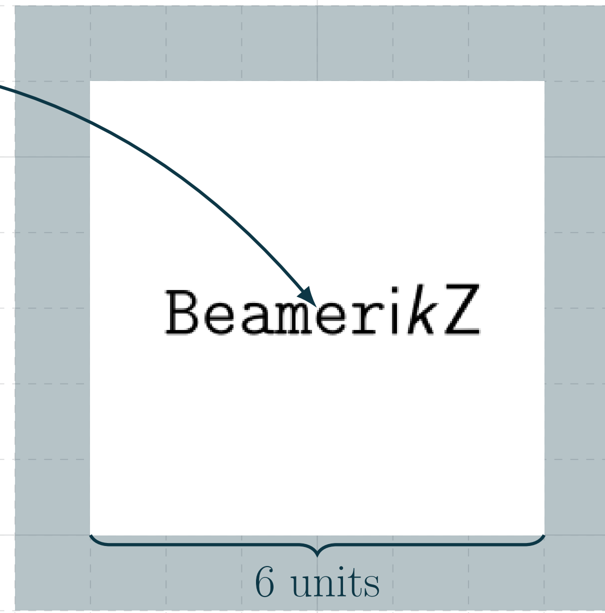
- The centre is at `(5,-7)`

Graphics

To include graphics, you may use:

```
\node[bzG] at (5,-7) {\includegraphics[width=6cm]{logo.png}};
```

- The centre is at `(5,-7)`
- Width is `6cm` = 6 units

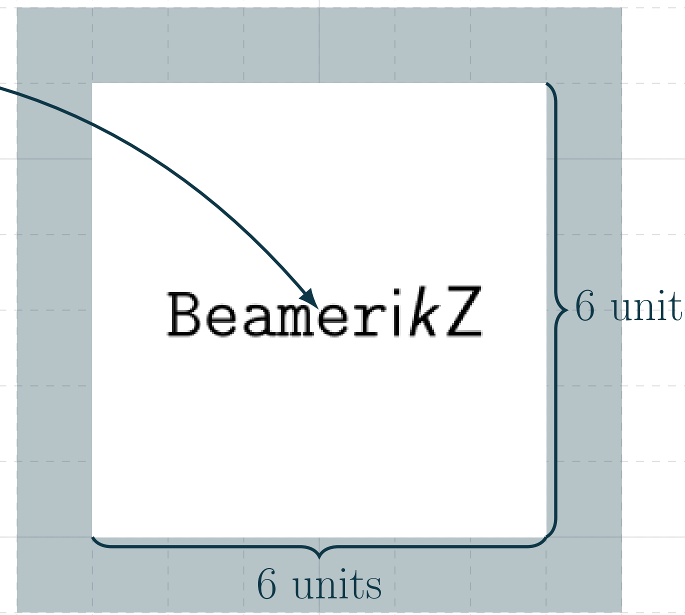


Graphics

To include graphics, you may use:

```
\node[bzG] at (5,-7) {\includegraphics[width=6cm]{logo.png}};
```

- The centre is at `(5,-7)`
- Width is `6cm` = 6 units
- Height is proportional to the ratio

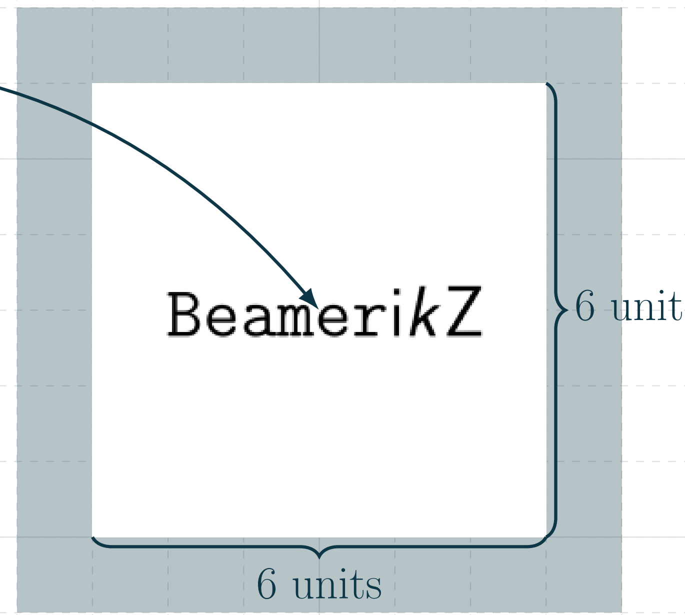


Graphics

To include graphics, you may use:

```
\node[bzG] at (5,-7) {\includegraphics[width=6cm]{logo.png}};
```

- The centre is at `(5,-7)`
- Width is `6cm` = 6 units
- Height is proportional to the ratio
- There are no margins or spacing



Arithmetic

Arithmetic

To perform complex computations, you may use the following two functions:

```
\bzEvalInt{\x}{7*(4+\bzH)+0.25}
\bzEval{\y}{6/(2-\bzH)}
```

Arithmetic

To perform complex computations, you may use the following two functions:

```
\bzEvalInt{\x}{7*(4+\bzH)+0.25}
\bzEval{\y}{6/(2-\bzH)}
```

`\bzH` $= -2.0$

Arithmetic

To perform complex computations, you may use the following two functions:

```
\bzEvalInt{\x}{7*(4+\bzH)+0.25}
\bzEval{\y}{6/(2-\bzH)}
```

`\bzH` $= -2.0$

They compute **immediately**, using **current** values of the variables.

Arithmetic

To perform complex computations, you may use the following two functions:

```
\bzEvalInt{\x}{7*(4+\bzH)+0.25}
\bzEval{\y}{6/(2-\bzH)}
```

`\bzH` $= -2.0$

They compute **immediately**, using **current** values of the variables.

The first stores the value as a rounded integer, the second is floating point.

Arithmetic

To perform complex computations, you may use the following two functions:

```
\bzEvalInt{\x}{7*(4+\bzH)+0.25}
\bzEval{\y}{6/(2-\bzH)}
```

`\bzH` $= -2.0$

They compute **immediately**, using **current** values of the variables.

The first stores the value as a rounded integer, the second is floating point.

Thus, we get `\x` $= 14$ and `\y` $= 1.5$, even though now `\bzH` $= -5.5$.

Arithmetic

To perform complex computations, you may use the following two functions:

```
\bzEvalInt{\x}{7*(4+\bzH)+0.25}
\bzEval{\y}{6/(2-\bzH)}
```

`\bzH` $= -2.0$

They compute **immediately**, using **current** values of the variables.

The first stores the value as a rounded integer, the second is floating point.

Thus, we get `\x` $= 14$ and `\y` $= 1.5$, even though now `\bzH` $= -5.5$.

Hint: do not confuse `\bzEvalInt` with `\bzEval`!

Arithmetic

To perform complex computations, you may use the following two functions:

```
\bzEvalInt{\x}{7*(4+\bzH)+0.25}
\bzEval{\y}{6/(2-\bzH)}
```

`\bzH` $= -2.0$

They compute **immediately**, using **current** values of the variables.

The first stores the value as a rounded integer, the second is floating point.

Thus, we get `\x` $= 14$ and `\y` $= 1.5$, even though now `\bzH` $= -5.5$.

Hint: do not confuse `\bzEvalInt` with `\bzEval`!

You may use `\bzEval{\curH}{\bzH}` to store in the variable `\curH` the current value of `\bzH` for later use!

Arithmetic

To perform complex computations, you may use the following two functions:

```
\bzEvalInt{\x}{7*(4+\bzH)+0.25}
\bzEval{\y}{6/(2-\bzH)}
```

`\bzH` $= -2.0$

They compute **immediately**, using **current** values of the variables.

The first stores the value as a rounded integer, the second is floating point.

Thus, we get `\x` $= 14$ and `\y` $= 1.5$, even though now `\bzH` $= -5.5$.

Hint: do not confuse `\bzEvalInt` with `\bzEval`!

You may use `\bzEval{\curH}{\bzH}` to store in the variable `\curH`

the current value of `\bzH` for later use!

And then use:

```
\draw (0, \bzH-1) edge[Circle-Latex, bend right=40] (9, \curH);
```

to use that value!

Arithmetic and scripting

Arithmetic and scripting

To avoid syntactic problems, use `\bzEval` to give names to parameters of your macros.

Arithmetic and scripting

To avoid syntactic problems, use `\bzEval` to give names to parameters of your macros.

```
\newcommand{\putDot}[2]{
  \bzEval{\x}{#1}
  \bzEval{\r}{#2}
  \node[draw, circle, rotate=\r] at (\x, \bzH) {Hi!};
}

\bzOn{
  \foreach \i in {0,...,9} {
    \putDot{\bzI+2*\i+1}{\i*20}
  }
}
```

Arithmetic and scripting

To avoid syntactic problems, use `\bzEval` to give names to parameters of your macros.

```
\newcommand{\putDot}[2]{
  \bzEval{\x}{#1}
  \bzEval{\r}{#2}
  \node[draw, circle, rotate=\r] at (\x, \bzH) {Hi!};
}

\bzOn{
  \foreach \i in {0,...,9} {
    \putDot{\bzI+2*\i+1}{\i*20}
  }
}
```

Gives:

Arithmetic and scripting

To avoid syntactic problems, use `\bzEval` to give names to parameters of your macros.

```
\newcommand{\putDot}[2]{
  \bzEval{\x}{#1}
  \bzEval{\r}{#2}
  \node[draw, circle, rotate=\r] at (\x, \bzH) {Hi!};
}

\bzOn{
  \foreach \i in {0,...,9} {
    \putDot{\bzI+2*\i+1}{\i*20}
  }
}
```

Gives:

Hint: For readability, you can put your macros outside `bzFrame` environment

Slide counter — basics

Slide counter — basics

`\bzS` $= 2$

To control the flow of time within a frame, a counter `\bzS` is used.

Slide counter — basics

`\bzS` $= 3$

To control the flow of time within a frame, a counter `\bzS` is used.

Each `\bzOn{...}` shows a new content, increasing `\bzS`.

Slide counter — basics

`\bzS` $= 4$

To control the flow of time within a frame, a counter `\bzS` is used.

Each `\bzOn{...}` shows a new content, increasing `\bzS`.

`\bzOne{...}` shows its content for one slide and disappears.

Slide counter — basics

`\bzS` $= 5$

To control the flow of time within a frame, a counter `\bzS` is used.

Each `\bzOn{...}` shows a new content, increasing `\bzS`.

The next `\bzOn{...}` appears at the moment when `\bzOne{...}` disappears.

Slide counter — basics

`\bzS` $= 6$

To control the flow of time within a frame, a counter `\bzS` is used.

Each `\bzOn{...}` shows a new content, increasing `\bzS`.

The next `\bzOn{...}` appears at the moment when `\bzOne{...}` disappears.

`\bzTwo{...}` shows its content for two slides and disappears.

Slide counter — basics

`\bzS` $= 7$

To control the flow of time within a frame, a counter `\bzS` is used.

Each `\bzOn{...}` shows a new content, increasing `\bzS`.

The next `\bzOn{...}` appears at the moment when `\bzOne{...}` disappears.

`\bzTwo{...}` shows its content for two slides and disappears.

The next `\bzOn{...}` appears afterwards.

Slide counter — basics

`\bzS` $= 8$

To control the flow of time within a frame, a counter `\bzS` is used.

Each `\bzOn{...}` shows a new content, increasing `\bzS`.

The next `\bzOn{...}` appears at the moment when `\bzOne{...}` disappears.

The next `\bzOn{...}` appears afterwards.

And only the second `\bzOn{...}` appears when `\bzTwo{...}` disappears.

Slide counter — basics

`\bzS` $= 9$

To control the flow of time within a frame, a counter `\bzS` is used.

Each `\bzOn{...}` shows a new content, increasing `\bzS`.

The next `\bzOn{...}` appears at the moment when `\bzOne{...}` disappears.

The next `\bzOn{...}` appears afterwards.

And only the second `\bzOn{...}` appears when `\bzTwo{...}` disappears.

`\bzOnly{...}` appears for one slide and takes one more to disappear.

Slide counter — basics

`\bzS` $= 10$

To control the flow of time within a frame, a counter `\bzS` is used.

Each `\bzOn{...}` shows a new content, increasing `\bzS`.

The next `\bzOn{...}` appears at the moment when `\bzOne{...}` disappears.

The next `\bzOn{...}` appears afterwards.

And only the second `\bzOn{...}` appears when `\bzTwo{...}` disappears.

Slide counter — basics

`\bzS` $= 11$

To control the flow of time within a frame, a counter `\bzS` is used.

Each `\bzOn{...}` shows a new content, increasing `\bzS`.

The next `\bzOn{...}` appears at the moment when `\bzOne{...}` disappears.

The next `\bzOn{...}` appears afterwards.

And only the second `\bzOn{...}` appears when `\bzTwo{...}` disappears.

The next `\bzOn{...}` appears after one more click.

Slide counter — basics

To control the flow of time within a frame, a counter `\bzS` is used.

Each `\bzOn{...}` shows a new content, increasing `\bzS`.

The next `\bzOn{...}` appears at the moment when `\bzOne{...}` disappears.

The next `\bzOn{...}` appears afterwards.

And only the second `\bzOn{...}` appears when `\bzTwo{...}` disappears.

The next `\bzOn{...}` appears after one more click.

Hint: Some package options display the current value of `\bzS`, see

`draft` and **`brief`** later on.

Slide counter — `\bzIn{from}{to}{...}` `\bzS` $= 1$

Slide counter — `\bzIn{from}{to}{...}` `\bzS` $= 2$

Because of the order of drawing (or other reasons), you may need

to manually handle when content (dis)appears.

Slide counter — `\bzIn{from}{to}{...}` `\bzS` $= 3$

Because of the order of drawing (or other reasons), you may need

to manually handle when content (dis)appears.

First, use `\bzEvalInt{\curS}{\bzS}` to store the current value of `\bzS`.

Slide counter — `\bzIn{from}{to}{...}` `\bzS` $= 4$

Because of the order of drawing (or other reasons), you may need to manually handle when content (dis)appears.

First, use `\bzEvalInt{\curS}{\bzS}` to store the current value of `\bzS`.

Then, use `\bzIn{from}{to}{...}` to control when some content is visible.

Slide counter — `\bzIn{from}{to}{...}`

`\bzS` $= 5$

Because of the order of drawing (or other reasons), you may need to manually handle when content (dis)appears.

First, use `\bzEvalInt{\curS}{\bzS}` to store the current value of `\bzS`.

Then, use `\bzIn{from}{to}{...}` to control when some content is visible.

```
\bzIn{\curS+2}{\curS+7}{...}
```

`\curS` $= 3$

Rules:

Slide counter — `\bzIn{from}{to}{...}` `\bzS` $= 6$

Because of the order of drawing (or other reasons), you may need to manually handle when content (dis)appears.

First, use `\bzEvalInt{\curS}{\bzS}` to store the current value of `\bzS`.

Then, use `\bzIn{from}{to}{...}` to control when some content is visible.

`\bzIn{\curS+2}{\curS+7}{...}` `\curS` $= 3$

Rules:

1. If both arguments are equal, then the content lasts one slide.

Slide counter — `\bzIn{from}{to}{...}`

`\bzS` $= 7$

Because of the order of drawing (or other reasons), you may need
to manually handle when content (dis)appears.

First, use `\bzEvalInt{\curS}{\bzS}` to store the current value of `\bzS`.

Then, use `\bzIn{from}{to}{...}` to control when some content is visible.

`\bzIn{\curS+2}{\curS+7}{...}` `\curS` $= 3$

Rules:

1. If both arguments are equal, then the content lasts one slide.

2. If the first argument is empty, it's visible from the beginning.

Slide counter — `\bzIn{from}{to}{...}` `\bzS` $= 8$

Because of the order of drawing (or other reasons), you may need to manually handle when content (dis)appears.

First, use `\bzEvalInt{\curS}{\bzS}` to store the current value of `\bzS`.

Then, use `\bzIn{from}{to}{...}` to control when some content is visible.

`\bzIn{\curS+2}{\curS+7}{...}` `\curS` $= 3$

Rules:

1. If both arguments are equal, then the content lasts one slide.

2. If the first argument is empty, it's visible from the beginning.

3. If the second argument is empty, it's visible until the end of the frame.

Slide counter — `\bzIn{from}{to}{...}` `\bzS` $= 9$

Because of the order of drawing (or other reasons), you may need to manually handle when content (dis)appears.

First, use `\bzEvalInt{\curS}{\bzS}` to store the current value of `\bzS`.

Then, use `\bzIn{from}{to}{...}` to control when some content is visible.

`\bzIn{\curS+2}{\curS+7}{...}` `\curS` $= 3$

Rules:

1. If both arguments are equal, then the content lasts one slide.

2. If the first argument is empty, it's visible from the beginning.

3. If the second argument is empty, it's visible until the end of the frame.

4. You can do any (integer) arithmetic within the arguments.

Slide counter — `\bzIn{from}{to}{...}` `\bzS` $= 10$

Because of the order of drawing (or other reasons), you may need to manually handle when content (dis)appears.

First, use `\bzEvalInt{\curS}{\bzS}` to store the current value of `\bzS`.

Then, use `\bzIn{from}{to}{...}` to control when some content is visible.

```
\bzIn{\curS+2}{\curS+7}{...}
```

`\curS` $= 3$

Rules:

1. If both arguments are equal, then the content lasts one slide.

2. If the first argument is empty, it's visible from the beginning.

3. If the second argument is empty, it's visible until the end of the frame.

4. You can do any (integer) arithmetic within the arguments.

5. You can use `\bzS` within the arguments — it takes its current value.

Slide counter — `\bzIn{from}{to}{...}` `\bzS` $= 11$

Because of the order of drawing (or other reasons), you may need to manually handle when content (dis)appears.

First, use `\bzEvalInt{\curS}{\bzS}` to store the current value of `\bzS`.

Then, use `\bzIn{from}{to}{...}` to control when some content is visible.

Rules:

1. If both arguments are equal, then the content lasts one slide.
2. If the first argument is empty, it's visible from the beginning.
3. If the second argument is empty, it's visible until the end of the frame.
4. You can do any (integer) arithmetic within the arguments.
5. You can use `\bzS` within the arguments — it takes its current value.
6. `\bzIn{from}{to}{...}` does not modify `\bzS`.

Slide counter — `\bzIn{from}{to}{...}` `\bzS` = 12

Because of the order of drawing (or other reasons), you may need to manually handle when content (dis)appears.

First, use `\bzEvalInt{\curS}{\bzS}` to store the current value of `\bzS`.

Then, use `\bzIn{from}{to}{...}` to control when some content is visible.

Rules:

1. If both arguments are equal, then the content lasts one slide.

2. If the first argument is empty, it's visible from the beginning.

3. If the second argument is empty, it's visible until the end of the frame.

4. You can do any (integer) arithmetic within the arguments.

5. You can use `\bzS` within the arguments — it takes its current value.

6. `\bzIn{from}{to}{...}` does not modify `\bzS`.

7. You should ensure that **`from`** $\leq$ **`to`**.

Slide counter — `\bzIn{from}{to}{...}` `\bzS` $= 13$

Because of the order of drawing (or other reasons), you may need to manually handle when content (dis)appears.

First, use `\bzEvalInt{\curS}{\bzS}` to store the current value of `\bzS`.

Then, use `\bzIn{from}{to}{...}` to control when some content is visible.

Rules:

1. If both arguments are equal, then the content lasts one slide.

2. If the first argument is empty, it's visible from the beginning.

3. If the second argument is empty, it's visible until the end of the frame.

4. You can do any (integer) arithmetic within the arguments.

5. You can use `\bzS` within the arguments — it takes its current value.

6. `\bzIn{from}{to}{...}` does not modify `\bzS`.

7. You should ensure that **`from`** $\leq$ **`to`**.

8. `\bzS` starts with 1.

Slide counter — summary

Slide counter — summary

Available commands:

Slide counter — summary

Available commands:

- `\sBzS{4}` — set slide counter value

Slide counter — summary

Available commands:

- `\sBzS{4}` — set slide counter value
- `\iBzS` — increase slide counter by 1 (+parametrised `\iBzS[2]`)

Slide counter — summary

Available commands:

- `\sBzS{4}` — set slide counter value
- `\iBzS` — increase slide counter by 1 (+parametrised `\iBzS[2]`)
- `\dBzS` — decrease slide counter by 1 (+parametrised `\dBzS[1]`)

Slide counter — summary

Available commands:

- `\sBzS{4}` — set slide counter value
- `\iBzS` — increase slide counter by 1 (+parametrised `\iBzS[2]`)
- `\dBzS` — decrease slide counter by 1 (+parametrised `\dBzS[1]`)
- `\zBzS` — reset `\bzS` to 1

Slide counter — summary

Available commands:

- `\sBzS{4}` — set slide counter value
- `\iBzS` — increase slide counter by 1 (+parametrised `\iBzS[2]`)
- `\dBzS` — decrease slide counter by 1 (+parametrised `\dBzS[1]`)
- `\zBzS` — reset `\bzS` to 1

Hint: if possible, avoid guessing correct numbers for `\bzIn{from}{to}{...}`,

use a variant of `\bzEvalInt{\curS}{\bzS}` instead!

Slide counter — summary

Available commands:

- `\sBzS{4}` — set slide counter value
- `\iBzS` — increase slide counter by 1 (+parametrised `\iBzS[2]`)
- `\dBzS` — decrease slide counter by 1 (+parametrised `\dBzS[1]`)
- `\zBzS` — reset `\bzS` to 1

Hint: if possible, avoid guessing correct numbers for `\bzIn{from}{to}{...}`,

use a variant of `\bzEvalInt{\curS}{\bzS}` instead!

Hint: remember that the content overlays each other.

The order comes from the order of entries in the source file,

not from the values of `\bzS`.

Slide counter — summary

Available commands:

- `\sBzS{4}` — set slide counter value
- `\iBzS` — increase slide counter by 1 (+parametrised `\iBzS[2]`)
- `\dBzS` — decrease slide counter by 1 (+parametrised `\dBzS[1]`)
- `\zBzS` — reset `\bzS` to 1

Hint: if possible, avoid guessing correct numbers for `\bzIn{from}{to}{...}`,

use a variant of `\bzEvalInt{\curS}{\bzS}` instead!

Hint: remember that the content overlays each other.

The order comes from the order of entries in the source file,

not from the values of `\bzS`.

Hint: to layerize the content, use the optional argument of `\bzOn[z]{...}`, . . .

Order of compilation

Named nodes

Named nodes

These commands create Ti*k*Z nodes:

`\bzLeft{...}`, `\bzText{...}`, `\bzItem{...}`, `\bzList{...}`,

`\bzCenter{...}`, `\bzRight{...}`, `\bzNext{...}`, `\bzBox{...}`.

Named nodes

Named nodes

You can append new content to the last node using `\bzNext{...}`:

Named nodes

You can append new content to the last node using `\bzNext{...}`:

```
\bzLeft{...}
```

Named nodes

You can append new content to the last node using `\bzNext{...}`:

```
\bzLeft{...}\bzNext{...}
```

Named nodes

You can append new content to the last node using `\bzNext{...}`:

```
\bzLeft{...}\bzNext{...}\bzNext{...}
```

Named nodes

You can append new content to the last node using `\bzNext{...}`:

```
\bzLeft{...}\bzNext{...}\bzNext{...}
```

You can also attach content to the left (once):

Named nodes

You can append new content to the last node using `\bzNext{...}`:

```
\bzLeft{...}\bzNext{...}\bzNext{...}
```

You can also attach content to the left (once):

```
\bzCenter{...}
```

Named nodes

You can append new content to the last node using `\bzNext{...}`:

```
\bzLeft{...}\bzNext{...}\bzNext{...}
```

You can also attach content to the left (once):

```
\bzPrev{...}\bzCenter{...}
```

Named nodes

You can append new content to the last node using `\bzNext{...}`:

```
\bzLeft{...}\bzNext{...}\bzNext{...}
```

You can also attach content to the left (once):

```
\bzPrev{...}\bzCenter{...}\bzNext{...}
```

Named nodes

You can append new content to the last node using `\bzNext{...}`:

```
\bzLeft{...}\bzNext{...}\bzNext{...}
```

You can also attach content to the left (once):

```
\bzPrev{...}\bzCenter{...}\bzNext{...}\bzNext{...}
```

Named nodes

You can append new content to the last node using `\bzNext{...}`:

```
\bzLeft{...}\bzNext{...}\bzNext{...}
```

You can also attach content to the left (once):

```
\bzPrev{...}\bzCenter{...}\bzNext{...}\bzNext{...}
```

Hint: to surround a node:

first use `\bzPrev{...}` and then `\bzNext{...}`!

Named nodes

You can append new content to the last node using `\bzNext{...}`:

`\bzLeft{...}\bzNext{...}\bzNext{...}`

You can also attach content to the left (once):

`\bzPrev{...}\bzCenter{...}\bzNext{...}\bzNext{...}`

Hint: to surround a node:

first use `\bzPrev{...}` and then `\bzNext{...}`!

This is controlled by a counter named `\bzN`.

Named nodes

You can append new content to the last node using `\bzNext{...}`:

`\bzLeft{...}\bzNext{...}\bzNext{...}`

You can also attach content to the left (once):

`\bzPrev{...}\bzCenter{...}\bzNext{...}\bzNext{...}`

Hint: to surround a node:

first use `\bzPrev{...}` and then `\bzNext{...}`!

This is controlled by a counter named `\bzN`.

Hint: you should never modify the value of that counter yourself!

Named nodes

You can append new content to the last node using `\bzNext{...}`:

```
\bzLeft{...}\bzNext{...}\bzNext{...}
```

You can also attach content to the left (once):

```
\bzPrev{...}\bzCenter{...}\bzNext{...}\bzNext{...}
```

Hint: to surround a node:

first use `\bzPrev{...}` and then `\bzNext{...}`!

This is controlled by a counter named `\bzN`.

Hint: you should never modify the value of that counter yourself!

You can store the number of the last node using:

```
\bzEvalInt{\curN}{\bzN}
```

Named nodes

You can append new content to the last node using `\bzNext{...}`:

`\bzLeft{...}\bzNext{...}\bzNext{...}`

You can also attach content to the left (once):

`\bzPrev{...}\bzCenter{...}\bzNext{...}\bzNext{...}`

Hint: to surround a node:

first use `\bzPrev{...}` and then `\bzNext{...}`!

This is controlled by a counter named `\bzN`.

Hint: you should never modify the value of that counter yourself!

You can store the number of the last node using:

`\bzEvalInt{\curN}{\bzN}`

You can later append content to that node using parametrised versions:

Named nodes

You can append new content to the last node using `\bzNext{...}`:

```
\bzLeft{...}\bzNext{...}\bzNext{...}
```

You can also attach content to the left (once):

```
\bzPrev{...}\bzCenter{...}\bzNext{...}\bzNext{...}
```

Hint: to surround a node:

first use `\bzPrev{...}` and then `\bzNext{...}`!

This is controlled by a counter named `\bzN`.

Hint: you should never modify the value of that counter yourself!

You can store the number of the last node using:

`\bzEvalInt{\curN}{\bzN}`**I'm next**

You can later append content to that node using parametrised versions:

```
\bzNext{\bf I'm next}[\curN]
```

Named nodes

You can append new content to the last node using `\bzNext{...}`:

`\bzLeft{...}\bzNext{...}\bzNext{...}`

You can also attach content to the left (once):

`\bzPrev{...}\bzCenter{...}\bzNext{...}\bzNext{...}`

Hint: to surround a node:

first use `\bzPrev{...}` and then `\bzNext{...}`!

This is controlled by a counter named `\bzN`.

Hint: you should never modify the value of that counter yourself!

You can store the number of the last node using:

I'm prev`\bzEvalInt{\curN}{\bzN}`**I'm next**

You can later append content to that node using parametrised versions:

```
\bzNext{\bf I'm next}[\curN]
```

```
\bzPrev{\bf I'm prev}[\curN]
```

Named nodes

You can append new content to the last node using `\bzNext{...}`:

```
\bzLeft{...}\bzNext{...}\bzNext{...}
```

You can also attach content to the left (once):

```
\bzPrev{...}\bzCenter{...}\bzNext{...}\bzNext{...}
```

Hint: to surround a node:

first use `\bzPrev{...}` and then `\bzNext{...}`!

This is controlled by a counter named `\bzN`.

Hint: you should never modify the value of that counter yourself!

You can store the number of the last node using:

I'm prev`\bzEvalInt{\curN}{\bzN}`**I'm next**

You can later append content to that node using parametrised versions:

```
\bzNext{\bf I'm next}[\curN]

\bzPrev{\bf I'm prev}[\curN]
```

Hint: nodes created by `\bzPrev{...}` do not have labels!

Named nodes — hints and tricks

Named nodes — hints and tricks

Hint: you can style these nodes: `\bzPrev[draw]{...}`

Named nodes — hints and tricks

Hint: you can style these nodes: `\bzPrev[draw]{...}`

To create an aligned large mathematical equation, you can use:

Named nodes — hints and tricks

Hint: you can style these nodes: `\bzPrev[draw]{...}`

To create an aligned large mathematical equation, you can use:

```
\bzCenter[scale=1.3]{$=$}
\bzPrev[scale=1.3]{$\int_a^b f'(x)\ $}
\bzNext[scale=1.3]{$\ f(x)\Big|_a^b$}
```

```
\bzCenter[scale=1.3]{$\leq$}
\bzPrev[scale=1.3]{$\Big|\int_a^b f(x)\Big|\ $}
\bzNext[scale=1.3]{$\ \int_a^b \big|f(x)\big|$}
```

Named nodes — hints and tricks

Hint: you can style these nodes: `\bzPrev[draw]{...}`

To create an aligned large mathematical equation, you can use:

```
\bzCenter[scale=1.3]{$=$}
\bzPrev[scale=1.3]{$\int_a^b f'(x)\ $}
\bzNext[scale=1.3]{$\ f(x)\Big|_a^b$}
```

```
\bzCenter[scale=1.3]{$\leq$}
\bzPrev[scale=1.3]{$\Big|\int_a^b f(x)\Big|\ $}
\bzNext[scale=1.3]{$\ \int_a^b \big|f(x)\big|$}
```

$$\int_a^b f'(x) = f(x)\Big|_a^b$$

Named nodes — hints and tricks

Hint: you can style these nodes: `\bzPrev[draw]{...}`

To create an aligned large mathematical equation, you can use:

```
\bzCenter[scale=1.3]{$=$}
\bzPrev[scale=1.3]{$\int_a^b f'(x)\ $}
\bzNext[scale=1.3]{$\ f(x)\Big|_a^b$}
```

```
\bzCenter[scale=1.3]{$\leq$}
\bzPrev[scale=1.3]{$\Big|\int_a^b f(x)\Big|\ $}
\bzNext[scale=1.3]{$\ \int_a^b \big|f(x)\big|$}
```

$$\int_a^b f'(x) = f(x)\Big|_a^b$$

$$\Big|\int_a^b f(x)\Big| \leq \int_a^b \big|f(x)\big|$$

Named nodes — hints and tricks

Hint: you can style these nodes: `\bzPrev[draw]{...}`

To create an aligned large mathematical equation, you can use:

```
\bzCenter[scale=1.3]{$=$}
\bzPrev[scale=1.3]{$\int_a^b f'(x)\ $}
\bzNext[scale=1.3]{$\ f(x)\Big|_a^b$}
```

```
\bzCenter[scale=1.3]{$\leq$}
\bzPrev[scale=1.3]{$\Big|\int_a^b f(x)\Big|\ $}
\bzNext[scale=1.3]{$\ \int_a^b \big|f(x)\big|$}
```

$$\int_a^b f'(x) = f(x)\Big|_a^b$$

$$\Big|\int_a^b f(x)\Big| \leq \int_a^b \big|f(x)\big|$$

However, a similar effect can be also achieved via `[bzR]`, `[bzL]`, and `[bzC]`:

Named nodes — hints and tricks

Hint: you can style these nodes: `\bzPrev[draw]{...}`

To create an aligned large mathematical equation, you can use:

```
\bzCenter[scale=1.3]{$=$}
\bzPrev[scale=1.3]{$\int_a^b f'(x)\ $}
\bzNext[scale=1.3]{$\ f(x)\Big|_a^b$}
```

```
\bzCenter[scale=1.3]{$\leq$}
\bzPrev[scale=1.3]{$\Big|\int_a^b f(x)\Big|\ $}
\bzNext[scale=1.3]{$\ \int_a^b \big|f(x)\big|$}
```

$$\int_a^b f'(x) = f(x)\Big|_a^b$$

$$\Big|\int_a^b f(x)\Big| \leq \int_a^b \big|f(x)\big|$$

However, a similar effect can be also achieved via `[bzR]`, `[bzL]`, and `[bzC]`:

```
\node[bzR] at (5-1, \bzH) {$(f(g(x))'$};
\node[bzC] at (5+0, \bzH) {$=$};
\node[bzL] at (5+1, \bzH) {$(f'(g(x))\cdot g'(x)$};
```

Named nodes — hints and tricks

Hint: you can style these nodes: `\bzPrev[draw]{...}`

To create an aligned large mathematical equation, you can use:

```
\bzCenter[scale=1.3]{$=$}
\bzPrev[scale=1.3]{$\int_a^b f'(x)\ $}
\bzNext[scale=1.3]{$\ f(x)\Big|_a^b$}
```

```
\bzCenter[scale=1.3]{$\leq$}
\bzPrev[scale=1.3]{$\Big|\int_a^b f(x)\Big|\ $}
\bzNext[scale=1.3]{$\ \int_a^b \big|f(x)\big|$}
```

$$\int_a^b f'(x) = f(x)\Big|_a^b$$

$$\Big|\int_a^b f(x)\Big| \leq \int_a^b \big|f(x)\big|$$

However, a similar effect can be also achieved via `[bzR]`, `[bzL]`, and `[bzC]`:

```
\node[bzR] at (5-1, \bzH) {$(f(g(x))'$};
\node[bzC] at (5+0, \bzH) {$=$};
\node[bzL] at (5+1, \bzH) {$(f'(g(x))\cdot g'(x)$};
```

$$(f(g(x))' \quad = \quad (f'(g(x))\cdot g'(x)$$

Named nodes — hints and tricks

Hint: you can style these nodes: `\bzPrev[draw]{...}`

To create an aligned large mathematical equation, you can use:

```
\bzCenter[scale=1.3]{$=$}
\bzPrev[scale=1.3]{$\int_a^b f'(x)\ $}
\bzNext[scale=1.3]{$\ f(x)\Big|_a^b$}
```

```
\bzCenter[scale=1.3]{$\leq$}
\bzPrev[scale=1.3]{$\Big|\int_a^b f(x)\Big|\ $}
\bzNext[scale=1.3]{$\ \int_a^b \big|f(x)\big|$}
```

$$\int_a^b f'(x) = f(x)\Big|_a^b$$

$$\Big|\int_a^b f(x)\Big| \leq \int_a^b \big|f(x)\big|$$

However, a similar effect can be also achieved via `[bzR]`, `[bzL]`, and `[bzC]`:

```
\node[bzR] at (5-1, \bzH) {$(f(g(x))'$};
\node[bzC] at (5+0, \bzH) {$=$};
\node[bzL] at (5+1, \bzH) {$(f'(g(x))\cdot g'(x)$};
```

$$(f(g(x))' \quad = \quad (f'(g(x))\cdot g'(x)$$

Hint: you can break a word using `[bzR]` and `[bzL]` (and the nodes match):

Named nodes — hints and tricks

Hint: you can style these nodes: `\bzPrev[draw]{...}`

To create an aligned large mathematical equation, you can use:

```
\bzCenter[scale=1.3]{$=$}
\bzPrev[scale=1.3]{$\int_a^b f'(x)\ $}
\bzNext[scale=1.3]{$\ f(x)\Big|_a^b$}
```

```
\bzCenter[scale=1.3]{$\leq$}
\bzPrev[scale=1.3]{$\Big|\int_a^b f(x)\Big|\ $}
\bzNext[scale=1.3]{$\ \int_a^b \big|f(x)\big|$}
```

$$\int_a^b f'(x) = f(x)\Big|_a^b$$

$$\Big|\int_a^b f(x)\Big| \leq \int_a^b \big|f(x)\big|$$

However, a similar effect can be also achieved via `[bzR]`, `[bzL]`, and `[bzC]`:

```
\node[bzR] at (5-1, \bzH) {$(f(g(x))'$};
\node[bzC] at (5+0, \bzH) {$=$};
\node[bzL] at (5+1, \bzH) {$(f'(g(x))\cdot g'(x)$};
```

$$(f(g(x))' \quad = \quad (f'(g(x))\cdot g'(x)$$

Hint: you can break a word using `[bzR]` and `[bzL]` (and the nodes match):

```
\node[bzR] at (0, \bzH) {deoxyribo}; \node[bzL] at (0, \bzH) {nucleic acid};
```

Named nodes — hints and tricks

Hint: you can style these nodes: `\bzPrev[draw]{...}`

To create an aligned large mathematical equation, you can use:

```
\bzCenter[scale=1.3]{$=$}
\bzPrev[scale=1.3]{$\int_a^b f'(x)\ $}
\bzNext[scale=1.3]{$\ f(x)\Big|_a^b$}
```

```
\bzCenter[scale=1.3]{$\leq$}
\bzPrev[scale=1.3]{$\Big|\int_a^b f(x)\Big|\ $}
\bzNext[scale=1.3]{$\ \int_a^b \big|f(x)\big|$}
```

$$\int_a^b f'(x) = f(x)\Big|_a^b$$

$$\Big|\int_a^b f(x)\Big| \leq \int_a^b \big|f(x)\big|$$

However, a similar effect can be also achieved via `[bzR]`, `[bzL]`, and `[bzC]`:

```
\node[bzR] at (5-1, \bzH) {$(f(g(x))'$};
\node[bzC] at (5+0, \bzH) {$=$};
\node[bzL] at (5+1, \bzH) {$(f'(g(x))\cdot g'(x)$};
```

$$(f(g(x))' \quad = \quad (f'(g(x))\cdot g'(x)$$

Hint: you can break a word using `[bzR]` and `[bzL]` (and the nodes match):

```
\node[bzR] at (0, \bzH) {deoxyribo}; \node[bzL] at (0, \bzH) {nucleic acid};
```

deoxyribo

Named nodes — hints and tricks

Hint: you can style these nodes: `\bzPrev[draw]{...}`

To create an aligned large mathematical equation, you can use:

```
\bzCenter[scale=1.3]{$=$}
\bzPrev[scale=1.3]{$\int_a^b f'(x)\ $}
\bzNext[scale=1.3]{$\ f(x)\Big|_a^b$}
```

```
\bzCenter[scale=1.3]{$\leq$}
\bzPrev[scale=1.3]{$\Big|\int_a^b f(x)\Big|\ $}
\bzNext[scale=1.3]{$\ \int_a^b \big|f(x)\big|$}
```

$$\int_a^b f'(x) = f(x)\Big|_a^b$$

$$\Big|\int_a^b f(x)\Big| \leq \int_a^b \big|f(x)\big|$$

However, a similar effect can be also achieved via `[bzR]`, `[bzL]`, and `[bzC]`:

```
\node[bzR] at (5-1, \bzH) {$(f(g(x))'$};
\node[bzC] at (5+0, \bzH) {$=$};
\node[bzL] at (5+1, \bzH) {$(f'(g(x))\cdot g'(x)$};
```

$$(f(g(x))' \quad = \quad (f'(g(x))\cdot g'(x)$$

Hint: you can break a word using `[bzR]` and `[bzL]` (and the nodes match):

```
\node[bzR] at (0, \bzH) {deoxyribo}; \node[bzL] at (0, \bzH) {nucleic acid};
```

deoxyribonucleic acid

Named nodes — explicit references

Named nodes — explicit references

Let's create a referable node:

Named nodes — explicit references

Let's create a referable node:

```
\bzCenter{Referable node!}
\bzEvalInt{\nodeN}{\bzN}
```

Named nodes — explicit references

Let's create a referable node:

```
\bzCenter{Referable node!}
\bzEvalInt{\nodeN}{\bzN}
```

Referable node!

Named nodes — explicit references

Let's create a referable node:

```
\bzCenter{Referable node!}
\bzEvalInt{\nodeN}{\bzN}
```

Referable node!

It's label can be obtained by `\bzLabel{\nodeN}`:

Named nodes — explicit references

Let's create a referable node:

```
\bzCenter{Referable node!}
\bzEvalInt{\nodeN}{\bzN}
```

Referable node!

It's label can be obtained by `\bzLabel{\nodeN}`:

```
\draw (7, \bzH) edge[Circle-Latex, bend right=50] (\bzLabel{\nodeN}.mid east);
```

Named nodes — explicit references

Let's create a referable node:

```
\bzCenter{Referable node!}
\bzEvalInt{\nodeN}{\bzN}
```

Referable node!

It's label can be obtained by `\bzLabel{\nodeN}`:

```
\draw (7, \bzH) edge[Circle-Latex, bend right=50] (\bzLabel{\nodeN}.mid east);
```

Hint: you cannot draw relatively to a node that has disappeared :)

Named nodes — explicit references

Let's create a referable node:

```
\bzCenter{Referable node!}
\bzEvalInt{\nodeN}{\bzN}
```

Referable node!

It's label can be obtained by `\bzLabel{\nodeN}`:

```
\draw (7, \bzH) edge[Circle-Latex, bend right=50] (\bzLabel{\nodeN}.mid east);
```

Hint: you cannot draw relatively to a node that has disappeared :)

You may use `calc`-based references, like:

```
\draw (8, \bzH) edge[Circle-Latex, bend right=50] ($(\bzLabel{\nodeN}.east)+(4,0)$);
```

Named nodes — explicit references

Let's create a referable node:

```
\bzCenter{Referable node!}
\bzEvalInt{\nodeN}{\bzN}
```

$$\underbrace{\text{Referable node!}}_{\text{the one}}$$

It's label can be obtained by `\bzLabel{\nodeN}`:

```
\draw (7, \bzH) edge[Circle-Latex, bend right=50] (\bzLabel{\nodeN}.mid east);
```

Hint: you cannot draw relatively to a node that has disappeared :)

You may use `calc`-based references, like:

```
\draw (8, \bzH) edge[Circle-Latex, bend right=50] ($(\bzLabel{\nodeN}.east)+(4,0)$);
```

You may also use the great `|-` and `-|` features of Ti*k*Z:

```
\coordinate (base) at ($(\bzLabel{\nodeN}.east)+(0,-0.5)$);
\draw
    (\bzLabel{\nodeN}.west |- base)
    edge[bzBraceU] node{the one}
    (base -| \bzLabel{\nodeN}.east);
```

Braces

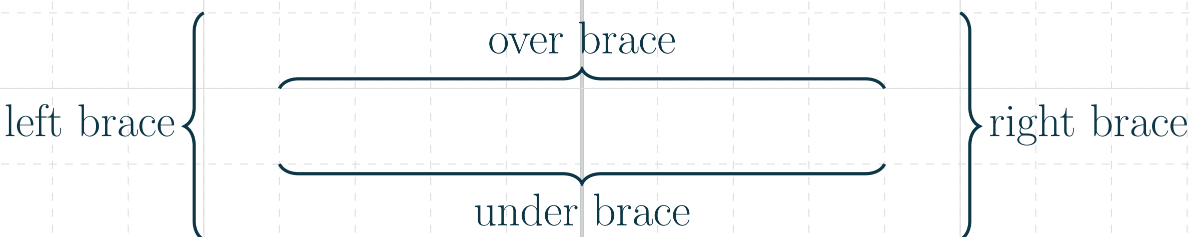
Braces

Use the following commands to draw braces:

Braces

Use the following commands to draw braces:

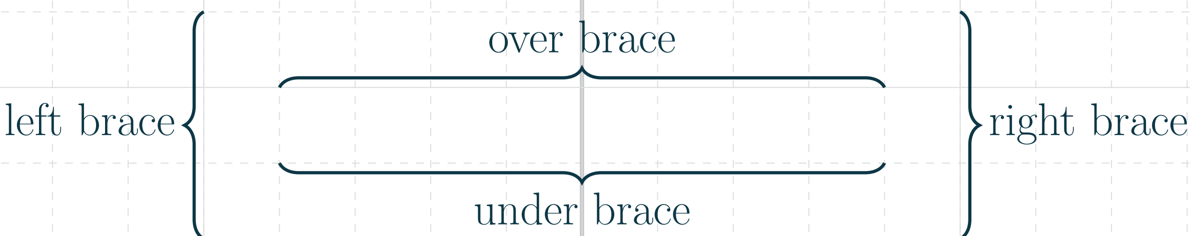
```
\path (-4, -11) edge[bzBraceU] node{under brace} (+4, -11);
\path (-4, -6)  edge[bzBraceO] node{over brace}  (+4, -6);
\path (-5, -10) edge[bzBraceL] node{left brace}  (-5, -7);
\path (+5, -10) edge[bzBraceR] node{right brace} (+5, -7);
```



Braces

Use the following commands to draw braces:

```
\path (-4, -11) edge[bzBraceU] node{under brace} (+4, -11);
\path (-4, -6)  edge[bzBraceO] node{over brace}  (+4, -6);
\path (-5, -10) edge[bzBraceL] node{left brace}  (-5, -7);
\path (+5, -10) edge[bzBraceR] node{right brace} (+5, -7);
```

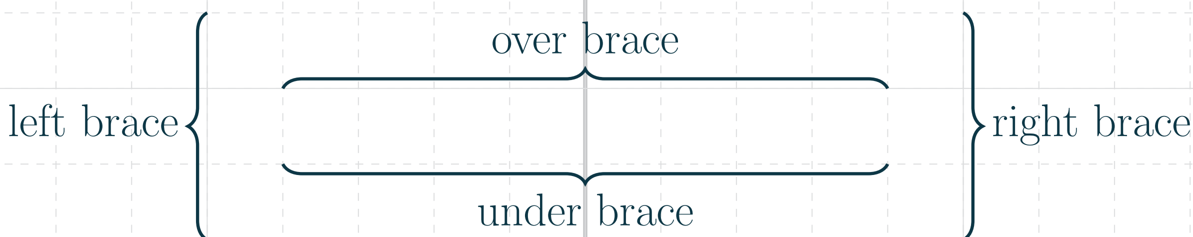


You can also create braces similarly to `\bzNext{...}`:

Braces

Use the following commands to draw braces:

```
\path (-4, -11) edge[bzBraceU] node{under brace} (+4, -11);
\path (-4, -6)  edge[bzBraceO] node{over brace}  (+4, -6);
\path (-5, -10) edge[bzBraceL] node{left brace}  (-5, -7);
\path (+5, -10) edge[bzBraceR] node{right brace} (+5, -7);
```



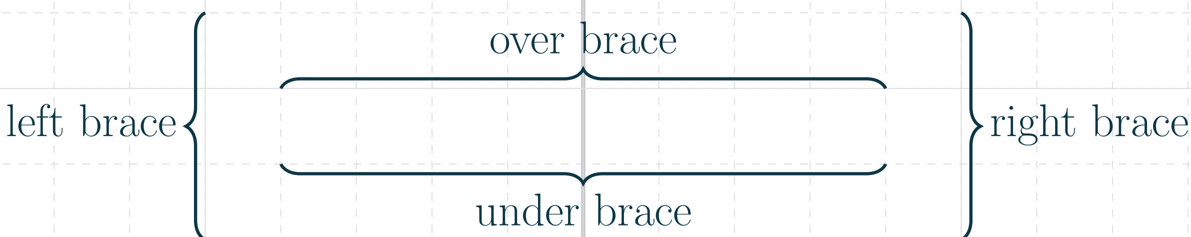
You can also create braces similarly to `\bzNext{...}`:

```
\bzCenter{First}
\bzBraceU{1}
\bzNext{\ }

\bzNext{Second}
\bzEvalInt{\secondN}{\bzN}
\bzNext{\ }

\bzNext{Third}
\bzBraceU{3}

\bzBraceU{2}[\secondN]
```

Braces

Use the following commands to draw braces:

```
\path (-4, -11) edge[bzBraceU] node{under brace} (+4, -11);
\path (-4, -6)  edge[bzBraceO] node{over brace}  (+4, -6);
\path (-5, -10) edge[bzBraceL] node{left brace}  (-5, -7);
\path (+5, -10) edge[bzBraceR] node{right brace} (+5, -7);
```



You can also create braces similarly to `\bzNext{...}`:

```
\bzCenter{First}
\bzBraceU{1}
\bzNext{\ }

\bzNext{Second}
\bzEvalInt{\secondN}{\bzN}
\bzNext{\ }

\bzNext{Third}
\bzBraceU{3}

\bzBraceU{2}[\secondN]
```

Beamer-like blocks

Beamer-like blocks

You can draw blocks like in Beamer.

Beamer-like blocks

You can draw blocks like in Beamer.

The basic command is `\bzBlock{...}{...}`:

```
\bzBlock{0.5}{0.5}
\bzCenter{A simple block!}
```

which produces a simple block with border, positioned relatively to `\bzH`:

A simple block!

Beamer-like blocks

You can draw blocks like in Beamer.

The basic command is `\bzBlock{...}{...}`:

```
\bzBlock{0.5}{0.5}
\bzCenter{A simple block!}
```

which produces a simple block with border, positioned relatively to `\bzH`:

A simple block!

The two arguments of `\bzBlock{...}{...}` measure the height of the block:

Beamer-like blocks

You can draw blocks like in Beamer.

The basic command is `\bzBlock{...}{...}`:

```
\bzBlock{0.5}{0.5}
\bzCenter{A simple block!}
```

which produces a simple block with border, positioned relatively to `\bzH`:

A simple block!

The two arguments of `\bzBlock{...}{...}` measure the height of the block:

- the first one above the current value of `\bzH` up to the top,

Beamer-like blocks

You can draw blocks like in Beamer.

The basic command is `\bzBlock{...}{...}`:

```
\bzBlock{0.5}{0.5}
\bzCenter{A simple block!}
```

which produces a simple block with border, positioned relatively to `\bzH`:

A simple block!

The two arguments of `\bzBlock{...}{...}` measure the height of the block:

- the first one above the current value of `\bzH` up to the top,
- the second one below the current value of `\bzH` down to the bottom.

Beamer-like blocks

You can draw blocks like in Beamer.

The basic command is `\bzBlock{...}{...}`:

```
\bzBlock{0.5}{0.5}
\bzCenter{A simple block!}
```

which produces a simple block with border, positioned relatively to `\bzH`:

A simple block!

The two arguments of `\bzBlock{...}{...}` measure the height of the block:

- the first one above the current value of `\bzH` up to the top,
- the second one below the current value of `\bzH` down to the bottom.

Moreover, the first optional argument is styling, like:

```
\bzBlock[fill=red]{...}{...}
```

Beamer-like blocks

You can draw blocks like in Beamer.

The basic command is `\bzBlock{...}{...}`:

```
\bzBlock{0.5}{0.5}
\bzCenter{A simple block!}
```

which produces a simple block with border, positioned relatively to `\bzH`:

A simple block!

The two arguments of `\bzBlock{...}{...}` measure the height of the block:

- the first one above the current value of `\bzH` up to the top,
- the second one below the current value of `\bzH` down to the bottom.

Moreover, the first optional argument is styling, like:

```
\bzBlock[fill=red]{...}{...}
```

The topmost block of this frame is obtained as:

```
\bzBlock[draw=none, fill=\bzCtext, opacity=0.4]{0.5}{0.5}
\bzBlock[]{0.5}{1.5}
\bzCenter[bzEB]{Beamer-like blocks}
\bzLeft{You can draw blocks like in Beamer.}
```

Class options — font size and `plain`

Class options — font size and `plain`

The class `BeamerikZ` accepts a number of class options.

Class options — font size and `plain`

The class `BeamerikZ` accepts a number of class options.

The first category are options regarding font size:

Class options — font size and `plain`

The class `BeamerikZ` accepts a number of class options.

The first category are options regarding font size:

- **`small`** defines font size of 14pt (smaller than usually),

Class options — font size and `plain`

The class `BeamerikZ` accepts a number of class options.

The first category are options regarding font size:

- **`small`** defines font size of 14pt (smaller than usually),
- **`basic`** is the default option with the font of 17pt used,

Class options — font size and `plain`

The class `BeamerikZ` accepts a number of class options.

The first category are options regarding font size:

- `small` defines font size of 14pt (smaller than usually),
- `basic` is the default option with the font of 17pt used,
- `large` uses font of 20pt.

Class options — font size and `plain`

The class `BeamerikZ` accepts a number of class options.

The first category are options regarding font size:

- `small` defines font size of 14pt (smaller than usually),
- `basic` is the default option with the font of 17pt used,
- `large` uses font of 20pt.

Specify these options when invoking document class, e.g.:

```
\documentclass[large]{beamerikz}
```

Class options — font size and `plain`

The class `BeamerikZ` accepts a number of class options.

The first category are options regarding font size:

- `small` defines font size of 14pt (smaller than usually),
- `basic` is the default option with the font of 17pt used,
- `large` uses font of 20pt.

Specify these options when invoking document class, e.g.:

```
\documentclass[large]{beamerikz}
```

This document is typeset using the default option of 17pt, i.e. with:

```
\documentclass{beamerikz}
```

Class options — font size and `plain`

The class **`BeamerikZ`** accepts a number of class options.

The first category are options regarding font size:

- **`small`** defines font size of 14pt (smaller than usually),
- **`basic`** is the default option with the font of 17pt used,
- **`large`** uses font of 20pt.

Specify these options when invoking document class, e.g.:

```
\documentclass[large]{beamerikz}
```

This document is typeset using the default option of 17pt, i.e. with:

```
\documentclass{beamerikz}
```

All the class options are processed sequentially, so the following code gives the default 17pt font size:

```
\documentclass[large, small, basic, large, basic, large, basic]{beamerikz}
```

Class options — font size and `plain`

The class **`BeamerikZ`** accepts a number of class options.

The first category are options regarding font size:

- **`small`** defines font size of 14pt (smaller than usually),
- **`basic`** is the default option with the font of 17pt used,
- **`large`** uses font of 20pt.

Specify these options when invoking document class, e.g.:

```
\documentclass[large]{beamerikz}
```

This document is typeset using the default option of 17pt, i.e. with:

```
\documentclass{beamerikz}
```

All the class options are processed sequentially, so the following code gives the default 17pt font size:

```
\documentclass[large, small, basic, large, basic, large, basic]{beamerikz}
```

If you really hate "Powered by **`Beameri`***k*Z" on the first plain frame, use the class option **`plain`**.

Class options — 16:9

Class options — 16:9

Package option `wide` sets the aspect ratio of the presentation to 16:9

```
\documentclass[wide]{beamerikz}
```

Class options — 16:9

Package option `wide` sets the aspect ratio of the presentation to 16:9

```
\documentclass[wide]{beamerikz}
```

It **does not** change the size of the text!

In particular, you can still use options `small`, `basic`, and `large`

Class options — 16:9

Package option `wide` sets the aspect ratio of the presentation to 16:9

```
\documentclass[wide]{beamerikz}
```

It **does not** change the size of the text!

In particular, you can still use options `small`, `basic`, and `large`

Also, the vertical space is unchanged: (-1.0 – 14.5)

Class options — 16:9

Package option `wide` sets the aspect ratio of the presentation to 16:9

```
\documentclass[wide]{beamerikz}
```

It **does not** change the size of the text!

In particular, you can still use options `small`, `basic`, and `large`

Also, the vertical space is unchanged: (-1.0 – 14.5)

However, the horizontal space is then: (-13.0 – +13.0) (plus margins)

Class options — 16:9

Package option `wide` sets the aspect ratio of the presentation to 16:9

```
\documentclass[wide]{beamerikz}
```

It **does not** change the size of the text!

In particular, you can still use options `small`, `basic`, and `large`

Also, the vertical space is unchanged: (-1.0 – 14.5)

However, the horizontal space is then: (-13.0 – +13.0) (plus margins)

This means that the default value of `bzI` becomes -13.

Class options — [show] frames

Class options — [show] frames

Compilation of complex presentations may take up to a couple of minutes!

Class options — `[show]` frames

Compilation of complex presentations may take up to a couple of minutes!

To speed-up the development process, **`BeamerikZ`** provides a mechanism, that allows do disable compilation of certain frames to focus on the currently developed ones.

Class options — `[show]` frames

Compilation of complex presentations may take up to a couple of minutes!

To speed-up the development process, **`BeamerikZ`** provides a mechanism, that allows do disable compilation of certain frames to focus on the currently developed ones.

Frames of interest should be marked with the option **`[show]`**:

```
\begin{bzFrame}[show]
...
\end{bzFrame}
```

Class options — [show] frames

Compilation of complex presentations may take up to a couple of minutes!

To speed-up the development process, **BeamerikZ** provides a mechanism, that allows do disable compilation of certain frames to focus on the currently developed ones.

Frames of interest should be marked with the option **[show]**:

```
\begin{bzFrame}[show]
...
\end{bzFrame}
```

While the rest should have this option commented out:

```
\begin{bzFrame}%[show]  % to make this frame [show] again, remove % in this line
...
\end{bzFrame}
```

Class options — `[show]` frames

Compilation of complex presentations may take up to a couple of minutes!

To speed-up the development process, `BeamerikZ` provides a mechanism, that allows do disable compilation of certain frames to focus on the currently developed ones.

Frames of interest should be marked with the option `[show]`:

```
\begin{bzFrame}[show]
...
\end{bzFrame}
```

While the rest should have this option commented out:

```
\begin{bzFrame}%[show]  % to make this frame [show] again, remove % in this line
...
\end{bzFrame}
```

The same option applies to the “plain” frames:

```
\begin{bzPlainFrame}%[show]  % to make this frame [show] again, just remove %
...
\end{bzPlainFrame}
```

Class options — [show] frames

Compilation of complex presentations may take up to a couple of minutes!

To speed-up the development process, BeamerikZ provides a mechanism, that allows do disable compilation of certain frames to focus on the currently developed ones.

Frames of interest should be marked with the option [show]:

```
\begin{bzFrame}[show]
...
\end{bzFrame}
```

While the rest should have this option commented out:

```
\begin{bzFrame}%[show]  % to make this frame [show] again, remove % in this line
...
\end{bzFrame}
```

The same option applies to the "plain" frames:

```
\begin{bzPlainFrame}%[show]  % to make this frame [show] again, just remove %
...
\end{bzPlainFrame}
```

Hint: to make use of these options, read on!

Class options — compilation modes

Class options — compilation modes

Similarly as `normal` and `large`, `BeamerikZ` reacts to the following options:

Class options — compilation modes

Similarly as `normal` and `large`, `BeamerikZ` reacts to the following options:

- `final` — the default option, when all the frames are normally compiled

Class options — compilation modes

Similarly as `normal` and `large`, `BeamerikZ` reacts to the following options:

- `final` — the default option, when all the frames are normally compiled
- `ready` — similar to `final`, but add compile info (see later)

Class options — compilation modes

Similarly as `normal` and `large`, `BeamerikZ` reacts to the following options:

- `final` — the default option, when all the frames are normally compiled
- `ready` — similar to `final`, but add compile info (see later)
- `draft` — compiles `[show]` frames, others are made one-shot

Class options — compilation modes

Similarly as `normal` and `large`, `BeamerikZ` reacts to the following options:

- `final` — the default option, when all the frames are normally compiled
- `ready` — similar to `final`, but add compile info (see later)
- `draft` — compiles `[show]` frames, others are made one-shot
- `short` — compiles all the frames in one-shot, `[show]` is ignored

Class options — compilation modes

Similarly as `normal` and `large`, `BeamerikZ` reacts to the following options:

- `final` — the default option, when all the frames are normally compiled
- `ready` — similar to `final`, but add compile info (see later)
- `draft` — compiles `[show]` frames, others are made one-shot
- `short` — compiles all the frames in one-shot, `[show]` is ignored
- `brief` — compiles only `[show]` frames, others are blank

Class options — compilation modes

Similarly as `normal` and `large`, `BeamerikZ` reacts to the following options:

- `final` — the default option, when all the frames are normally compiled
- `ready` — similar to `final`, but add compile info (see later)
- `draft` — compiles `[show]` frames, others are made one-shot
- `short` — compiles all the frames in one-shot, `[show]` is ignored
- `brief` — compiles only `[show]` frames, others are blank

Thus, in terms of compilation time we get (assuming few `[show]` frames):

$$\texttt{brief} < \texttt{short} \leq \texttt{draft} < \texttt{final} \leq \texttt{ready}$$

Class options — `draft` and `brief`

Class options — `draft` and `brief`

The modes **`draft`** and **`brief`** are meant to help with development of slides.

Class options — `draft` and `brief`

The modes **`draft`** and **`brief`** are meant to help with development of slides.

For that purpose, they feature:

Class options — `draft` and `brief`

The modes **`draft`** and **`brief`** are meant to help with development of slides.

For that purpose, they feature:

1. A grid of coordinates on each **`[show]`** frame to help arrange the nodes.

Class options — `draft` and `brief`

The modes **`draft`** and **`brief`** are meant to help with development of slides.

For that purpose, they feature:

1. A grid of coordinates on each **`[show]`** frame to help arrange the nodes.
2. A slide counter in the bottom left of each **`[show]`** frame.

Class options — `draft` and `brief`

The modes **`draft`** and **`brief`** are meant to help with development of slides.

For that purpose, they feature:

1. A grid of coordinates on each **`[show]`** frame to help arrange the nodes.

2. A slide counter in the bottom left of each **`[show]`** frame.

   Hint: use this counter to synchronize `\bzIn{from}{to}{...}` arguments!

Class options — `draft` and `brief`

The modes **`draft`** and **`brief`** are meant to help with development of slides.

For that purpose, they feature:

1. A grid of coordinates on each **`[show]`** frame to help arrange the nodes.

2. A slide counter in the bottom left of each **`[show]`** frame.

Hint: use this counter to synchronize `\bzIn{from}{to}{...}` arguments!

Moreover, **`draft`** speeds-up the compilation process by skipping pictures.

Class options — `ready`

Class options — `ready`

The last thing to be explained is the `ready` class option.

Class options — `ready`

The last thing to be explained is the **`ready`** class option.

It works exactly as **`final`** except for the first **`bzPlainFrame`**:

Class options — `ready`

The last thing to be explained is the `ready` class option.

It works exactly as `final` except for the first `bzPlainFrame`:

```
\begin{bzPlainFrame}
...
\end{bzPlainFrame}
```

Class options — `ready`

The last thing to be explained is the **`ready`** class option.

It works exactly as **`final`** except for the first **`bzPlainFrame`**:

```
\begin{bzPlainFrame}
...
\end{bzPlainFrame}
```

Where it adds an additional `\bzOn{...}` with the time of the current recompilation, written on the bottom of the slide.

Class options — `ready`

The last thing to be explained is the `ready` class option.

It works exactly as `final` except for the first `bzPlainFrame`:

```
\begin{bzPlainFrame}
...
\end{bzPlainFrame}
```

Where it adds an additional `\bzOn{...}` with the time of the current recompilation, written on the bottom of the slide.

It serves two purposes:

Class options — `ready`

The last thing to be explained is the **`ready`** class option.

It works exactly as **`final`** except for the first **`bzPlainFrame`**:

```
\begin{bzPlainFrame}
...
\end{bzPlainFrame}
```

Where it adds an additional `\bzOn{...}` with the time of the current recompilation, written on the bottom of the slide.

It serves two purposes:

- Makes it easier for you to make sure which version you are to present.

Class options — `ready`

The last thing to be explained is the **`ready`** class option.

It works exactly as **`final`** except for the first **`bzPlainFrame`**:

```
\begin{bzPlainFrame}
...
\end{bzPlainFrame}
```

Where it adds an additional `\bzOn{...}` with the time of the current recompilation, written on the bottom of the slide.

It serves two purposes:

- Makes it easier for you to make sure which version you are to present.
- Allows you to test your slide-switcher without unravelling the next slide :)

Credits

Credits

The author would like to thank:

Credits

The author would like to thank:

- Till Tantau for his amazing work on Beamer and PGF/Ti*k*Z!

Credits

The author would like to thank:

- Till Tantau for his amazing work on Beamer and PGF/Ti*k*Z!
- Szczepan Hummel for suggestions and a prototype of `\bzNext{...}`,

Credits

The author would like to thank:

- Till Tantau for his amazing work on Beamer and PGF/Ti*k*Z!
- Szczepan Hummel for suggestions and a prototype of `\bzNext{...}`,
- Filip Mazowiecki and Michał Pilipczuk for never-ending enthusiasm,

Credits

The author would like to thank:

- Till Tantau for his amazing work on Beamer and PGF/Ti*k*Z!
- Szczepan Hummel for suggestions and a prototype of `\bzNext{...}`,
- Filip Mazowiecki and Michał Pilipczuk for never-ending enthusiasm,
- Abhishek Aich, Bartosz Bednarczyk, and Adam Gregosiewicz for feature requests and suggestions.