BEAMER-REVEAL

Walter Daems (walter.daems@uantwerpen.be)

Release 2026-02-08 — v1.09

1 Introduction

BEAMER [1] is a very powerful and convenient document class to create presentations and slides. However, integrating multimedia in it, is still a bit of a faff. On one side you have the integrated `multimedia` facility of BEAMER [1] and on the other side the `movie15` [2] and `media9` [3] packages. But unless you use the stock acrobat reader on Microsoft Windows, these things barely work, if at all.

Next to this LaTeX ecosystem for slides, you have the `reveal.js` framework [4], that allows easy integration of multimedia content. Why not combine both worlds? With that I mean:

- make your slides in BEAMER, using all the nice packages that come with LaTeX,
- include any browser-compatible multimedia content you'd like, even generate some LaTeX animations,
- convert that presentation to the reveal.js framework.

That's exactly what this package does.

Note that you are an *early adopter* when using this package. Expect frequent changes.

2 Rationale

Let's talk about the elephant in the room: why not work in the `reveal.js` framework directly, e.g. using Quarto [5]? I see two reasons:

- It avoids a new learning curve, allowing you to continue to build on your LaTeX and BEAMER expertise.
- It avoids that you have to convert your thousands of slides into a new format.

For me, those are all the reasons I need.

3 Synopsis

Figure 1 shows the overall flow. You start by making slides (frames) the usual way using the `beamer` class. Your source file (`slides.tex`) uses the package `beamer-reveal.sty` and references the multimedia files of your choice. These multimedia files (e.g. videos) can be super-imposed on any slide you like. Your favorite LaTeX-compiler typesets your slides to PDF and produces an auxiliary `.rvl`-file containing extra information for the conversion script. The conversion script `beamer-reveal.pl` fuses the PDF and the auxiliary file into a reveal.js presentation, using the templates provided by the BEAMER-REVEAL package. In fact, this is done by converting your slides to `JPG` format and using them as the background on the REVEAL-slides. The multimedia content appears as HTML5 elements on that background.
Notes are also converted to `JPG` format and available in the speaker view that is based on notes plugin of `reveal.js`.
You then can use your favorite browser as your viewer instead of the good old PDF reader.

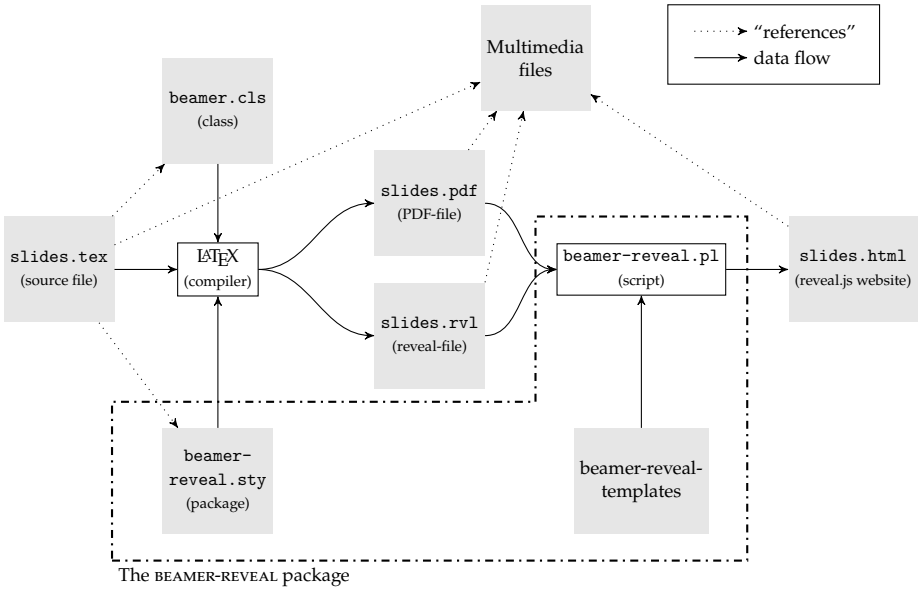


Figure 1: Workflow imposed by the BEAMER-REVEAL package

As a nice bonus, the BEAMER-REVEAL package allows you to generate LaTeX animations. It's not on the level of Manim [6], but for shorter animations inlined in your deck of slides it is more than functional!

Finally, you can also include material in iframes. Iframes are HTML constructs that act as a mini-browser, allowing to incorporate multiple multimedia blocks in HTML. As an example, this allows incorporating asymptote material that has been generated for displaying as HTML using webGL. Note that in many cases, your browser will prohibit these iframes to load external web content. To solve this, you can run a local webserver and access your presentation through it, or you can embed the iframe in the main HTML file. This is explained in Section 7.

4 Quirks

Combining BEAMER and REVEAL posed one major challenge: how to make sure that the HTML5 elements appear exactly where you want them to be, i.e. perfectly aligned with your LaTeX content as it has been converted to PDF. The core of the problem is twofold:

- In BEAMER the aspectratio of the slides is determined by the class option `aspectratio`. Your PDF viewer uses letterboxing (black bars on the side) if the aspectratio of your presentation does not correspond to the aspectratio of your screen. To the other hand, REVEAL puts your slides fullscreen without letterboxing, and therefore the aspectratio is determined by the screen resolution.
- Given the vector-nature of LaTeX and PDF, resolution is not a parameter you normally care about (everything is vector graphics anyway), while given the pixel-based nature of traditional multimedia files, resolution is an issue that you need to think about carefully.

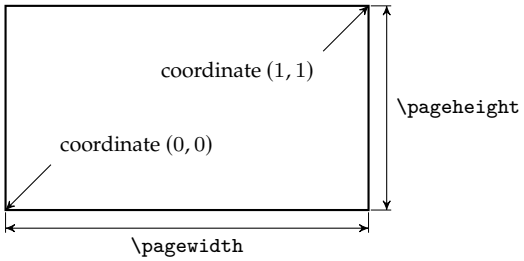
There is only one good solution: ask the users to (1) set the correct aspectratio using the beamer class options and (2) specify the width or height of their displaying screen using a BEAMER-REVEAL class option.

The latter ensures that (a) the background content and the multimedia files that are generated based on the LaTeX source, are generated with sufficient resolution, and that (b) the file sizes stay within acceptable limits.

Actually, there is a one more problem, and that is that the canvas size and aspectratio of the display area in your browser is dependent on whether you are viewing full screen or not. Therefore, an extra requirement arises: (3) when presenting a slide-deck only you must use your browser in full-screen mode. Otherwise, the alignment is off.

In order not to have to care about resolutions or the actual `\pagewidth` or `\pageheight` of your presen-

tation, I've chosen to impose another constraint: (4) if you want to add multimedia material in overlay mode, the user must specify locations on the screen (where to put the multimedia content) as relative fractions (x, y) where $(0, 0)$ corresponds to the bottom left of the screen and $(1, 1)$ corresponds to the top right of the screen.



However, if you have to specify width and heights of the multimedia boxes that you want to overlay on your slides, you don't want to specify width and height as relative numbers (of the slide width and height). In that scenario, displaying a 16:9 video on a 16:10 slide with a width equal to a quarter of the slide width, would require you to specify: `width=0.25,height=0.225`, which is weird. In addition, if you'd have to present at a venue where the projector has a different aspectratio from the one you anticipated, you would be forced to recalculate all the widths and heights of every video or image. You don't want that. Presenting on itself already brings sufficient stress without that extra worry.
Therefore, I imposed a fifth constraint: (5) the user must always specify either width or height of the box (as a fraction of the respective slide width or height) together with the aspectratio. Given width and aspectratio the BEAMER-REVEAL package can correctly determine its relative height; likewise height and aspectratio can be converted to the correct relative width. Therefore, it is illegal to specify both width and height at the same time. As the package knows the aspectratio of the screen (set correctly by the user) no recalculations need to be done. In the example above, that would mean specifying: `width=0.25,aspectratio=16/9`, which is very logical.

When confronted with the situation where you need to present on an old `4:3` projector, instead of the `16:10` you are used to, this allows you to just change the beamer aspectratio to `43`, recompile, rerun `beamer-reveal.pl` and you are good to go on stage.

So summarized, these are the five rules to go by:

1. Set the correct aspectratio as a beamer class option.
2. Specify the X or Y-resolution of your displaying screen as a package option to the BEAMER-REVEAL package.
3. Always put your browser in full-screen-mode when presenting.
4. Specify positions relative to the slide width and height, $(0, 0)$ being bottom left and $(1, 1)$ being top right.
5. Specify box sizes of the HTML5 content always as the width or length in combination with the aspectratio.

One final remark: you can put multimedia boxes on your slides in two modes:

overlay mode in this mode the box is put on your slide like tikz puts an image in overlay mode on a document: it does not consume any space on the slide;

insert mode in this mode the box is actually typeset by LaTeX on your slide. The actual content (video, image, …) will not be there, but the space will be consumed.

Note that for both modes — to work properly — your compiler must be run twice before running `beamer-reveal.pl`.

5 Portability

These class files should be ready to use with all common modern LaTeX compilers that produce PDF (pdfLaTeX, XeLaTeX, LuaLaTeX, …) from the major TeX-distributions (TeTeX, TexLive, MikTeX). If you experience problems with one of these, please inform the author.

The script `beamer-reveal.pl` is a Perl script, that works on all major platforms (UNIX, Linux, BSD, Debian, MS-Windows, MAC-OS, …). It makes use of the Poppler library and its `pdftoppm` tool, which is also available for those platforms. In case you want to use LaTeX animations, it also uses your very own favorite LaTeX-compiler, `pdfcrop` (which is part of the major TeX-distributions) and `ffmpeg` a well-known video conversion tool.

If these tools are available on your platform, then all should be well.

6 Installing the `beamer-reveal.pl` script

6.1 On Linux-like operating systems

Open a terminal with a shell. Below '$' represents your shell prompt. If needed, update your package list first. Then install the required tools:

- on a Debian-like OS:
```
$ sudo apt install -y perl cpanminus poppler-utils \
  texlive-extra-utils ffmpeg
```
- on a Redhat-like OS:
```
$ sudo dnf install -y epel-release
$ sudo dnf install -y \
  https://download1.rpmfusion.org/free/el/rpmfusion-free-release-$(rpm -E %rhel).noarch.rpm
$ sudo dnf install -y perl perl-App-cpanminus poppler-utils \
  texlive-pdfcrop ffmpeg
```
- on openSUSE:
```
$ sudo zypper install -y perl perl-App-cpanminus \
  poppler-tools texlive-pdfcrop ffmpeg
```
- on an Arch-like OS:
```
$ sudo pacman -Syu --needed perl cpanminus poppler \
  texlive-binextra ffmpeg
```
- on Alpine:
```
$ sudo apk add perl cpan-app-cpanminus poppler-utils \
  texlive-extra-utils ffmpeg
```
- on macOS:
```
$ brew install perl cpanminus poppler texlive ffmpeg
```

6.2 On MS windows operating systems

Open a powershell. Below '$' represents the powershell prompt. Install the required tools as follows:

```
$ winget install StrawberryPerl.StrawberryPerl
$ winget install -e --id oschwartz10612.Poppler
$ winget install MiKTeX.MiKTeX
$ winget install Gyan.FFmpeg
$ winget install --id Python.Python.3 --source winget
```

You probably need to answer 'Yes' quite some times. If you're asked to restart, you only need that before moving over to section 6.4 'Checking your setup'. You probably already have MiKTeX installed. In that case you can skip that. However, make sure the 'pdfcrop' program (part of MiKTeX) is installed and available on your path.

6.3 Install BEAMER-REVEAL

Open a shell on Linux/macOS, or a powershell prompt on MS-Windows. Run the following command at the prompt:

```
$ cpanm BeamerReveal
```

This will take a while as this needs to install a number of extra Perl libraries that have been used in BEAMER-REVEAL. You can use the same command to update the script if a new release has been made.

The number of the Perl package is: `20260208.1203`

You will see that number appear during the installation. Any version more recent version number than this one will work. New releases will fix bugs that are reported. If there is a breaking change in the Perl package, a new CTAN update will be released as well.

6.4 Checking your setup

Open a shell on Linux/macOS, or a command line or powershell prompt on MS-Windows. Then see whether you can invoke the help information for the tools.

```
$ cpanm -h
$ pdftoppm -h
$ pdfcrop --help
$ ffmpeg -h
```

To test BEAMER-REVEAL on MS-Windows, run:

```
$ beamer-reveal -h
```

On any other platform, run:

```
$ beamer-reveal.pl -h
```

If they all display correct help info, you're good to go.

7 Using the `beamer-reveal.pl` script

If you prepared your BEAMER-presentation according to Section 12, then converting it into a reveal.js HTML presentation, is as simple as running the following command. Make sure the current directory of the shell is the directory your LaTeX-source file and your compiled PDF-file is in. For all commands below, stay in that same working directory!

If you document is called `jobname.tex`, then just run:

```
$ lualatex jobname.tex
$ lualatex jobname.tex
```

On MS-Windows, run:

```
$ beamer-reveal jobname
```

On any other platform, run:

```
$ beamer-reveal.pl jobname
```

Conversion is very fast. Of course, if you need to render some LaTeX animations, it may take a little more than a jiffy. Especially if you are working on MS-Windows, because there the generation is fully single-threaded.

Some side notes:

- you need to run `beamer-reveal.pl` from the same working directory as your LaTeX-compiler; the script needs to be able to access your source files from the same viewpoint as your compiler.
- `beamer-reveal.pl` also has some convenient command-line options to set the `aux` directory (that contains your `.rvl` file), the `pdf` directory that is the output directory of your LaTeX-run and the

`output` directory (in which it will place the resulting reveal site). This allows easy integration with some build frameworks such as `latexmk`. Checkout the help message to learn about it.
`$ beamer-reveal.pl -h`

Next, you can load your document in your browser, e.g.:

```
$ firefox jobname.html
```

If your presentation contains iframe content, you need to start a local webserver. You don't need a network connection for that. This is how:

```
$ python -m http.server
```

Then, you can access it through: http://localhost:8000.

As of v1.06 there is an alternative, and that is to make the iframe element 'embedded'. This will cause the iframe content to live as Base64-encoded ASCII string in the main `\jobname.html` file. Therefore it will be trusted by your browser and will load without any complaint.

8 Demo

If you want to see and try out the result of the example that is part of this documentation, check:

- a 16-by-9 version on: https://www.digmanwaves.net/beamer-reveal/169
- a 16-by-10 version on: https://www.digmanwaves.net/beamer-reveal/1610

9 Open issues

Below, you'll find a list of open issues. I appreciate any input I can get on these matters.

- Regarding browsers, I'm a big fan of Firefox. However, I must admit that the Chrome/chromium line does a better job regarding smooth slide transitions. I can't seem to get preloading to work in Firefox browser, while in Chrome/chromium, it just works out of the box.
- I did not yet see any solutions to perform robust letterboxing on a browser (i.e. putting black stripes on the top and bottom if you are displaying a 16:9 presentation on a 16:10 screen). With robust, I mean a solution in which you don't have to change the content of your presentation html-file.

10 Reach out

Is there any feature you are missing? Some problems you encounter? Some inconsistencies in the interface or the documentation? Some additional features that Reveal supports, but are not in BEAMER-REVEAL? Let me know by dropping me an e-mail.

If you think BEAMER-REVEAL makes no sense, let me know why you think so. I'm keen to learn.

On the other hand, if you like BEAMER-REVEAL and are using it, just send me a kind word. It keeps me going way better than wine or pizza.

11 Thanks

Thanks to Paul Levrie for proofreading this documentation and testing the package.

12 Usage

12.1 Package options

12.1.1 Regarding dimensions

The following package options are available:

`width`	the width (in pixels) of the screen you will display the presentation on
`height`	the height (in pixels) of the screen you will display the presentation on

Only specify one of the two options. The other dimension will be deduced from the `aspectratio` option that passed onto the beamer class.
Higher values will give higher resolution of the slides (in the background), but also larger file sizes. A safety factor is already used when converting the slides to jpg-format, so taking the true height or width is recommended. Only when you are bothered with jpg-artifacts in the final result, you should consider increasing the width- or height-value.

12.1.2 Reveal specific stuff

The following options are opening up some cool stuff that reveal enables:

`embed`	this causes all graphical material in your slides (`.jpg`, `.png`, `.mp4`, `.ogg`, `.wav`, …) including the slide backgrounds and the notes slides to be embedded as Base64-encoded ASCII string into the HTML. Note however that this still is not the case for animations, stills and the `reveal.js` boilerplate javascripts.
`autoslide=<int>`	This causes your presentation to progress automatically, going from slides to slide every `<int>` milliseconds. You can override this value on an individual basis per slide, by providing the same option to the beamer `frame` environment.
`loop`	This causes your presentation to loop; this makes sense if you *autoslide* it, enabling to run your presentation in *kiosk* mode.

12.2 An enhanced frame environment

`frame` (*env.*) The frame environment is defined by BEAMER. However, it is equipped with four new environment options by the BEAMER-REVEAL package:

`titleslide`	specifies that this slide is a title slide. It will be on the top level of the reveal menu. This menu can be invoked by pressing 'm' in your presentation (or clicking on the pan-cake icon on the bottom left of your presentation).
`sectionslide`	specifies that this slide is a section slide. It will be on the second level of the reveal menu.
`subsectionslide`	specifies that this slide is a subsection slide. It will be on the third level of the reveal menu.
`transition`	one of `none` (default), `fade`, `zoom`, `concave` or `convex`; these correspond to the available reveal.js transitions. Not recommended for use. Why? Animation without purpose is bad practice.
`embed`	This will cause the slide background, the notes and all media elements on your slide to be embedded as a Base64-encode ASCII string in your HTML file.
`autoslide=<int>`	This will set the individual time for this slide to `<int>` milliseconds. This option only takes effect if you set the package option `autoslide` as well. If you set the value to 0, the presentation wil stop the autoslide mode at that

slide. If your slide contains a video or audio fragment, or an animation that plays for a specific aumount of time, then this time will override the value you specify here. The slide show will only progress as soon as the animation or video/audio has finished!

Slides that are not titleslides, sectionslides or subsectionslides are ordinary slides. They will appear on the lowest level in the reveal menu.

12.3 The macros to use inside the frame environment

To understand the operation of the macros, it is important to realize that the slide content of your beamer-generated PDF will be put as a background image onto the reveal.js slides. The extra material such as videos, images and audio will be put as an overlay on top of that background. The macros allow you to define what material to put in overlay and where and how big it should appear. All macros take options. Some of the options need a value, some not. Options that take no value are marked with a dagger-symbol in superscript (†).

`\video` This macro will create a video box on the current slide. Of course a video box can cover the entire slide if desired.

Its syntax is:

- overlay mode:
  `\video<overlay-spec>[options] (nodename) \at (x,y) { filename }`
- insert mode:
  `\video<overlay-spec>[options] (nodename) { filename }`

The arguments are:

`overlay-spec` a standard beamer overlay specification that allows you to determine on which overlays the video is to appear; this argument will end up in a traditional `\only<>{}` clause that beamer provides.

`options` the following keys are available. In general they require a value: `key=value`.

Size options:

`width`	the width of the video box (a fraction relative to the width of the slide)
`height`	the height of the video box (a fraction relative to the height of the slide)
`aspectratio`	the aspectratio of the video box

Remember that you never specify both `width` and `height`, only one of those two in combination with the `aspectratio`.

Placement options:

`anchor`	value is one of `center`, `north`, `west`, `south`, `east`, `north east`, `north west`, `south east`, `south west`; this specifies where the anchor of the video is positioned; the anchor will be positioned at `(x,y)`
`above`†	synonym to `anchor=south`
`below`†	synonym to `anchor=north`
`left`†	synonym to `anchor=east`
`right`†	synonym to `anchor=west`
`above left`†	synonym to `anchor=south east`
`above right`†	synonym to `anchor=south west`
`below left`†	synonym to `anchor=north east`
`below right`†	synonym to `anchor=north west`

Appearance and sound options:

`fit`	the way the video should occupy the box: `fill`, `cover` or `fit`
`background`	the color of the background of the box
`draw`†	generates an outline around the box that allows you to inspect where the video will end up in your PDF-file
`autoplay`†	causes the video to start playing a soon as it appears on the slide
`controls`†	causes player controls to appear below your video
`loop`†	causes the video to run in loop mode
`muted`†	silences the audio of the player

Various options (only one for now):

`embed`†	causes the content of the video-file to be embedded as a Base64-encoded ASCII string in the main html file the script produces. I don't recommend this for video files.

`nodename` (optional) name to assign to the rectangular node corresponding to the bounding box of the video, such that you can later refer to that node (and its derived anchors (e.g. as `'(nodename.south)'`) to position other material.

`x, y` the x- and y-coordinate of the box, specified as a fraction of the slide width and slide height (dimensionless), fixing the anchor of the box, w.r.t. the bottom left of your slide.

`filename` a filename or URL that leads to the video file (e.g. an mp4 file). Any file playable by your browser will work.

Note that you can pause and (re)start your video by pressing 'v' when presenting your slide deck.

`\audio` This macro will create an audio block on the current slide. This block is rather an abstract concept, unless you activate the controls of the player. Indeed, the audio block will be invisible unless you specify the option to display the controls of the player.

Its syntax is:

- overlay mode:
  `\audio<overlay-spec>[options] (nodename) \at (x,y) { filename }`
- insert mode:
  `\audio<overlay-spec>[options] (nodename) { filename }`

The arguments are:

`overlay-spec` a standard beamer overlay specification that allows you to determine on which overlays the audio is to appear; this argument will end up in a traditional `\only<>{}` clause that beamer provides.

`options` the following keys are available. In general they require a value: `key=value`.

Size options:

`width`	the width of the audio box (a fraction relative to the width of the slide)
`height`	the height of the audio box (a fraction relative to the height of the slide)
`aspectratio`	the aspectratio of the audio box

Placement options:

`anchor`	value is one of `center`, `north`, `west`, `south`, `east`, `north east`, `north west`, `south east`, `south west`; this specifies where the anchor of the audio box is positioned; the anchor will be positioned at `(x,y)`
`above`†	synonym to `anchor=south`

`below`†	synonym to `anchor=north`
`left`†	synonym to `anchor=east`
`right`†	synonym to `anchor=west`
`above left`†	synonym to `anchor=south east`
`above right`†	synonym to `anchor=south west`
`below left`†	synonym to `anchor=north east`
`below right`†	synonym to `anchor=north west`

Appearance and sound options:

`fit`	the way the audio block should occupy the box: `fill`, `cover` or `fit`
`background`	the color of the background of the box
`draw`†	generates an outline around the box that allows you to inspect where the audio box will end up in your PDF-file
`autoplay`†	causes the audio to start playing a soon as it appears on the slide
`controls`†	causes player controls to appear
`loop`†	causes the audio to run in loop mode
`muted`†	silences the audio of the player

Various options (only one for now):

`embed`†	causes the content of the audio-file to be embedded as a Base64-encoded ASCII string in the main html file the script produces. I don't recommend this for audio files.

`nodename` (optional) name to assign to the rectangular node corresponding to the bounding box of the audio, such that you can later refer to that node (and its derived anchors (e.g. as '`(nodename.south)`') to position other material.

`x,y` the x- and y-coordinate of the box, specified as a fraction of the slide width and slide height (dimensionless), fixing the anchor of the box, w.r.t. the bottom left of your slide.

`filename` a filename or URL that leads to the audio file (e.g. an mp3 or ogg vorbis file). Any file playable by your browser will work.

Note that you can pause and (re)start your audio fragment by pressing 'a' when presenting your slide deck.

`\iframe` This macro will create an iframe box on the current slide. Of course an iframe box can cover the entire slide if desired.

Its syntax is:

- overlay mode:
  `\iframe<overlay-spec>[options] (nodename) \at (x,y) { filename }`
- insert mode:
  `\iframe<overlay-spec>[options] (nodename) { filename }`

The arguments are:

`overlay-spec` a standard beamer overlay specification that allows you to determine on which overlays the iframe is to appear; this argument will end up in a traditional `\only<>{}` clause that beamer provides.

`options` the following keys are available. In general they require a value: `key=value`.

Size options:

`width`	the width of the iframe box (a fraction relative to the width of the slide)
`height`	the height of the iframe box (a fraction relative to the height of the slide)

`aspectratio` the aspectratio of the iframe box

Remember that you never specify both `width` and `height`, only one of those two in combination with the `aspectratio`.

Placement options:

`anchor`	value is one of `center`, `north`, `west`, `south`, `east`, `north east`, `north west`, `south east`, `south west`; this specifies where the anchor of the video is positioned; the anchor will be positioned at `(x,y)`
`above`	synonym to `anchor=south`;
`below`	synonym to `anchor=north`;
`left`†	synonym to `anchor=east`;
`right`†	synonym to `anchor=west`;
`above left`†	synonym to `anchor=south east`;
`above right`†	synonym to `anchor=south west`;
`below left`†	synonym to `anchor=north east`;
`below right`†	synonym to `anchor=north west`;

Appearance options:

`fit`	the way the iframe should occupy the box: `fill`, `cover` or `fit`
`background`	the color of the background of the box
`draw`	generates an outline around the box that allows you to inspect where the iframe will end up in your PDF-file;[0]

Various options (only one for now):

`embed`†	causes the content of the iframe html-file to be embedded as a Base64-encoded ASCII string in the main html file the script produces. For iframe material, I think this makes sense if you want to avoid using an auxiliary webbrowser (e.g. if your are presenting using someone else's computer).

`nodename` (optional) name to assign to the rectangular node corresponding to the bounding box of the iframe, such that you can later refer to that node (and its derived anchors (e.g. as `'(nodename.south)'`) to position other material.

`x, y` the x- and y-coordinate of the box, specified as a fraction of the slide width and slide height (dimensionless), fixing the anchor of the box, w.r.t. the bottom left of your slide.

`filename` a filename or URL that leads to the iframe content (e.g. an HTML file generated by asymptote). Note that for the content to work, you might need to serve your presentation through a local html server:

```
$ python -m http.server
```

`\image` This macro will create an image box on the current slide. Of course an image box can cover the entire slide if desired.

Note that using the `\includegraphics` command of the `graphicx` package is still preferred to include the standard image formats, such as PDF, PNG, TIFF and JPG. However, the `image` command also allows to include (animated) GIFs, webp and SVG files.

Its syntax is:

- overlay mode:
  ```
  \image<overlay-spec>[options] (nodename) \at (x,y) { filename }
  ```
- insert mode:
  ```
  \image<overlay-spec>[options] (nodename) { filename }
  ```

The arguments are:

`overlay-spec` a standard beamer overlay specification that allows you to determine on which overlays the image is to appear; this argument will end up in a traditional `\only<>{}` clause that beamer provides.

`options` the following keys are available. In general they require a value: `key=value`.

Size options:

`width`	the width of the image box (a fraction relative to the width of the slide)
`height`	the height of the image box (a fraction relative to the height of the slide)
`aspectratio`	the aspectratio of the image box

Remember that you never specify both `width` and `height`, only one of those two in combination with the `aspectratio`.

Placement options:

`anchor`	value is one of `center`, `north`, `west`, `south`, `east`, `north east`, `north west`, `south east`, `south west`; this specifies where the anchor of the image is positioned; the anchor will be positioned at `(x,y)`
`above`†	synonym to `anchor=south`;
`below`†	synonym to `anchor=north`;
`left`†	synonym to `anchor=east`;
`right`†	synonym to `anchor=west`;
`above left`†	synonym to `anchor=south east`;
`above right`†	synonym to `anchor=south west`;
`below left`†	synonym to `anchor=north east`;
`below right`†	synonym to `anchor=north west`;

Appearance options:

`fit`	the way the image should occupy the box: `fill`, `cover` or `fit`
`background`	the color of the background of the box
`draw`	generates an outline around the box that allows you to inspect where the image will end up in your PDF-file;[0]

Various options (only one for now):

`embed`†	causes the content of the image-file to be embedded as a Base64-encoded ASCII string in the main html file the script produces. I don't recommend this for image files.

`nodename` (optional) name to assign to the rectangular node corresponding to the bounding box of the image, such that you can later refer to that node (and its derived anchors (e.g. as '`(nodename.south)`') to position other material.

`x`, `y` the x- and y-coordinate of the box, specified as a fraction of the slide width and slide height (dimensionless), fixing the anchor of the box, w.r.t. the bottom left of your slide.

`filename` a filename or URL of the image file (e.g. a GIF file).

`\animation` This macro will create an animation box on the current slide. Different from the other boxes, this box will determine its own width and height, based on the dimensions of the LaTeX content embedded in it. In fact, it is illegal to specify aspectratio, width or height.

The animation is generated as follows: the content of macro will be written by the `beamer-reveal.pl` script to a separate LaTeX file using the `standalone` class. The preamble that it uses will be the part of the preamble in your beamer source file, in between the loading of the `beamer-reveal` package and the line containing `\begin{document}`. The animation block will be embed-

ded in a loop that will be executed `duration` · `framerate` times, providing a macro `\progress` that contains a fraction that goes up from 0 (first iteration) to 1 last iteration. The PDF-file that is generated with this standalone file, is converted to an mp4-file that is included as a video on your slide.
The tools used for this are the LaTeX-compiler you used for your beamer sourcefile, `pdfcrop` [7], `pdftoppm` [8] and `ffmpeg` [9].

This animation generation takes advantage of the multicore nature of your computer, by simple, but smart parallelization (on non-MS-Windows operating systems).

Its syntax is:

- overlay mode:
  `\animation<overlay-spec>[options] (nodename) \at (x,y) { content }`
- insert mode:
  `\animation<overlay-spec>[options] (nodename) { content }`

The arguments are:

`overlay-spec` a standard beamer overlay specification that allows you to determine on which overlays the animation is to appear; this argument will end up in a traditional `\only<>{}` clause that beamer provides.

`options` the following keys are available. In general they require a value: `key=value`.

Size options: no size options; size is determined from the LaTeX code itself!

Generation options:

`framerate`	number of frames per second that the animation should contain.
`duration`	duration (in seconds) of the animation
`progress`	tokenlist shaped as `{a|handout:n1/b1,n2/b2,...}` with `a` the progress value for the PDF, and `b1`, `b2`, … the progress values for handouts `n1`, `n2`, …. This assumes that the overlay-spec of the `\animation` macro also specifies to generate the required handout slide numbers. Take a look at the example in this manual for more clarity on this. If the `handout:` section is missing, the `pdf progress` value will be used for the/all handout(s).
`expand once`	will cause the content to be expanded once; use this if you described your animation in a macro and want to expand that macro.

The number of frames that will be generated for the animation is:

$$\texttt{number-of-frames} = \texttt{framerate} \cdot \texttt{duration} \tag{1}$$

Your PDF file will contain a single shot of the animation (as if it were a preview shot). For this frame, the value of the `\progress` macro will be set to `pdf progress`.

Placement options:

`anchor`	value is one of `center`, `north`, `west`, `south`, `east`, `north east`, `north west`, `south east`, `south west`; this specifies where the anchor of the animation is positioned; the anchor will be positioned at `(x,y)`
`above`†	synonym to `anchor=south`
`below`†	synonym to `anchor=north`
`left`†	synonym to `anchor=east`
`right`†	synonym to `anchor=west`
`above left`†	synonym to `anchor=south east`
`above right`†	synonym to `anchor=south west`
`below left`†	synonym to `anchor=north east`

`below right`†	synonym to `anchor=north west`

Appearance options:

`background`	the color of the background of the box
`draw`†	generates an outline around the box that allows you to inspect where the video will end up in your PDF-file
`autoplay`†	causes the video to start playing a soon as it appears on the slide
`controls`†	causes player controls to appear below your video
`loop`†	causes the animation to run in loop mode

`nodename` (optional) name to assign to the rectangular node corresponding to the bounding box of the animation, such that you can later refer to that node (and its derived anchors (e.g. as '`(nodename.south)`') to position other material.

`x, y` the x- and y-coordinate of the box, specified as a fraction of the slide width and slide height (dimensionless), fixing the anchor of the box, w.r.t. the bottom left of your slide.

`content` the LaTeX code that generates every frame based on the `\progress` 'time variable'. Any macros used in this content need to be defined in the content itself or in the preamble of your slide deck in between the `\usepackage{beamer-reveal}` and the `\begin{document}` as this portion is also used when generation the animation. If that is not the case, you can specify an extra option `expand once` to perform a single expansion, such that the macros and environments on the top level are expanded once. The example contained in this manual shows you an example of this.

Note that you can pause and (re)start your animation fragment by pressing 'v' (as it is in fact a video) when presenting your slide deck.

`\still` This macro will create the still (of an animation) box on the current slide. The idea is that you can split your animation in parts and separate these parts with a slide (or a subframe, if you are using multiple overlays) containing a still corresponding to the point between the animation parts. Just as the animation box, this box will determine its own width and height, based on the dimensions of the LaTeX content embedded in it. In fact, it is illegal to specify aspectratio, width or height.

The still is generated as follows: the content of macro will be written by the `beamer-reveal.pl` script to a separate LaTeX file using the `standalone` class. The preamble that it uses will be the part of the preamble in your beamer source file, in between the loading of the `beamer-reveal` package and the line containing `\begin{document}`. The animation block will be embedded in an environment that sets the macro `\progress` to a value in between 0 and 1. The PDF-file that is generated with this standalone file, is converted to a jpg-file that is included as a still image on your slide.
The tools used for this are the LaTeX-compiler you used for your beamer sourcefile, `pdfcrop` [7] and `pdftoppm` [8].

Its syntax is:

- overlay mode:
  `\still<overlay-spec>[options] (nodename) \at (x,y) { content }`
- insert mode:
  `\still<overlay-spec>[options] (nodename) { content }`

The arguments are:

`overlay-spec` a standard beamer overlay specification that allows you to determine on which overlays the still is to appear; this argument will end up in a traditional `\only<>{}` clause that beamer provides.

`options` the following keys are available. In general they require a value: `key=value`.

Size options: no size options; size is determined from the LaTeX code itself!

Generation options:

`progress` tokenlist shaped as `{a|handout:n1/b1,n2/b2,...}` with `a` the progress value for the PDF, and `b1`, `b2`, … the progress values for handouts `n1`, `n2`, …. This assumes that the overlay-spec of the `\animation` macro also specifies to generate the required handout slide numbers. Take a look at the example in this manual for more clarity on this. If the `handout:` section is missing, the `pdf progress` value will be used for the/all handout(s).

`expand once` will cause the content to be expanded once; use this if you described your still in a macro and want to expand that macro.

Your PDF file will contain the image typeset by your LaTeX run. In the reveal version of the presentation, the image will be overlaid by the generated still, such that it aligns perfectly with a corresponding video on an adjacent slide.

Placement options:

`anchor` value is one of `center`, `north`, `west`, `south`, `east`, `north east`, `north west`, `south east`, `south west`; this specifies where the anchor of the animation is positioned; the anchor will be positioned at `(x,y)`

`above`† synonym to `anchor=south`

`below`† synonym to `anchor=north`

`left`† synonym to `anchor=east`

`right`† synonym to `anchor=west`

`above left`† synonym to `anchor=south east`

`above right`† synonym to `anchor=south west`

`below left`† synonym to `anchor=north east`

`below right`† synonym to `anchor=north west`

Appearance options:

`background` the color of the background of the box

`draw`† generates an outline around the box that allows you to inspect where the video will end up in your PDF-file

`nodename` (optional) name to assign to the rectangular node corresponding to the bounding box of the animation, such that you can later refer to that node (and its derived anchors (e.g. as '`(nodename.south)`') to position other material.

`x`, `y` the x- and y-coordinate of the box, specified as a fraction of the slide width and slide height (dimensionless), fixing the anchor of the box, w.r.t. the bottom left of your slide.

`animation-LaTeX-content` the LaTeX code that generates every frame based on the `\progress` 'time variable'. Any macros used in this content need to be defined in the content itself or in the preamble of your slide deck in between the `\usepackage{beamer-reveal}` and the `\begin{document}` as this portion is also used when generation the still. If that is not the case, you can specify an extra option `expand once` to perform a single expansion, such that the macros and environments on the top level are expanded once. The example contained in this manual shows you an example of this.

12.4 Short overview of the options for the different media macros

Most of the macros have very similar options, but some are not common to all of them. Therefore, I made a small overview. I admit, mostly to help myself to keep the overview while developing the package.

	Size	Placement	Appearance and sound							Various	Generation			
Macro	`w` / `h` / `ar`	`anchor`	`fit`	`background`	`draw`	`autoplay`	`controls`	`loop`	`muted`	`embed`	`framerate`	`duration`	`progress`	`expand once`
`\video`	•[1]	•	•	•	•	•	•	•	•	•	×	×	×	×
`\audio`	•[1]	•	•	•	•	•	•	•	•	•	×	×	×	×
`\iframe`	•[1]	•	•	•	•	×	×	×	×	•	×	×	×	×
`\image`	•[1]	•	•	•	•	×	×	×	×	•	×	×	×	×
`\animation`	×	•	×	•	•	•	•	•	×	•	•	•	•	•
`\still`	×	•	×	•	•	×	×	×	×	•	×	×	•	•

[1] `aspectratio` and `width`, or `aspectratio` and `height`, never `width` and `height` together

13 Example

13.1 Using the example

An example will allow you to get an idea of the convenience of the package. The following steps will allow you to observe it in your browser:

```
$ lualatex beamer-reveal-example.tex
$ beamer-reveal.pl beamer-reveal-example
$ python -m http.server
```

Then open in your browser: localhost:8000 and enjoy!

13.2 The source code of the example

```
<*example>
\documentclass[11pt,aspectratio=169,t]{beamer}%
\setbeamertemplate{navigation symbols}{}

\usepackage[width=1920,autoslide=4000]{beamer-reveal}
\usepackage{tikz}
\usepackage{siunitx}

\newcommand\eu{\mathrm{e}}
\newcommand\ju{\mathrm{j}}

\title{Test slide deck}
\subtitle{\textsc{beamer-reveal}}
\author{Walter Daems}

\setbeamertemplate{note page}{\insertnote}

\begin{document}

\begin{frame}[titleslide]
  \titlepage
\end{frame}

\AtBeginSection{
  \begin{frame}[sectionslide]{Overview}
    \tableofcontents[currentsection]
  \end{frame}
}
\AtBeginSubsection{
  \begin{frame}[subsectionslide]{Overview}
    \tableofcontents[currentsection,currentsubsection]
  \end{frame}
}

\section{Introduction}
\subsection{Slide making}

\begin{frame}
  {Good news}
  {}
  You can keep on making your slides the way you are used to!
  \begin{itemize}
  \item all the nice \LaTeX{} stuff at your fingertips
  \item no tempation to use too much unnecessary animation
  \end{itemize}
```

```
  \bigskip

  Indeed, there are no tools that can typeset equations like the tools form the \TeX-ecosystem:
  \begin{equation}
    \eu^{-\ju\pi}+1=0
  \end{equation}

  \note[item]{Make joke about fingertips}
  \note[item]{Don't speak about your personal temptations!}
\end{frame}

\begin{frame}[transition=concave,embed,autoslide=2000]
  {Pr\^et-\`a-porter}
  {A dummy slide}
  \vfill
  Showing off the 'concave' slide transition animation. Not recommended!
  \vfill
\end{frame}

\begin{frame}[transition=convex]
  {Très chique}
  {A dummy slide}
  \vfill
  Showing off the 'convex' slide transition animation. Not recommended!
  \vfill
\end{frame}

\subsection{Pimping your slides}

\begin{frame}
  {And even more good news}
  {\ldots almost seems to good to be true\ldots}
  \small
  However, now you can pimp your slides like never before. You can incorporate:
  \begin{itemize}
  \item videos and audio fragments
  \item animated GIFs and LaTeX animations
  \item iframe content
  \end{itemize}
  without being tied to Acrobat reader.
  In addition, there are some extra features
  \begin{itemize}
  \item press '?' for keyboard help, amongst which you will find:
  \item press 'm' to open the slide menu on the left
  \item press 'o' to get an overview of the slides
  \item press 's' to start a speaker view
  \item press 'g' to go to a specific slide by typing its slide number
  \end{itemize}
  The pancake menu on the bottom left also opens the menu.
\end{frame}

\note{
  Don't spend too much time on this slide
}

\begin{frame}[transition=zoom]
  {A dymmy slide}
  {number three}
  \vfill
  Showing off the 'zoom slide transition animation. Not recommended!
  \vfill
\end{frame}
```

```
\section{In detail}

\subsection{Candy for the eye}

\begin{frame}
  {Placing videos}
  {}
  \only<1>{On this first slide there is nothing to see. On the next animation frame, a video will appear.}
  \only<2>{Here it is!}
  \video<2>[above,draw,autoplay,height=0.7,aspectratio=16/9,
            background=yellow,fit=contain]
  (myvid) \at (0.5,0.1) {Media/beamer-reveal-testvideo.mp4}
  \tikz[remember picture,overlay] \node<2>[anchor=north,font=\tiny] at (myvid.south)
       {An example video (C) Walter Daems};
\end{frame}

\begin{frame}
  {Placing images (possibly animated)}
  {}
  \begin{columns}
    \column[T]{0.45\textwidth}
    Below you will find a png (for which you don't need reveal, BTW).
    \vspace*{1cm}

    Of course, you can exploit the transparency of the background layer in the PNG!
    \vspace*{2cm}

    And on the top right you will find a swinging pendulum (an animated GIF).
    \column[T]{0.45\textwidth}
    % this is a image whose box consumes area on your slide
    \image[width=0.33,aspectratio=1,fit=contain,draw] (myimage) {Media/beamer-reveal-AnimatedPendulum.gif}
  \end{columns}
  % this is an overlayed image (not consuming any space)
  \image[width=0.25,aspectratio=1,fit=contain] \at (0.1,0.6) {Media/beamer-reveal-WiresTp.png}
\end{frame}

\begin{frame}
  {Placing iframe material (possibly animated)}
  {e.g. generated with asymptote}

  Click and drag on the iframe below. You can manipulate it! Use your mouse
  scroll-wheel to zoom in or out.
  \iframe[draw,anchor=north,height=0.6,aspectratio=16/9,embed,
          fit=cover] \at (0.5,0.7) {Media/beamer-reveal-PCB.html}
\end{frame}

\note{
  Don't mention the amount of hours that went into this iframe!
}

\subsection{Resonance for the ear}

\begin{frame}
  {Adding audio to your slides}
  {}
  Below, there is an audio block
  that automatically starts playing.\\
  \audio[autoplay,controls,width=0.1,aspectratio=16/9,
    background=blue,fit=cover] {Media/beamer-reveal-AudioSample.ogg}
\end{frame}
```

```
\subsection{Make (video) animations with LaTeX}

\newcommand\myanimation{%
      \begin{tikzpicture}[font=\footnotesize,transform shape,scale=0.75]
        \pgfmathsetmacro\angle{\progress*540}%
        \clip (-2,-5.25) rectangle (8,2);
        \node[below left,inner sep=1pt] at (0,0) {\tiny 0};
        \node[below left,inner sep=1pt] at (2.5,0) {\tiny 0};
        \node[above right,inner sep=1pt] at (0,-2) {\tiny 0};

        \begin{scope}[every node/.style={right}]
          \node[thick,draw,rectangle] at (2.5,-2)
               {\large $x(t) = A \cdot \eu^{j\omega t}$};
               \node at (3.5,-3)
                     {\large $\eu^{j\alpha} = \cos\alpha+j\sin\alpha$};
                     \node at (2.5,-4)
                           {\large $x(t) = \underbrace{A \cos \omega t}_{\text{\textcolor{orange}{real}}}
                             + \underbrace{j A \sin \omega t}_{\text{\textcolor{olive}{imaginary}}}$};
        \end{scope}
        \draw[->,thick] (3,-2.4) -- (3,-3.4);
        \draw[blue,thick] (0,0) circle (1);

        \draw[->] (-1.25,0) -- (1.25,0) node[below] {Re};
        \draw[->] (0,-1.25) -- (0,1.25) node[left] {Im};

        % circle
        \draw[olive,very thick] (0,0) -- (0,{sin(\angle)});
        \draw[orange,very thick] (0,0) -- ({cos(\angle)},0);
        \draw[blue,thick,->] (0,0) -- node[left,font=\tiny] {A} +(\angle:1);
        \draw[->] (0.4,0) arc (0:\angle:0.4);
        \node at (0.5*\angle:0.7) {\scriptsize $\omega \tilde t$};

        % right graph
        \draw[very thick,olive] ({2.5+\angle/180},0) -- +(0,{sin(\angle)});
        \draw[densely dotted] ({min(0,cos(\angle))},{sin(\angle)})
        -- ({2.5+\angle/180},{sin(\angle)});
        \draw[thick] ({2.5+\angle/180},0) +(0,1pt) -- +(0,-1pt) node[below] {$\tilde t$};

        % bottom graph
        \draw[very thick,orange] (0,{-2-\angle/180}) -- +({cos(\angle)},0);
        \draw[densely dotted] ({cos(\angle)},{max(0,sin(\angle))})
        -- ({cos(\angle)},{-2-\angle/180});
        \draw[thick] (0,{-2-\angle/180}) +(1pt,0) -- +(-1pt,0) node[left] {$\tilde t$};

        % right graph
        \foreach \y/\l in {-1/-A,1/A} {
          \draw[gray,densely dotted] (2.5,\y) -- (6.25,\y);
          \draw (2.5,\y) +(+1pt,0) -- +(-1pt,0) node[left] {$\l$};
        }
        \draw[->] (2.0,0) -- (6.5,0) node[below] {$t$};
        \draw[->] (2.5,-1.25) -- (2.5,1.25) node[left] {$Im(x(t))$};
        \draw[olive,thick,domain=-0.25:3.5,samples=30,smooth] plot
        ({\x+2.5},{sin(pi*\x r)});

        % bottom graph
        \foreach \y/\l in {-1/-A,1/A} {
          \draw[gray,densely dotted] (\y,-2) -- (\y,-4.5);
          \draw (2.5,\y) +(+1pt,0) -- +(-1pt,0) node[left] {$\l$};
        }
        \draw[->] (-1.25,-2) -- (1.25,-2) node[above] {$Re(x(t))$};
        \draw[->] (0,-1.5) -- (0,-5) node[left] {$t$};
```

```
        \draw[orange,thick,domain=-0.25:2.6,samples=30,smooth] plot
        ({cos(pi*\x r)},{-2-\x});
      \end{tikzpicture}%
}

\begin{frame}
  {Making animations with LaTeX (using TikZ as example)}
  {It is easier than ever before}
  \small The animation content is exported to a standalone \LaTeX-document that creates a
  loop over it, for a \texttt{duration} seconds at
  \texttt{framerate} frames per second providing a \texttt{\textbackslash{}progress}
  variable that goes gradually from 0 to 1 in \texttt{duration} $\times$
  \texttt{framerate} frames. The beamer-reveal.pl script transforms it
  to mp4 maximally exploiting your multi-core CPU.

  \begin{center}
    \animation<1|handout:1-3>[framerate=25,duration=7.5,autoplay,
      progress={0.1|handout:1/0,2/0.2,3/0.5},expand once] {\myanimation}
  \end{center}
\end{frame}

\begin{frame}
  {Making animations with LaTeX (using TikZ as example)}
  {Another slide with just a 'still' - but with some music!}
  \small The animation content is exported to a standalone \LaTeX-document that creates a
  loop over it, for a \texttt{duration} seconds at
  \texttt{framerate} frames per second providing a \texttt{\textbackslash{}progress}
  variable that goes gradually from 0 to 1 in \texttt{duration} $\times$
  \texttt{framerate} frames. The beamer-reveal.pl script transforms it
  to mp4 maximally exploiting your multi-core CPU.

  \begin{center}
    \still[progress=0.75,expand once] {\myanimation}
  \end{center}
  \audio[width=0.1,aspectratio=16/9,fit=cover,autoplay] \at (0,0)
        {Media/beamer-reveal-AudioSample.ogg}
\end{frame}

\end{document}
</example>
```

14 Implementation

14.1 The preamble of the package

```
⟨*reveal⟩
\ExplSyntaxOn
\tl_new:N \g_@@_fileversion_tl
\tl_set:Nn \g_@@_fileversion_tl {v1.09}
\tl_new:N \g_@@_filedate_tl
\tl_set:Nn \g_@@_filedate_tl {2026-02-08}
\RequirePackage{l3keys2e}
⟨/reveal⟩
```

14.2 Error/warning messages

```
⟨*reveal⟩
\msg_new:nnn{ beamer-reveal } { inconsistent-dimensions } {
  aspect~ratio~of~beamer~(#1)~and~reveal~(#2:#3)~are~not~consistent.\\
  You~must~specify~consistent~values~for~width/height~and~aspectratio~
  otherwise~your~reveal~items~(videos/images/animations)~will~not~appear~
  on~the~right~locations~on~your~reveal-slidedeck.
}
\msg_new:nnn{ beamer-reveal } { missing-aspectratio } {
  missing~aspect~ratio.\\
  You~need~to~specify~at~least~an~aspect~ratio~for~a~beamer-reveal~item~
  you~want~to~put~on~the~reveal~slide.
}
\msg_new:nnn{ beamer-reveal } { missing-width-or-height } {
  missing~width~or~height.\\
  You~need~to~specify~at~least~a~width~or~a~height~for~a~beamer-reveal~item~
  you~want~to~put~on~the~reveal~slide.
}
\msg_new:nnn{ beamer-reveal } { overconstrained-box } {
  overconstrained~box.\\
  You~cannot~both~specify~the~width~and~the~height~of~a~beamer-reveal~item~
  you~want~to~put~on~the~slide.
  Specify~width~and~aspectratio~or~height~and~aspectratio.
}
\msg_new:nnn{ beamer-reveal } { dynamic-option-for-staticcontent } {
  dynamic~option~given~(autoplay,~controls,~loop,~muted)~for~static~
  content~(\image,~\iframe).\\
  These~options~make~no~sense~for~the~\image~command.~Remove~them.
}
\msg_new:nnn{ beamer-reveal } { animation-option-for-nonanimation } {
  duration~and~framerate~are~options~that~can~only~be~given~for~the~
  \animation~command.\\
  Remove~them~from~the~\video,~\audio,~\image~and~\iframe~commands.
}
\msg_new:nnn { beamerreveal / Syntax } { missing-coordinate }
    { you~specified~'\c_backslash_str at'~but~gave~no~coordinate. }
\msg_new:nnn { beamerreveal / Syntax } { old-at-syntax }
    { I~choked~on~'a';~note~that~the~syntax~has~changed:~replace~'at'~with~'\c_backslash_str at'. }
\msg_new:nnn { beamerreveal / Syntax } { missing-at }
    { you~specified~a~coordinate~but~not~an~'\c_backslash_str at'~token;~did~you~forget~that? }
⟨/reveal⟩
```

14.3 Some pdflatex backwards compatibility

```
⟨*reveal⟩
\@ifundefined{pagewidth}{%
  \let\pagewidth\pdfpagewidth
  \let\pageheight\pdfpageheight
```

```
}{}
⟨/reveal⟩
```

14.4 Package options

First some global variables to store the global width and height of the presentation, that can be specified as package options:

```
⟨*reveal⟩
\tl_new:N \g_@@_beameraspectratio_tl
\tl_set:Nn \g_@@_beameraspectratio_tl {43}
\fp_new:N \g_@@_canvaswidth_fp
\fp_set:Nn \g_@@_canvaswidth_fp { \dim_to_fp:n {\paperwidth} }
\fp_new:N \g_@@_canvasheight_fp
\fp_set:Nn \g_@@_canvasheight_fp { \dim_to_fp:n {\paperheight} }
\fp_new:N \g_@@_canvasaspectratio_fp
\fp_set:Nn \g_@@_canvasaspectratio_fp { \g_@@_canvaswidth_fp / \g_@@_canvasheight_fp }
\int_new:N \g_@@_canvaswidth_int
\int_set:Nn \g_@@_canvaswidth_int {0}
\int_new:N \g_@@_canvasheight_int
\int_set:Nn \g_@@_canvasheight_int {0}
\bool_new:N \g_@@_embed_bool
\bool_set:Nn \g_@@_embed_bool { \c_false_bool }
\bool_new:N \g_@@_loop_bool
\bool_set:Nn \g_@@_loop_bool { \c_false_bool }
\int_new:N \g_@@_autoslide_int
\int_set:Nn \g_@@_autoslide_int { -1 }
\keys_define:nn { beamerreveal } {
  width     .int_set:N        = \g_@@_canvaswidth_int,
  width     .value_required:n = true,
  height    .int_set:N        = \g_@@_canvasheight_int,
  height    .value_required:n = true,
  embed     .bool_set:N       = \g_@@_embed_bool,
  loop      .bool_set:N       = \g_@@_loop_bool,
  autoslide .int_set:N        = \g_@@_autoslide_int,
  autoslide .value_required:n = true,
}
\ProcessKeyOptions[beamerreveal]
\int_compare:nNnTF { \g_@@_canvaswidth_int } = {0}
    {
      \int_compare:nNnTF { \g_@@_canvasheight_int } = {0}
          {
            % we assume 4x3 on an HD screen
            \int_set:Nn \g_@@_canvaswidth_int { 1920 }
            \int_set:Nn \g_@@_canvasheight_int
                { \fp_eval:n { round( \g_@@_canvaswidth_int / \g_@@_canvasaspectratio_fp ) } }
          }
          {
            % we assume 4x3
            \int_set:Nn \g_@@_canvaswidth_int
                { \fp_eval:n { round( \g_@@_canvasheight_int * \g_@@_canvasaspectratio_fp ) } }
          }
    }
    {
      \int_compare:nNnTF { \g_@@_canvasheight_int } = {0}
          {
            % we assume 4x3 on an HD screen
            \int_set:Nn \g_@@_canvasheight_int{ \fp_eval:n
              { round( \g_@@_canvaswidth_int / \g_@@_canvasaspectratio_fp ) } }
          }
          {
            % both are set, we're good to go
```

```
            \fp_new:N \l_@@_canvasaspectratioresidue_fp
            \fp_set:Nn \l_@@_canvasaspectratioresidue_fp
               { \fp_abs:n { \g_@@_canvaswidth_int / \g_@@_canvasheight_int - \g_@@_canvasaspectratio_fp } }
            \fp_compare:nNnTF { \l_@@_canvasaspectratioresidue_fp } > {0.001}
               {
                 \msg_warning:nneee { beamer-reveal } { inconsistent-dimensions }
                     { \tl_use:N \g_@@_beameraspectratio_tl }
                     { \int_use:N \g_@@_canvaswidth_int }
                     { \int_use:N \g_@@_canvasheight_int }
               }{}
          }
    }
⟨/reveal⟩
```

14.5 Extra options for the frame environment of BEAMER

```
⟨*reveal⟩
\bool_new:N \g_@@_titlepage_bool
\define@key{beamerframe}{titleslide}[true]{%
  \ExplSyntaxOn
  \bool_gset_true:N \g_@@_titlepage_bool
  \ExplSyntaxOff
}
\bool_new:N \g_@@_sectionslide_bool
\define@key{beamerframe}{sectionslide}[true]{%
  \ExplSyntaxOn
  \bool_gset_true:N \g_@@_sectionslide_bool
  \ExplSyntaxOff
}
\bool_new:N \g_@@_subsectionslide_bool
\define@key{beamerframe}{subsectionslide}[true]{%
  \ExplSyntaxOn
  \bool_gset_true:N \g_@@_subsectionslide_bool
  \ExplSyntaxOff
}
\tl_new:N \g_@@_transition_tl
\define@key{beamerframe}{transition}[none]{%
  \ExplSyntaxOn
  \tl_gset:Nn \g_@@_transition_tl { #1 }
  \ExplSyntaxOff
}
\define@key{beamerframe}{embed}[true]{%
  \ExplSyntaxOn
  \bool_gset_true:N \g_@@_embed_bool
  \ExplSyntaxOff
}
\define@key{beamerframe}{autoslide}[0]{%
  \ExplSyntaxOn
  \int_set:Nn \g_@@_autoslide_int {#1}
  \ExplSyntaxOff
}
⟨/reveal⟩
```

14.6 File writing

File handles and auxiliary functions to write data to the `.rvl` file.

```
⟨*reveal⟩
\iow_new:N \g_@@_rvlfile
\iow_open:Nn \g_@@_rvlfile {\jobname.rvl}
⟨/reveal⟩
```

\writecomment_@@:n

```
⟨*reveal⟩
\cs_new:Npn \writecomment_@@:n #1
   { \iow_now:Ne \g_@@_rvlfile {\c_percent_str\c_percent_str\c_space_tl #1} }
⟨/reveal⟩
```

\writecontrol_@@:n

```
⟨*reveal⟩
\cs_new:Npn \writecontrol_@@:nn #1 #2
   { \iow_now:Ne \g_@@_rvlfile {@ @#1:~#2} }
```

\writeliteral_@@:n

```
\cs_new:Npn \writeliteral_@@:n #1
   { \iow_now:Nx \g_@@_rvlfile {#1} }
```

\writeraw_@@:n

```
\cs_new:Npn \writeraw_@@:n #1
   { \iow_now:Nn \g_@@_rvlfile { #1 } }
```

\writeraw_@@:n

```
\cs_generate_variant:Nn \iow_now:Nn {No }
\cs_new:Npn \writeraw_@@:o #1
   { \iow_now:No \g_@@_rvlfile { #1 } }
```

Now initialize our `.rvl` file.

```
\writecomment_@@:n {Beamer-reveal~driver~file~
  (\tl_use:N \g_@@_fileversion_tl - \tl_use:N \g_@@_filedate_tl)}
\writecontrol_@@:nn {Presentation} {}
\tl_new:N \l_@@_my_compiler_tl
\tl_set:Nn \l_@@_my_compiler_tl { unknown }
\sys_if_engine_pdftex:T { \tl_set:Nn \l_@@_my_compiler_tl { pdflatex } }
\sys_if_engine_xetex:T  { \tl_set:Nn \l_@@_my_compiler_tl { xelatex } }
\sys_if_engine_luatex:T { \tl_set:Nn \l_@@_my_compiler_tl { lualatex } }
\AtBeginDocument{
  \writeliteral_@@:n {-parameters:
    latexversion={\tl_use:N \g_@@_fileversion_tl},
    latexdate={\tl_use:N \g_@@_filedate_tl},
    sourcefilename={\currfilename},
    title={\inserttitle},
    author={\beamer@shortauthor},
    compiler={\tl_use:N \l_@@_my_compiler_tl },
    \bool_if:NT \g_@@_embed_bool {embed={},}
    \bool_if:NTF \g_@@_loop_bool {loop={true}}{loop={false}},
    \int_compare:nNnTF \g_@@_autoslide_int < {0} {}{autoslide={\int_use:N \g_@@_autoslide_int},}
    canvaswidth={\int_use:N \g_@@_canvaswidth_int},
    canvasheight={\int_use:N \g_@@_canvasheight_int}
  }
}
⟨/reveal⟩
```

14.7 Frame generation

```
⟨*reveal⟩
\AddToHook{ env / frame / begin} {
  \bool_gset_false:N \g_@@_titlepage_bool
  \bool_gset_false:N \g_@@_sectionslide_bool
  \bool_gset_false:N \g_@@_subsectionslide_bool
  \bool_gset_false:N \g_@@_embed_bool
  \int_gset:Nn \g_@@_autoslide_int {-1}
  \tl_gset:Nn \g_@@_transition_tl {none}
}
\AddToHook{ env / beamer@frameslide / before} {
  \writecontrol_@@:nn {BeamerFrame} {}
}
\AddToHook{ env / beamer@frameslide / after} {
  \writeliteral_@@:n {-parameters:
    rawpage={\insertpagenumber},
    truepage={\insertframenumber},
    overlay={\insertoverlaynumber},
    transition={\tl_use:N \g_@@_transition_tl},
    \bool_if:NT \g_@@_embed_bool {embed={},}
    \int_compare:nNnTF \g_@@_autoslide_int < {0} {} {autoslide={\int_use:N \g_@@_autoslide_int},}
    \bool_if:NT \g_@@_embed_bool {embed={},}
    \bool_if:NTF \g_@@_sectionslide_bool
         {
           title={\secname},toc={section}
         }
         {
           \bool_if:NTF \g_@@_subsectionslide_bool
                {
                  title={\subsecname},toc={subsection}
                }
                {
                  \bool_if:NTF \g_@@_titlepage_bool
                       {
                         title={\beamer@shorttitle},toc={titlepage}
                       }
                       {
                         \sys_if_engine_pdftex:TF % avoid pdflatex screwing up old accents
                             {title={\unexpanded\expandafter{\beamer@shortframetitle}}}
                             {title={\beamer@shortframetitle}}
                       }
                }
         }
  }
}
\AtBeginDocument {
  \AddToHook { cmd / note / before } {
    \writeliteral_@@:n
                 {-hasnote:true}
  }
}
⟨/reveal⟩
```

14.8 Common keys for the macros

```
⟨*reveal⟩
\fp_new:N \l_@@_mediawidth_fp
\fp_new:N \l_@@_mediaheight_fp
\fp_new:N \l_@@_mediaframerate_fp
\fp_new:N \l_@@_mediaduration_fp
\fp_new:N \l_@@_mediapdfprogress_fp
```

```
\tl_new:N \l_@@_mediafit_tl
\tl_new:N \l_@@_mediabackground_tl
\tl_new:N \l_@@_mediahandoutprogress_tl
\tl_new:N \l_@@_mediaprogress_tl
\fp_new:N \l_@@_xposdelta_fp
\fp_new:N \l_@@_yposdelta_fp
\bool_new:N \l_@@_mediaembed_bool
\bool_new:N \l_@@_mediaautoplay_bool
\bool_new:N \l_@@_medialoop_bool
\bool_new:N \l_@@_mediaboxdraw_bool
\bool_new:N \l_@@_mediamuted_bool
\bool_new:N \l_@@_mediacontrols_bool
\tl_new:N \l_@@_mediaanchor_tl
\bool_new:N \l_@@_mediaexpandonce_bool

\msg_new:nnn { beamerreveal / Syntax } { unknown-key }
{ Unknown~key~'#1'~for~media~(video,~animated,~...)~command. }
\msg_new:nnn { beamerreveal / Syntax } { illegal-keys-video }
{ Illegal~key(s)~'#1'~for~a~\video. }
\msg_new:nnn { beamerreveal / Syntax } { illegal-keys-image }
{ Illegal~key(s)~'#1'~for~an~\image. }
\msg_new:nnn { beamerreveal / Syntax } { illegal-keys-iframe }
{ Illegal~key(s)~'#1'~for~an~\iframe. }
\msg_new:nnn { beamerreveal / Syntax } { illegal-keys-animation }
{ Illegal~key(s)~'#1'~for~an~\animation. }
\msg_new:nnn { beamerreveal / Syntax } { key-syntax-changed }
{ The~syntax~of~the~key~of~'#1'~changed;~change~this~into~'#2'. }
\keys_define:nn { beamerreveal / media } {
  width .fp_set:N               = \l_@@_mediawidth_fp,
  width .value_required:n       = true,
  width .initial:n              = 0,
  width .groups:n               = { size },
  height .fp_set:N              = \l_@@_mediaheight_fp,
  height .value_required:n      = true,
  height .initial:n             = 0,
  height .groups:n              = { size },
  aspectratio .fp_set:N         = \l_@@_mediaaspectratio_fp,
  aspectratio .value_required:n = true,
  aspectratio .groups:n         = { size },
  fit .tl_set:N                 = \l_@@_mediafit_tl,
  fit .value_required:n         = true,
  fit .initial:n                = fill,
  fit .groups:n                 = { fit },
  background .tl_set:N          = \l_@@_mediabackground_tl,
  background .value_required:n  = true,
  background .initial:n         = white,
  background .groups:n          = { draw },
  draw .bool_set:N              = \l_@@_mediaboxdraw_bool,
  draw .initial:n               = false,
  draw .groups:n                = { draw },
  embed .bool_set:N             = \l_@@_mediaembed_bool,
  embed .initial:n              = false,
  autoplay .bool_set:N          = \l_@@_mediaautoplay_bool,
  autoplay .initial:n           = false,
  autoplay .groups:n            = { dynamic },
  loop .bool_set:N              = \l_@@_medialoop_bool,
  loop .initial:n               = false,
  loop .groups:n                = { dynamic },
  controls .bool_set:N          = \l_@@_mediacontrols_bool,
  controls .initial:n           = false,
  controls .groups:n            = { dynamic },
  muted .bool_set:N             = \l_@@_mediamuted_bool,
```

```
muted .initial:n              = false,
muted .groups:n               = { sound },
duration .fp_set:N             = \l_@@_mediaduration_fp,
duration .initial:n            = 0,
duration .value_required:n     = true,
duration .groups:n             = { animation },
pdfprogress .code:n            = { \msg_fatal:nnnn { beamerreveal / Syntax } { key-syntax-changed }
  { pdfprogress=#1 } { progress=#1 } },
pdfprogress .groups:n          = { animation },
pdf~progress .code:n           = { \msg_fatal:nnnn { beamerreveal / Syntax } { key-syntax-changed }
  { pdf~progress=#1 } { progress=#1 } },
pdf~progress .groups:n         = { animation },
handout~progress .code:n       = { \msg_fatal:nnnn { beamerreveal / Syntax } { key-syntax-changed }
  { handout~progress=#1 } { progress={0.1|handout:1/0.2,2/0.5,3/0.75} } },
handout~progress .groups:n     = { animation },
progress .tl_set:N             = \l_@@_mediaprogress_tl,
progress .value_required:n     = true,
progress .groups:n             = { animation },
framerate .fp_set:N            = \l_@@_mediaframerate_fp,
framerate .initial:n           = 0,
framerate .value_required:n    = true,
framerate .groups:n            = { animation },
expand~once .bool_set:N         = \l_@@_mediaexpandonce_bool,
expand~once .initial:n          = false,
expand~once .groups:n           = { animation },
anchor .choice:,
anchor / center .code:n       = { \fp_set:Nn \l_@@_xposdelta_fp { -0.5 }
                                  \fp_set:Nn \l_@@_yposdelta_fp { -0.5 }
                                  \tl_set:Nn \l_@@_mediaanchor_tl {#1} },
anchor / west .code:n         = { \fp_set:Nn \l_@@_xposdelta_fp { 0 }
                                  \fp_set:Nn \l_@@_yposdelta_fp { -0.5 }
                                  \tl_set:Nn \l_@@_mediaanchor_tl {#1} },
anchor / north~west .code:n   = { \fp_set:Nn \l_@@_xposdelta_fp { 0 }
                                  \fp_set:Nn \l_@@_yposdelta_fp { 0 }
                                  \tl_set:Nn \l_@@_mediaanchor_tl {#1} },
anchor / north .code:n        = { \fp_set:Nn \l_@@_xposdelta_fp { -0.5 }
                                  \fp_set:Nn \l_@@_yposdelta_fp { 0 }
                                  \tl_set:Nn \l_@@_mediaanchor_tl {#1} },
anchor / north~east .code:n   = { \fp_set:Nn \l_@@_xposdelta_fp { -1 }
                                  \fp_set:Nn \l_@@_yposdelta_fp { 0 }
                                  \tl_set:Nn \l_@@_mediaanchor_tl {#1} },
anchor / east .code:n         = { \fp_set:Nn \l_@@_xposdelta_fp { -1 }
                                  \fp_set:Nn \l_@@_yposdelta_fp { -0.5 }
                                  \tl_set:Nn \l_@@_mediaanchor_tl {#1} },
anchor / south~east .code:n   = { \fp_set:Nn \l_@@_xposdelta_fp { -1 }
                                  \fp_set:Nn \l_@@_yposdelta_fp { -1 }
                                  \tl_set:Nn \l_@@_mediaanchor_tl {#1} },
anchor / south .code:n        = { \fp_set:Nn \l_@@_xposdelta_fp { -0.5 }
                                  \fp_set:Nn \l_@@_yposdelta_fp { -1 }
                                  \tl_set:Nn \l_@@_mediaanchor_tl {#1} },
anchor / south~west .code:n   = { \fp_set:Nn \l_@@_xposdelta_fp { 0 }
                                  \fp_set:Nn \l_@@_yposdelta_fp { -1 }
                                  \tl_set:Nn \l_@@_mediaanchor_tl {#1} },
anchor .value_required:n      = true,
anchor .initial:n             = center,
anchor .groups:n              = { position },
above .meta:n                 = { anchor = south },
above .groups:n               = { position },
below .code:n                 = { anchor = north },
below .groups:n               = { position },
left .code:n                  = { anchor = east },
left .groups:n                = { position },
```

```
  right .code:n                = { anchor = west },
  right .groups:n              = { position },
  above~left .code:n           = { anchor = south east },
  above~left .groups:n         = { position },
  above~right .code:n          = { anchor = south west },
  above~right .groups:n        = { position },
  below~left .code:n           = { anchor = north east },
  below~left .groups:n         = { position },
  below~right .code:n          = { anchor = north west },
  below~right .groups:n        = { position },
  unknown .code:n =
  {
    \msg_fatal:nne { beamerreveal / Syntax } { unknown-key } {\l_keys_key_str}
  },
}
⟨/reveal⟩
```

14.9 Fiddling with relative widths/heights

As mentioned in the section 'quirks' the package only allows specifying box dimensions as height and an apsect ratio, or width and an aspectratio. BEAMER-REVEAL will recalculate these into fractional values relative to the slide width and height.
In order not to get lost in the calculations, let's find the equations we need to implement. Let's call the absolute width and height W and L and the relative width and height w and h. Let W_P and H_P be the absolute page width and height. Obviously:

$$w = \frac{W}{W_P} \qquad\qquad h = \frac{H}{H_P} \tag{2}$$

Now let's denote the aspectratio of an object as A and the aspectratio of the page as A_P.

$$A = \frac{W}{H} \qquad\qquad A_P = \frac{W_P}{H_P} \tag{3}$$

This allows to calculate w from h and vice versa:

$$w = \frac{W}{W_P} = \frac{AH}{A_P H_P} = \frac{A}{A_P} h \qquad\qquad h = \frac{H}{H_P} = \frac{W/A}{W_P/A_P} = \frac{A_P}{A} w \tag{4}$$

The following macro calculates verifies if at least the aspectratio A has been given otherwise a fatal error is generated. It then calculates w from h or h from w unless both w and h are missing, or unless both have been specified.

`\process_a_w_h_@@:NNN`

```
⟨*reveal⟩
\cs_new:Npn \process_a_w_h_@@:NNN #1#2#3 {
  \fp_compare:nNnTF { #1 } = {0} {
    \msg_fatal:nn { beamer-reveal } { missing-aspectratio }
  } {
    \fp_compare:nNnTF { #2 } = {0} {
      \fp_compare:nNnTF { #3 } = {0} {
        \msg_fatal:nn { beamer-reveal } { missing-width-or-height }
      } {
        % calculating w
        \fp_gset:Nn \l_tmpa_fp { round( #1 / \g_@@_canvasaspectratio_fp * #3, 6 ) }
        \fp_gset:Nn \l_tmpb_fp { #3 }
      }
    } {
      \fp_compare:nNnTF { #3 } = {0} {
        % calculating h
        \fp_gset:Nn \l_tmpa_fp { #2 }
```

```
        \fp_gset:Nn \l_tmpb_fp { round( \g_@@_canvasaspectratio_fp / #1 * #2, 6 ) }
      } {
        \msg_fatal:nn { beamer-reveal } { overconstrained-box }
      }
    }
  }
  \fp_set_eq:NN #2 \l_tmpa_fp
  \fp_set_eq:NN #3 \l_tmpb_fp
}
⟨/reveal⟩
```

14.10 Auxiliary functions

\@@extractloleft

```
⟨*reveal⟩
\newdimen\xloleft
\newdimen\yloleft
\newcommand*\@@extractloleft[1]{\path  (#1);\pgfgetlastxy{\xloleft}{\yloleft};}
⟨/reveal⟩
```

\@@extractupright

```
⟨*reveal⟩
\newdimen\xupright
\newdimen\yupright
\newcommand*\@@extractupright[1]{\path (#1);\pgfgetlastxy{\xupright}{\yupright};}
⟨/reveal⟩
```

Finally, some native l3exp fp variables to hold x and y position.

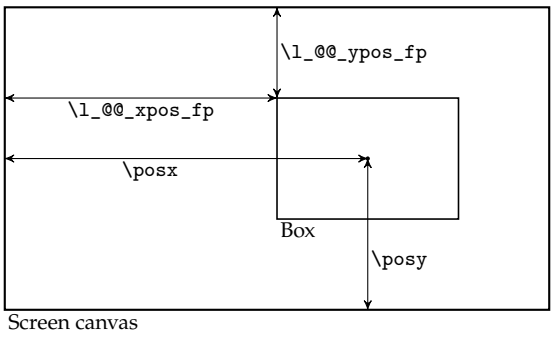
```
⟨*reveal⟩
\fp_new:N \l_@@_xpos_fp
\fp_new:N \l_@@_ypos_fp
⟨/reveal⟩
```

14.11 Macros

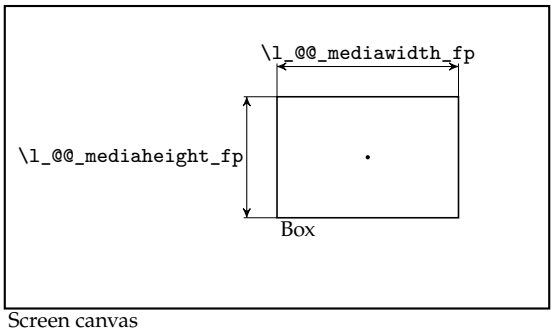
14.11.1 Main macros

The macro's fiddle with positions and sizes of boxes. Therefore it helps to have the following pictures in mind when reading the code.

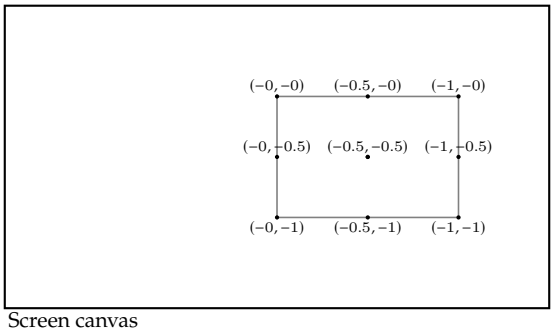
The parameters \posx and \posy are the relative BEAMER coördinates of the box. The parameters \l_@@_xpos_fp and \l_@@_ypos_fp are the HTML top and left distances of the top left corner of the box. All parameters are relative to the screen canvas dimensions (and therefore have values in between 0 and 1).



The width and height of the box have dedicated parameters.



The anchor of a box can be any of the main 8 wind directions (north, east, south, west, and the ones in between those). This anchor location in combination with the width and height of the box, allow for a correction on the HTML coordinates. The correction coefficients (relative to width and height) are:



The flow of the macros is as follows. We use the `\video` macro as example:

```
- |\video|:
  - knows it is a |Dynamic| element (= XXX below)
  - parses the appropriate keys from the option parameter list
  - raises a fatal error if unknown keys have been detected
  - hands over to the
    |\contentDispatcher|:
    - checks if this is overlay content (|\at (x,y)|); hands over to the
      |\overlayXXXContent|: generates overlay content
    - checks if this is insert content (no |\at (x,y)|); hands over to the
      |\insertXXXContent|: generates insert content
    - otherwise: raises correct fatal syntax errors
```

\video

438 ⟨*reveal⟩

```
\tl_new:N \l_@@_keyoverflow_tl
\NewDocumentCommand\video{D<>{1-} O{} d() t\at d() m }{
  \only<#1> {
    % begin group to keep the key setting local
    \group_begin:
    \keys_set_exclude_groups:nnnN { beamerreveal / media } { animation } { #2 } \l_@@_keyoverflow_tl
    \tl_if_empty:NTF \l_@@_keyoverflow_tl {} {
      \msg_fatal:nne { beamerreveal / Syntax } { illegal-keys-video } { \l_@@_keyoverflow_tl }
    }
    \contentDispatcher_@@:nnnnnNN {#3} {#4} {#5} {#6} {video}
                                  \overlayDynamicContent_@@:nnnn
                                  \insertDynamicContent_@@:nnn
    \group_end:
  }
}
⟨/reveal⟩
```

\audio

```
⟨*reveal⟩
\NewDocumentCommand\audio{ D<>{1-} O{} d() t\at d() m }{
  \only<#1> {
    % begin group to keep the key setting local
    \group_begin:
    \keys_set_exclude_groups:nnnN { beamerreveal / media } { animation } { #2 } \l_@@_keyoverflow_tl
    \tl_if_empty:NTF \l_@@_keyoverflow_tl {} {
      \msg_fatal:nne { beamerreveal / Syntax } { illegal-keys-audio } { \l_@@_keyoverflow_tl }
    }
    \contentDispatcher_@@:nnnnnNN {#3} {#4} {#5} {#6} {audio}
                                  \overlayDynamicContent_@@:nnnn
                                  \insertDynamicContent_@@:nnn
    \group_end:
  }
}
⟨/reveal⟩
```

\iframe

```
⟨*reveal⟩
\NewDocumentCommand\iframe{D<>{1-} O{} d() t\at d() m }{
  \only<#1> {
    % begin group to keep the key setting local
    \group_begin:
    \keys_set_exclude_groups:nnnN { beamerreveal / media } { dynamic, sound, animation } { #2 } \l_@@_keyoverflow
    \tl_if_empty:NTF \l_@@_keyoverflow_tl {} {
      \msg_fatal:nne { beamerreveal / Syntax } { illegal-keys-iframe } { \l_@@_keyoverflow_tl }
    }
    \contentDispatcher_@@:nnnnnNN {#3} {#4} {#5} {#6} {iframe}
                                  \overlayDynamicContent_@@:nnnn
                                  \insertDynamicContent_@@:nnn
    \group_end:
  }
}
⟨/reveal⟩
```

\image

```
⟨*reveal⟩
\NewDocumentCommand\image{D<>{1-} O{} d() t\at d() m}{
  \only<#1> {
    % begin group to keep the key setting local
    \group_begin:
    \keys_set_exclude_groups:nnnN { beamerreveal / media } { dynamic, sound, animation } { #2 } \l_@@_keyoverflow
    \tl_if_empty:NTF \l_@@_keyoverflow_tl {} {
      \msg_fatal:nne { beamerreveal / Syntax } { illegal-keys-image } { \l_@@_keyoverflow_tl }
    }
    \contentDispatcher_@@:nnnnnNN {#3} {#4} {#5} {#6} {image}
                                  \overlayStaticContent_@@:nnnn
                                  \insertStaticContent_@@:nnn
    \group_end:
  }
}
⟨/reveal⟩
```

\animation

```
⟨*reveal⟩
\NewDocumentCommand\animation{D<>{1-} O{} d() t\at d() +m}{
  \only<#1> {
    % begin group to keep the key setting local
    \group_begin:
    \keys_set_exclude_groups:nnnN { beamerreveal / media } { size, sound, fit } { #2 } \l_@@_keyoverflow_tl
    \tl_if_empty:NTF \l_@@_keyoverflow_tl {} {
      \msg_fatal:nne { beamerreveal / Syntax } { illegal-keys-animation } { \l_@@_keyoverflow_tl }
    }
    \contentDispatcher_@@:nnnnnNN {#3} {#4} {#5} {#6} {animation}
                                  \overlayFixedContent_@@:nnnn
                                  \insertFixedContent_@@:nnn
    \group_end:
  }
}
⟨/reveal⟩
```

\still

```
⟨*reveal⟩
\NewDocumentCommand\still{D<>{1-} O{} d() t\at d() +m}{
  \only<#1> {
    % begin group to keep the key setting local
    \group_begin:
    \keys_set_exclude_groups:nnnN { beamerreveal / media } { size, sound, fit, dynamic } { #2 } \l_@@_keyoverflow
    \tl_if_empty:NTF \l_@@_keyoverflow_tl {} {
      \msg_fatal:nne { beamerreveal / Syntax } { illegal-keys-animation } { \l_@@_keyoverflow_tl }
    }
    \contentDispatcher_@@:nnnnnNN {#3} {#4} {#5} {#6} {still}
                                  \overlayFixedContent_@@:nnnn
                                  \insertFixedContent_@@:nnn
    \group_end:
  }
}
⟨/reveal⟩
```

14.11.2 Auxiliary macros

\at

This is actually no macro, it even won't clash with a macro called \at that the user may declare. It is just a parsing token as documented in usrguide.pdf under 'control sequence tokens'.

\contentDispatcher_@@:nnnnn

```
⟨*reveal⟩
\cs_new:Npn \contentDispatcher_@@:nnnnnNN #1 #2 #3 #4 #5 #6 #7 {
  \IfBooleanTF{#2}
              {% \at
                \IfNoValueTF{#3}
                            { \msg_fatal:nn { beamerreveal / Syntax } { missing-coordinate } }
                            { #6 {#1} {#3} {#4} {#5} }
              }
              {
                %no \at
                \IfNoValueTF{#3}
                            {
                              \tl_if_eq:nnTF {#4} a
                                 { \msg_fatal:nn { beamerreveal / Syntax } { old-at-syntax } }
                                 { #7 {#1} {#4} {#5} }
                            }
                            { \msg_fatal:nn { beamerreveal / Syntax } { missing-at } }
              }
}
⟨/reveal⟩
```

\overlayDynamicContent_@@:nnnn

```
⟨*reveal⟩
\cs_new:Npn \overlayDynamicContent_@@:nnnn #1#2#3#4 {
  % convert combination width/aspectratio to height, or
  %                     height/aspectratio to width
  \process_a_w_h_@@:NNN \l_@@_mediaaspectratio_fp \l_@@_mediawidth_fp \l_@@_mediaheight_fp

    % extract relative position from bottom left of page (0,0) to top right (1,1)
  \seq_set_split:Nnn \l_tmpa_seq { , } { #2 }
  \pgfmathsetmacro{\posx}{\seq_item:Nn \l_tmpa_seq {1}}
  \pgfmathsetmacro{\posy}{\seq_item:Nn \l_tmpa_seq {2}}
  % convert to html coordinates from top left corner, and correct for anchor location
  \fp_set:Nn \l_@@_xpos_fp { \posx + \l_@@_xposdelta_fp * \l_@@_mediawidth_fp }
  \fp_set:Nn \l_@@_ypos_fp { (1-\posy) + \l_@@_yposdelta_fp * \l_@@_mediaheight_fp }

  % write info to .rvl file
  \writeliteral_@@:n {
    -#4:
      width={\fp_use:N \l_@@_mediawidth_fp},
      height={\fp_use:N \l_@@_mediaheight_fp},
      fit={\tl_use:N \l_@@_mediafit_tl},
      background={\tl_use:N \l_@@_mediabackground_tl},
      \bool_if:NTF \l_@@_mediaautoplay_bool {autoplay={},} {}
      \bool_if:NTF \l_@@_mediacontrols_bool {controls={},} {}
      \bool_if:NTF \l_@@_medialoop_bool {loop={},} {}
      \bool_if:NTF \l_@@_mediamuted_bool {muted={},} {}
      \bool_if:NTF \l_@@_mediaembed_bool {embed={},} {}
```

```
      x={\fp_use:N \l_@@_xpos_fp},
      y={\fp_use:N \l_@@_ypos_fp},
      file={#3}
  }

  % write node to PDF file
  \bool_if:NTF \l_@@_mediaboxdraw_bool {
    \begin{tikzpicture}[overlay,remember~picture,font=\tiny]
      \@@extractloleft{$(current~page.south~west)$}
      \@@extractupright{$(current~page.north~east)$}
      \node[
        anchor = \tl_use:N \l_@@_mediaanchor_tl,
        minimum~width={\fp_use:N \l_@@_mediawidth_fp * (\xupright - \xloleft)},
        minimum~height={\fp_use:N \l_@@_mediaheight_fp * (\yupright - \yloleft)},
        draw,
        fill=\tl_use:N \l_@@_mediabackground_tl,
      ] (#1) at ({\xloleft*(1-\posx)+\xupright*\posx},{\yloleft*(1-\posy)+\yupright*\posy})
           {\textcolor{gray}{#3}};
    \end{tikzpicture}
  }{}
}
⟨/reveal⟩
```

\insertDynamicContent_@@:nnn

```
⟨*reveal⟩
\cs_new:Npn \insertDynamicContent_@@:nnn #1#2#3 {
  % parameters: nodename filename content-type
  % convert combination width/aspectratio to height, or
  %                      height/aspectratio to width
  \process_a_w_h_@@:NNN \l_@@_mediaaspectratio_fp \l_@@_mediawidth_fp \l_@@_mediaheight_fp

  \fp_set:Nn \l_tmpa_fp { \dim_eval:n { \pagewidth } }
  \fp_set:Nn \l_tmpb_fp { \dim_eval:n { \pageheight } }

  % write node to PDF file
  \begin{tikzpicture}[remember~picture,font=\tiny]
    \node[
      outer~sep=0pt,inner~sep=0pt,rectangle,
      anchor = \tl_use:N \l_@@_mediaanchor_tl,
      minimum~width={\fp_use:N \l_@@_mediawidth_fp * \fp_use:N \l_tmpa_fp },
      minimum~height={\fp_use:N \l_@@_mediaheight_fp * \fp_use:N \l_tmpb_fp },
      \bool_if:NTF \l_@@_mediaboxdraw_bool {draw}{},
      fill=\tl_use:N \l_@@_mediabackground_tl,
    ] (#1) {\bool_if:NTF \l_@@_mediaboxdraw_bool {\textcolor{gray}{#2}}{}};
    \path let \p1=($(#1.north~west) - (current~page.north~west)$)
    in \pgfextra{
      \fp_set:Nn \l_@@_xpos_fp { \x1 / \fp_use:N \l_tmpa_fp }
      \fp_set:Nn \l_@@_ypos_fp { - \y1 / \fp_use:N \l_tmpb_fp }
      % write info to .rvl file
      \writeliteral_@@:n {
        -#3:
        width={\fp_use:N \l_@@_mediawidth_fp},
        height={\fp_use:N \l_@@_mediaheight_fp},
        fit={\tl_use:N \l_@@_mediafit_tl},
        background={\tl_use:N \l_@@_mediabackground_tl},
        \bool_if:NTF \l_@@_mediaautoplay_bool {autoplay={},} {}
        \bool_if:NTF \l_@@_mediacontrols_bool {controls={},} {}
        \bool_if:NTF \l_@@_medialoop_bool {loop={},} {}
        \bool_if:NTF \l_@@_mediamuted_bool {muted={},} {}
```

```
        x={\fp_use:N \l_@@_xpos_fp},
        y={\fp_use:N \l_@@_ypos_fp},
        file={#2}
      }
    };
  \end{tikzpicture}
}
⟨/reveal⟩
```

\overlayStaticContent_@@:nnnn

```
⟨*reveal⟩
\cs_new:Npn \overlayStaticContent_@@:nnnn #1#2#3#4 {
  % convert combination width/aspectratio to height, or
  %                      height/aspectratio to width
  \process_a_w_h_@@:NNN \l_@@_mediaaspectratio_fp \l_@@_mediawidth_fp \l_@@_mediaheight_fp

  % extract relative position from bottom left of page (0,0) to top right (1,1)
  \seq_set_split:Nnn \l_tmpa_seq { , } { #2 }
  \pgfmathsetmacro{\posx}{\seq_item:Nn \l_tmpa_seq {1}}
  \pgfmathsetmacro{\posy}{\seq_item:Nn \l_tmpa_seq {2}}
  % convert to html coordinates from top left corner, and correct for anchor location
  \fp_set:Nn \l_@@_xpos_fp { \posx + \l_@@_xposdelta_fp * \l_@@_mediawidth_fp }
  \fp_set:Nn \l_@@_ypos_fp { (1-\posy) + \l_@@_yposdelta_fp * \l_@@_mediaheight_fp }

  % write info to .rvl file
  \writeliteral_@@:n {
    -#4:
    width={\fp_use:N \l_@@_mediawidth_fp},
    height={\fp_use:N \l_@@_mediaheight_fp},
    fit={\tl_use:N \l_@@_mediafit_tl},
    background={\tl_use:N \l_@@_mediabackground_tl},
    \bool_if:NTF \l_@@_mediaautoplay_bool {autoplay={true},} {}
    \bool_if:NTF \l_@@_mediaembed_bool {embed={},} {}
    x={\fp_use:N \l_@@_xpos_fp},
    y={\fp_use:N \l_@@_ypos_fp},
    file={#3}
  }

  % write node to PDF file
  \bool_if:NTF \l_@@_mediaboxdraw_bool {
    \begin{tikzpicture}[overlay,remember~picture,font=\tiny]
      \@@extractloleft{$(current~page.south~west)$}
      \@@extractupright{$(current~page.north~east)$}
      \node[
        anchor = \tl_use:N \l_@@_mediaanchor_tl,
        minimum~width={\fp_use:N \l_@@_mediawidth_fp * (\xupright - \xloleft)},
        minimum~height={\fp_use:N \l_@@_mediaheight_fp * (\yupright - \yloleft)},
        draw,
        fill=\tl_use:N \l_@@_mediabackground_tl,
      ] (#1) at ({\xloleft*(1-\posx)+\xupright*\posx},{\yloleft*(1-\posy)+\yupright*\posy})
           {\textcolor{gray}{#3}};
    \end{tikzpicture}
  }{}
}
⟨/reveal⟩
```

\insertStaticContent_@@:nnn

```
⟨*reveal⟩
\cs_new:Npn \insertStaticContent_@@:nnn #1#2#3 {
  % convert combination width/aspectratio to height, or
  %                      height/aspectratio to width
  \process_a_w_h_@@:NNN \l_@@_mediaaspectratio_fp \l_@@_mediawidth_fp \l_@@_mediaheight_fp

  \fp_set:Nn \l_tmpa_fp { \dim_eval:n { \pagewidth } }
  \fp_set:Nn \l_tmpb_fp { \dim_eval:n { \pageheight } }

  % write node to PDF file
  \begin{tikzpicture}[remember~picture,font=\tiny]%
    \node[
      outer~sep=0pt,inner~sep=0pt,rectangle,
      anchor = \tl_use:N \l_@@_mediaanchor_tl,
      minimum~width={\fp_use:N \l_@@_mediawidth_fp * \fp_use:N \l_tmpa_fp },
      minimum~height={\fp_use:N \l_@@_mediaheight_fp * \fp_use:N \l_tmpb_fp },
      \bool_if:NTF \l_@@_mediaboxdraw_bool {draw}{},
      \bool_if:NTF \l_@@_mediaembed_bool {embed={},} {}
      fill=\tl_use:N \l_@@_mediabackground_tl,
    ] (#1) {\bool_if:NTF \l_@@_mediaboxdraw_bool {\textcolor{gray}{#2}}{}};
    \path let \p1=($(#1.north~west) - (current~page.north~west)$)
    in \pgfextra{
      \fp_set:Nn \l_@@_xpos_fp { \x1 / \fp_use:N \l_tmpa_fp }
      \fp_set:Nn \l_@@_ypos_fp { - \y1 / \fp_use:N \l_tmpb_fp }
      % write info to .rvl file
      \writeliteral_@@:n {
        -#3:
        width={\fp_use:N \l_@@_mediawidth_fp},
        height={\fp_use:N \l_@@_mediaheight_fp},
        fit={\tl_use:N \l_@@_mediafit_tl},
        background={\tl_use:N \l_@@_mediabackground_tl},
        \bool_if:NTF \l_@@_mediaautoplay_bool {autoplay={true},} {}
        x={\fp_use:N \l_@@_xpos_fp},
        y={\fp_use:N \l_@@_ypos_fp},
        file={#2}
      }
    };
  \end{tikzpicture}
}
⟨/reveal⟩
```

\overlayFixedContent_@@:nnnn

```
⟨*reveal⟩
\cs_new:Npn \overlayFixedContent_@@:nnnn #1#2#3#4 {
  % set up the progress macro
  \regex_split:nVN {\|handout:} { \l_@@_mediaprogress_tl } \l_tmpa_seq
  \fp_set:Nn \l_@@_mediapdfprogress_fp { \seq_item:Nn \l_tmpa_seq {1} }
  \tl_set:Nx \l_@@_mediahandoutprogress_tl { \seq_item:Nn \l_tmpa_seq {2} }
  \str_if_eq:eeTF { \use:c { beamer@currentmode } } { handout }
     {
       \tl_if_empty:NTF \l_@@_mediahandoutprogress_tl {
         \pgfmathsetmacro \progress { \fp_use:N \l_@@_mediapdfprogress_fp }
       }{
         \regex_split:nVN { , } { \l_@@_mediahandoutprogress_tl } \l_tmpa_seq
         \seq_map_inline:Nn  \l_tmpa_seq {
           \regex_split:nnN { / } { ##1 } \l_tmpb_seq
```

```
          \int_compare:nT { \use:c {beamer@overlaynumber} == \seq_item:Nn \l_tmpb_seq {1} }
              {
                \pgfmathsetmacro\progress{ \seq_item:Nn \l_tmpb_seq {2} }
              }
        }
      }
    }
    { \pgfmathsetmacro \progress { \fp_use:N \l_@@_mediapdfprogress_fp } }

% now set the content in a box and measure it
\hbox_set:Nn \l_tmpa_box {\tl_trim_spaces:n{#2}}
\pgfmathsetmacro\progress{ \fp_use:N \l_@@_mediapdfprogress_fp }%
\hbox_set:Nn \l_tmpa_box {\tl_trim_spaces:n{#3}}
\fp_set:Nn \l_tmpa_fp { \dim_eval:n { \box_wd:N \l_tmpa_box } }
\fp_set:Nn \l_tmpb_fp { \dim_eval:n { \pagewidth } }
\fp_set:Nn \l_@@_mediawidth_fp { \fp_eval:n { \l_tmpa_fp / \l_tmpb_fp } }
\fp_set:Nn \l_tmpa_fp { \dim_eval:n { \box_ht:N \l_tmpa_box } }
\fp_set:Nn \l_tmpb_fp { \dim_eval:n { \pageheight } }
\fp_set:Nn \l_@@_mediaheight_fp { \fp_eval:n { \l_tmpa_fp / \l_tmpb_fp } }

% extract relative position from bottom left of page (0,0) to top right (1,1)
\seq_set_split:Nnn \l_tmpa_seq { , } { #2 }
\pgfmathsetmacro{\posx}{\seq_item:Nn \l_tmpa_seq {1}}
\pgfmathsetmacro{\posy}{\seq_item:Nn \l_tmpa_seq {2}}

% convert to html coordinates from top left corner, and correct for anchor location
\fp_set:Nn \l_@@_xpos_fp { \posx + \l_@@_xposdelta_fp * \l_@@_mediawidth_fp }
\fp_set:Nn \l_@@_ypos_fp { (1-\posy) + \l_@@_yposdelta_fp * \l_@@_mediaheight_fp }

% write .rvl information
\writeliteral_@@:n {
  -#4:
  width={ \fp_use:N \l_@@_mediawidth_fp },
  height={ \fp_use:N \l_@@_mediaheight_fp },
  framerate={ \fp_use:N \l_@@_mediaframerate_fp },
  duration={ \fp_use:N \l_@@_mediaduration_fp },
  progress={ \fp_use:N \l_@@_mediapdfprogress_fp },
  \bool_if:NTF \l_@@_mediaembed_bool {embed={},} {}
  x={\fp_use:N \l_@@_xpos_fp},
  y={\fp_use:N \l_@@_ypos_fp},
  fit={fit},
  background={\tl_use:N \l_@@_mediabackground_tl},
  \bool_if:NTF \l_@@_mediaautoplay_bool {autoplay={},} {}
  \bool_if:NTF \l_@@_medialoop_bool {loop={},} {}
  \bool_if:NTF \l_@@_mediacontrols_bool {controls={},} {}
}

% put node in PDF files
\begin{tikzpicture}[overlay,remember~picture,font=\tiny]%
  \@@extractloleft{$(current~page.south~west)$}
  \@@extractupright{$(current~page.north~east)$}
  \node[
    outer~sep=0pt,inner~sep=0pt,rectangle,
    \bool_if:NTF \l_@@_mediaboxdraw_bool {draw}{},
  ]
  (#1) at ({\xloleft*(1-\posx)+\xupright*\posx},{\yloleft*(1-\posy)+\yupright*\posy})
        {\box_use:N \l_tmpa_box};
\end{tikzpicture}%


%
% write code to .rvl file
```

```
  \bool_if:NTF \l_@@_mediaexpandonce_bool
       { \writeraw_@@:o { #3 } } { \writeraw_@@:n { #3 } }
}
⟨/reveal⟩
```

\insertFixedContent_@@:nnn

```
⟨*reveal⟩
\cs_new:Npn \insertFixedContent_@@:nnn #1#2#3 {
  %% % first create the box with the content and measure it
  % check if we are in handout mode
  \regex_split:nVN {\|handout:} { \l_@@_mediaprogress_tl } \l_tmpa_seq
  \fp_set:Nn \l_@@_mediapdfprogress_fp { \seq_item:Nn \l_tmpa_seq {1} }
  \tl_set:Nx \l_@@_mediahandoutprogress_tl { \seq_item:Nn \l_tmpa_seq {2} }
  \str_if_eq:eeTF { \use:c { beamer@currentmode } } { handout }
      {
        \tl_if_empty:NTF \l_@@_mediahandoutprogress_tl {
          \pgfmathsetmacro \progress { \fp_use:N \l_@@_mediapdfprogress_fp }
        }{
          \regex_split:nVN { , } { \l_@@_mediahandoutprogress_tl } \l_tmpa_seq
          \seq_map_inline:Nn  \l_tmpa_seq {
            \regex_split:nnN { / } { ##1 } \l_tmpb_seq
            \int_compare:nT { \use:c {beamer@overlaynumber} == \seq_item:Nn \l_tmpb_seq {1} }
                {
                  \pgfmathsetmacro\progress{ \seq_item:Nn \l_tmpb_seq {2} }
                }
          }
        }
      }
      { \pgfmathsetmacro \progress { \fp_use:N \l_@@_mediapdfprogress_fp } }

  % now set the content in a box and measure it
  \hbox_set:Nn \l_tmpa_box {\tl_trim_spaces:n{#2}}
  \fp_set:Nn \l_tmpa_fp { \dim_eval:n { \box_wd:N \l_tmpa_box } }
  \fp_set:Nn \l_tmpb_fp { \dim_eval:n { \pagewidth } }
  \fp_set:Nn \l_@@_mediawidth_fp { \fp_eval:n { \l_tmpa_fp / \l_tmpb_fp } }
  \fp_set:Nn \l_tmpa_fp { \dim_eval:n { \box_ht:N \l_tmpa_box } }
  \fp_set:Nn \l_tmpb_fp { \dim_eval:n { \pageheight } }
  \fp_set:Nn \l_@@_mediaheight_fp { \fp_eval:n { \l_tmpa_fp / \l_tmpb_fp } }
  % restore the pagewidth in tmpa
  \fp_set:Nn \l_tmpa_fp { \dim_eval:n { \pagewidth } }

  % write node to PDF file
  \begin{tikzpicture}[remember~picture,font=\tiny]
    \node[
      outer~sep=0pt,inner~sep=0pt,rectangle,
      \bool_if:NTF \l_@@_mediaboxdraw_bool {draw}{},
    ] (#1) {\box_use:N \l_tmpa_box};
    \path let \p1=($(#1.north~west) - (current~page.north~west)$)
    in \pgfextra{
      \fp_set:Nn \l_@@_xpos_fp { \x1 / \fp_use:N \l_tmpa_fp }
      \fp_set:Nn \l_@@_ypos_fp { - \y1 / \fp_use:N \l_tmpb_fp }
      % write info to .rvl file
      \writeliteral_@@:n {
        -#3:
        width={ \fp_use:N \l_@@_mediawidth_fp },
        height={ \fp_use:N \l_@@_mediaheight_fp },
        framerate={ \fp_use:N \l_@@_mediaframerate_fp },
        duration={ \fp_use:N \l_@@_mediaduration_fp },
        progress={ \fp_use:N \l_@@_mediapdfprogress_fp },
```

```
        \bool_if:NTF \l_@@_mediaembed_bool {embed={},} {}
        x={\fp_use:N \l_@@_xpos_fp},
        y={\fp_use:N \l_@@_ypos_fp},
        fit={fit},
        background={\tl_use:N \l_@@_mediabackground_tl},
        \bool_if:NTF \l_@@_mediaautoplay_bool {autoplay={},} {}
        \bool_if:NTF \l_@@_medialoop_bool {loop={},} {}
        \bool_if:NTF \l_@@_mediacontrols_bool {controls={},} {}
      }
    };
  \end{tikzpicture}
  \bool_if:NTF \l_@@_mediaexpandonce_bool
       { \writeraw_@@:o { #2 } } { \writeraw_@@:n { #2 } }
}
```
⟨/reveal⟩

14.12 Postamble

⟨*reveal⟩
```
\AtEndDocument{
  \ExplSyntaxOn
  \iow_close:N \g_@@_rvlfile
  \ExplSyntaxOff
}
\ExplSyntaxOff
```
⟨/reveal⟩

References

[1] Till Tantau et al., "beamer - A LaTeX class for producing presentations and slides", https://ctan.org/pkg/beamer, online, accessed in December 2025.

[2] Alexander Grahn, "movie15 - Multimedia inclusion package", https://ctan.org/pkg/movie15, online, accessed in December 2025.

[3] Alexander Grahn, "media9 - Multimedia inclusion package with Adobe Reader-9/X compatibility", https://ctan.org/pkg/media9, online, accessed in December 2025.

[4] Hakim El Hattab et al., "reveal.js - The HTML Presentation framework", https://revealjs.com, online, accessed in January 2026.

[5] Posit Software - PBC, "Quarto - An open-source scientific and technical publishing system", https://quarto.org, online, accessed in January 2026.

[6] Grant Sanderson, "manim - A community maintained Python library for creating mathematical animations", https://www.manim.community, online, accessed in December 2025.

[7] Heiko Oberdiek, "pdfcrop - Crop PDF graphics", https://ctan.org/pkg/pdfcrop, online, accessed in December 2025.

[8] Poppler developers, "Poppler Utilities (pdftoppm)", https://poppler.freedesktop.org, oline, accessed in December 2025.

[9] FFmpeg developers, "ffmpeg tool", https://ffmpeg.org, oline, accessed in December 2025.

Change History

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.