LectureDoc^2 Tutorial

Author: *Prof. Dr. Michael Eichberg*

Version: 2.1

Introduction

LectureDoc is an authoring system for creating lecture material; i. e., lecture slides, notes and exercises from a single document.

A single LectureDoc document contains discussions of topis which are then used as templates for creating advanced slides as well as a standard document. LectureDoc is intended to be used in combination with rst2ld (reStructuredText to LectureDoc) which is a tool that converts reStructuredText documents into LectureDoc and makes authoring slides as easy as writing a text document.

This tutorial is written in reStructuredText (*rst* in the following) and can be used as a template for creating your own lecture slides. The *code* of this tutorial is available on GitHub: https://github.com/Delors/reStructuredTextToLectureDoc2/blob/main/ld_base_example.en.rst.

Basics

A basic slide consists of a (section) header and some reStructuredText content.

Example

```
Basics
-----------

A basic slide consists of a (section) header
and some reStructuredText content.
```

Embedding Formulae

Embed math equations using reStructuredText's default directive (`.. math::`) and role (`:math:`...``).

Example

The following rst fragment:

```
Computation in :math:`GF(2)`:

.. math::

    \begin{matrix}
        1 + 1 & = 1 - 1 & = 0 \\
        1 + 0 & = 1 - 0 & = 1 \\
        0 + 1 & = 0 - 1 & = 1
    \end{matrix}
```

Will render like this:

Computation in $GF(2)$:

$$\begin{matrix} 1 + 1 & = 1 - 1 & = 0 \\ 1 + 0 & = 1 - 0 & = 1 \\ 0 + 1 & = 0 - 1 & = 1 \end{matrix}$$

You can use `no-title` in combination with the `class` directive to avoid that the title is shown on the slide/document. However, the title is still used for indexes.

Example

```
.. class:: no-title

I will only show up in an index...
----------------------------------
```

Animation

Basic *appear* animations can be created using the (CSS) class `incremental`[1].
You can also define a corresponding custom role (`.. role:: incremental`) to animate parts of a text.

Example

```
Animation
----------

Basic *appear* animations can be created using the (CSS) class
``incremental``. You can also define a corresponding custom role
(``.. role:: incremental``) :incremental:`to animate parts of a text.`

.. example::
    :class: incremental

    ...
```

[1] Animation progress can be reset by pressing the `r` key.

Animation of Lists

In case of (un-)ordered and definition lists (`ol` or `ul` in HTML) it is sufficient to associate the class `incremental-list` using the `class` directive with the list. It is also possible, to only specify the `incremental` class attribute for the required list items.

Example

The following code:

```
.. class:: incremental-list

- this
- is
- a test
```

Will render incrementally like this:

- this
- is
- a test

Slide Dimensions

The slide dimensions can be controlled by specifying the corresponding meta information. If not specified, the dimension is set to 1920×1200 (default); i.e., a ratio of 16:10.

Example

In HTML documents add the following meta tag:

```
<meta name="slide-dimensions" content="1600x1200">
```

In reStructuredText documents add at the beginning:

```
.. meta::
   :slide-dimensions: 1600x1200
```

Associating a document with a unique id

Many functions in LectureDoc2 - e.g. persistence of the slide progress - require that a document is associated with a unique id. This id can be set using the meta directive.
If no id is set, the respective functions are not available.

Example

```
.. meta::
    :id: lecturedoc2-tutorial
    :description: LectureDoc2 Tutorial
    :author: Michael Eichberg
    :license: Released under the terms of the `2-Clause BSD license`.
```

Adding Supplemental Information

Adding information that should not be on the slides, but provide additional information/explanations, can be added using the `supplemental` directive.

Example

```
.. supplemental::

   **Formatting Slides**

   Formatting slides is done using classes and roles.
```

Formatting Slides

Creating heavily formatted slides is easily possible using rst directives and roles which are mapped to CSS classes.

1. Structuring Documents

Creating Sections

Creating a slide which marks the beginning of a new section can be done using the `new-section` class.

Example

```
.. class:: new-section

Structuring Documents
---------------------

.. class:: new-subsection

Creating Sections
-----------------
```

Slide Transitions

Slide transitions can be controlled using the `transition-...` classes[2]:

- `transition-fade`
- `transition-move-left`
- `transition-move-to-top`
- `transition-scale`
- `transition-flip`

Example

```
.. class:: transition-move-to-top

Slide Transitions
-----------------
```

[2] See the LectureDoc2 Cheat Sheet for a comprehensive list of predefined transitions.

Adding Code

Adding code can be done using reStructuredText's code directive.

Example

The following code:

```
.. code:: python
    :number-lines:

    for i in range(0,10):
        print(i)
```

Will render like this:

```
for i in range(0,10):
print(i)
```

Links to External Resources

LectureDoc2 supports links to external resources:

- https://github.com/Delors/LectureDoc2
- LectureDoc2 Sourcecode

Example

```
LectureDoc2 supports links to external resources:

- https://github.com/Delors/LectureDoc2
- `LectureDoc2 Sourcecode <https://github.com/Delors/LectureDoc2>`_
```

Links to Internal Targets

LectureDoc2 supports links to external resources:

- The title of a slide can be used as a link target ➠ Advanced Formatting
- An element which is explicitly marked as a target can be used as a link target:

  ➠ Link Target in Incremental Block

Example

Slide with explicit marked-up element:

```
Adv. Formatting
---------------------

.. container:: incremental

  .. _Link Target in Block:

  See the LectureDoc2 Cheat Sheet.
```

References are defined as follows:

```
Links to internal targets:

- Link to slide: `Adv. Formatting`_
- Link to a marked-up element:

  `Link Target in Block`_
```

Scientific Citations

Citations are fully supported in LectureDoc2.

A reference to a book: [Martin2017] (Details are found in the bibliography (see next slide)).

Example

```
A reference to a book: [Martin2017]_
```

Bibliography

- [Martin2017] Clean Architecture: A Craftsman's Guide to Software Structure and Design; Robert C. Martin, Addison-Wesley, 2017
- ...

Example

```
.. [Martin2017] Clean Architecture: ...; Robert C. Martin, Addison-Wesley, 2017
```

Advanced Formatting

LectureDoc comes with a set of predefined (CSS) classes that can be used to format the slides. Some of these classes have explicit support by LectureDoc and will be rendered differently in the different situations (e.g., document view vs. slide view will render *stacked layouts* or *supplemental information* differently).

- red
- peripheral
- ~~obsolete~~

See the LectureDoc2 Cheat Sheet for a comprehensive list of predefined CSS classes.

Stacked layouts

Stacked layouts enables updating parts of a slide by putting the content into layers and then showing the layers incrementally.

Example

This text is gray.

```
.. dech:: monospaced

  .. card::

    :gray:`This text is gray.`

  .. card:: overlay

    xxxxxxxxxxxxxxxx
```

Presenter-Notes

Presenter notes can be added to a slide using the `presenter-note` directive.

A presenter note - including its presence - is only visible after entering the master password (press `m` and then enter: `123456`).

Example

```
.. presenter-note::

   This is a presenter note.

   It is only visible after entering the master password (123456).
```

Integrated Exercises

Exercises can be integrated into the slide set.

Example

1.1. Exercise: 1+1

Compute: $\sqrt{2} = ?$

To unlock the solution go to the document view (press `c`) and enter the password (sqrt).

```
.. exercise:: Exercise: 1+1

    Compute: :math:`\sqrt 2 = ?`.

    .. solution::
        :pwd: sqrt

        Solution: :math:`1,4142135624`.
```

If you have multiple exercises, you can define a master password (123456) to unlock all solutions at once (press `m` to open the dialog).

```
.. meta::
    :master-password: 123456
```

2. Images

Example

```
.. class:: padding-none no-title transition-scale

Image in the Background
-----------------------

.. dech::

    .. card::

        .. image:: ld_base_example/tag_cloud.webp
           :width: 100%
           :align: center

    .. card:: overlay

        Content on the slide...
```

Inline SVGs

Inline SVGs are fully supported by LectureDoc, but styles and definitions that are used in multiple inline SVGs have to be centralized!

This is due to the copying of the slide templates which - if you use ids to reference definitions in the SVGs - makes them no longer unique. This is a violation of the spec and causes troubles in Chrome and Firefox.

Adding Shared Definitions

To add shared SVG definitions, use the `.. meta::` directive and the `:svg-defs:` property.

Example

```
:svg-defs:
    <marker id="arrow"
        viewBox="0 0 10 10" refX="10" refY="5"
        markerUnits="strokeWidth"
        markerWidth="4" markerHeight="4"
        orient="auto-start-reverse">
        <path d="M 0 0 L 10 5 L 0 10 z" /></marker>
    <g id="star">
        <polygon
            class="star"
            points="12,2 15,9 22,9 16,14 18,21 12,17 6,21 8,14 2,9 9,9"
            fill="gold"
            style="transform: scale(0.05)"/></g>
```

Defining Shared Styles

To add shared SVG styles, use the `.. meta::` directive and the `:svg-style:` property.

Example

```
:svg-style:
    .std-line {
        stroke:rgb(0,0,0);
        stroke-width:0.2ch;
        stroke-dasharray: 1 1;
    }
```

Example

Use of the previously defined class `std-line` (line 7), `star` (`href="#star"` line 8) and `arrow` (`marker-end="url(#arrow)"` line 11).

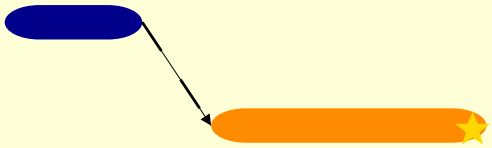
```
<div    style="width: 90ch; height:16ch">
<svg    version="1.1" xmlns="http://www.w3.org/2000/svg"
        viewBox="0 0 48 8" font-size="0.75"  >
    <rect   width="4" height="1" x="8" y="3" rx="1" ry="1"
            style="fill:darkblue" />
    <line   x1="4" y1="3.5" x2="8" y2="3.5"
            class="std-line"
            marker-end="url(#arrow)"/>
    <rect   width="8" height="1" x="14" y="6" rx="1" ry="1"
            style="fill:darkorange" />
```

```
    <use    href="#star" x="21" y="6" />
</svg>
</div>
```

The example also demonstrates how to define an SVG whose size is completely dependent on the size of the surrounding font-size.

Example

Rendered SVG



Embedding Images

In general, embeddings images is done using the image directive. However, due to the fact that we render the topic once as classical slides and once in a document-oriented way, it is important to understand how LectureDoc handles images.

General Guidelines

In general an image should be designed/generated/created with its usage on a slide in mind. That is, an image should fit on a slide with a *logical resolution* of 1900 by 1080/1200 pixels. Hence, if text is found on the image it should not be smaller than 30px; ideally it should use the same font size as used by the slide. Those images should then be added to the document using the image directive. In this case it is optional to specify the width and/or height. Such images will be automatically scaled by LectureDoc when the content is shown in the document view. The scaling factor is determined by the ratio between the default font-size used for the document and the default font-size used for the slides.

HighDPI Images

As said, images are generally assumed to have a resolution that fits a slide. However, in many cases the source image may have a resolution that is (much) higher. In this case, it is possible to scale the image using the directive's width and/or height attribute. LectureDoc will then update the width and height attributes when shown in document mode. This requires that the images' width and heights are given in pixels.

Images/SVGs With Font-size Dependent Sizing

SVGs where the size is (alread) dependent on the font-size should not specify any width or height attributes.