LectureDoc2 Renaissance

Autor: Prof. Dr. Michael Eichberg
Kontakt: michael.eichberg@dhbw.de, Raum 149B

Folien: https://delors.github.io/LectureDoc2/renaissance/folien.en.rst.html
Fehler melden: https://github.com/Delors/delors.github.io/issues

1. Introduction

This slide set primarily serves as a testbed for various features of the *LectureDoc2* system. It also demonstrates how to use the system to create slides for a lecture.

Supplemental Information and Presenter Notes[1]

You first have to enter the master password (press M and then enter Demo!) to see the presenter notes.

This is additional information that is not shown in the main text.

Some more text.

This is some more additional information that is not shown in the main text.

Just a long text to demonstrate that the supplemental information is not immediately shown on the slide and that it can be scrolled. Just a long text to demonstrate that the supplemental information is not immediately shown on the slide and that it can be scrolled. Just a long text to demonstrate that the supplemental information is not immediately shown on the slide and that it can be scrolled. Just a long text to demonstrate that the supplemental information is not immediately shown on the slide and that it can be scrolled. Just a long text to demonstrate that the supplemental information is not immediately shown on the slide and that it can be scrolled. Just a long text to demonstrate that the supplemental information is not immediately shown on the slide and that it can be scrolled. Just a long text to demonstrate that the supplemental information is not immediately shown on the slide and that it can be scrolled. Just a long text to demonstrate that the supplemental information is not immediately shown on the slide and that it can be scrolled. Just a long text to demonstrate that the supplemental information is not immediately shown on the slide and that it can be scrolled. Just a long text to demonstrate that the supplemental information is not immediately shown on the slide and that it can be scrolled. Just a long text to demonstrate that the supplemental information is not immediately shown on the slide and that it can be scrolled. Just a long text to demonstrate that the supplemental information is not immediately shown on the slide and that it can be scrolled. Just a long text to demonstrate that the supplemental information is not immediately shown on the slide and that it can be scrolled. Just a long text to demonstrate that the supplemental information is not immediately shown on the slide and that it can be scrolled. Just a long text to demonstrate that the supplemental information is not immediately shown on the slide and that it can be scrolled. Just a long text to demonstrate that the supplemental information is not immediately shown on the slide and that it can be scrolled. Just a long text to demonstrate that the supplemental information is not immediately shown on the slide and that it can be scrolled. Just a long text to demonstrate that the supplemental information is not immediately shown on the slide and that it can be scrolled. Just a long text to demonstrate that the supplemental information is not immediately shown on the slide and that it can be scrolled.

---

[1] Supplemental information and presenter notes are not immediately shown on the slide.

Decks and Cards

A deck is a collection of cards.

Where each card "replaces" the previous cards during the presentation belonging to the same deck.

This card contains a simple note. Where the height of the deck as a whole is determined by the tallest card.

Note

This is a simple note.

The Tallest One

Above the crowd, I stand so high, A bridge between the ground and sky. I see the world in a broader frame, Yet hear the jokes—they're all the same.

—Jan. 2025 ChatGPT (Prompt: I need a short poem about being the tallest one.)

Decks can be nested and can overlay each other!

However, a card with a nested deck is not allowed to also use floating elements (e.g. notes). In general, the use of floating elements in combination with overlays is discouraged.

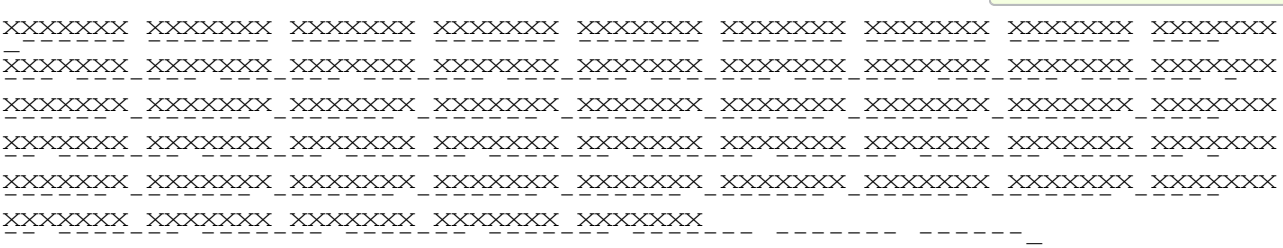
```
The first sentence of the first card in the nested deck.

A sentence in between.

The last sentence of the first card in the nested deck.
```

Hint

This is the last meaningful card in the nested deck. The next two ones are a technical detail.

Note

This is another simple note.



Hint

This is the last card in the top-level deck.

Stories

Stories are used for content that should appear in a stepwise manner and which may scroll content out of the view.

1. This is the first step.
2. This is the second step.
3. This is the third step.
4. This is the fourth step.
5. This is the fifth step.
6. This is the sixth step.
7. This is the seventh step.
8. This is the eighth step.
9. This is the ninth step.
10. This is the tenth step.
11. This is the eleventh step.
12. This is the twelfth step.
13. This is the thirteenth step.
14. This is the fourteenth step.
15. This is the fifteenth step.

```
Some monospaced text.
```

```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

```python
print("Hello, World!")
```

```rust
fn main() {
    println!("Hello, World!");
}
```

```zig
const std = @import("std");

pub fn main() void {
    std.debug.print("Hello, World!\n", .{});
}
```

Scrollables

A scrollable is a container whose content does not fit into the available space of a slide. During the presentation the content can be scrolled by the presenter and scrolling is relayed in secondary windows.

```
/* A small library to encrypt and decrypt strings using AES-GCM and PBKDF2.
 *
 * Based on code found at: https://github.com/themikefuller/Web-Cryptography
 *
 * License: BSD-3-Clause
 */
export {
    decrypt as decryptAESGCMPBKDF,
    encrypt as encryptASEGCMPBKDF
}

async function encrypt(plaintext, password, iterations) {

    const encodedPlaintext = new TextEncoder().encode(plaintext);
    const encodedPassword = new TextEncoder().encode(password);

    const pass = await crypto.subtle.importKey(
        'raw',
        encodedPassword,
        { "name": "PBKDF2" },
        false,
        ['deriveBits']);

    const salt = crypto.getRandomValues(new Uint8Array(32));
    const iv = crypto.getRandomValues(new Uint8Array(12));

    const keyBits = await crypto.subtle.deriveBits(
        {
            "name": "PBKDF2",
            "salt": salt,
            "iterations": iterations,
            "hash": { "name": "SHA-256" }
        },
        pass,
        256);

    const key = await crypto.subtle.importKey(
        'raw',
        keyBits, { "name": "AES-GCM" },
        false,
        ['encrypt']);

    const enc = await crypto.subtle.encrypt(
        {
            "name": "AES-GCM",
            "iv": iv
        },
        key,
        encodedPlaintext);

    const iterationsB64 = btoa(rounds.toString());

    const saltB64 = btoa(Array.from(new Uint8Array(salt)).map(val => {
        return String.fromCharCode(val)
    }).join(''));
```

```
    const ivB64 = btoa(Array.from(new Uint8Array(iv)).map(val => {
        return String.fromCharCode(val)
    }).join(''));

    const encB64 = btoa(Array.from(new Uint8Array(enc)).map(val => {
        return String.fromCharCode(val)
    }).join(''));

    return iterationsB64 + ':' + saltB64 + ':' + ivB64 + ':' + encB64;
};

async function decrypt(encrypted, password) {

    const parts = encrypted.split(':');
    const rounds = parseInt(atob(parts[0]));

    const salt = new Uint8Array(atob(parts[1]).split('').map(val => {
        return val.charCodeAt(0);
    }));

    const iv = new Uint8Array(atob(parts[2]).split('').map(val => {
        return val.charCodeAt(0);
    }));

    const enc = new Uint8Array(atob(parts[3]).split('').map(val => {
        return val.charCodeAt(0);
    }));

    const encodedPassword = new TextEncoder().encode(password);
    const pass = await crypto.subtle.importKey(
        'raw',
        encodedPassword,
        { "name": "PBKDF2" },
        false,
        ['deriveBits']);

    const keyBits = await crypto.subtle.deriveBits(
        {
            "name": "PBKDF2",
            "salt": salt,
            "iterations": rounds,
            "hash": {
                "name": "SHA-256"
            }
        },
        pass,
        256);

    let key = await crypto.subtle.importKey(
        'raw',
        keyBits, { "name": "AES-GCM" },
        false,
        ['decrypt']);

    let dec = await crypto.subtle.decrypt(
        {
            "name": "AES-GCM",
            "iv": iv
        },
        key,
```

```
        enc);

    return (new TextDecoder().decode(dec));
};
```

Scrollables with explicit height!

A scrollable can have an explicit height that will be used for the slide view.

```
/**
 * Adds an event listener to the scrollable element that fires when the element
 * is scrolled. In that case, the event is sent to the specified channel to
 * make secondary windows aware of the scrolling event in the primary window.
 *
 * The data is sent using the {@link postMessage} method where the msg is the event title
 * and the data is a two element array where the first element is the id of the
 * element that is being scrolled and the second element is the current scrollTop.
 *
 * The primary window is always the window that user interacts with. The secondary
 * is every other window showing the same site.
 *
 * @param {Channel} channel - The channel that will be used to send the event.
 * @param {string} eventTitle - The title of the event that will be sent to the channel. The
 *                              title has to be unique w.r.t. to the channel.
 * @param {HTMLElement} scrollableElement - The element that is being scrolled.
 * @param {string} id - The id of the element that is being scrolled.
 */
export function addScrollingEventListener(channel, eventTitle, scrollableElement, id) {
    // We will relay a scroll event to a secondary window, when there was no
    // more scrolling for at least TIMEOUTms. Additionally, if there is already an
    // event handler scheduled, we will not schedule another one.
    //
    // If we would directly relay the event, it may be possible that it will
    // result in all kinds of strange behaviors, because we cannot easily
    // distinguish between a programmatic and a user initiated scroll event.
    // (Using window blur and focus events didn't work reliably.)
    // This could result in a nasty ping-pong effect where scrolling between
    // two different position would happen indefinitely.
    const TIMEOUT = 50;
    let lastEvent = undefined;
    let eventHandlerScheduled = false;
    scrollableElement.addEventListener("scroll", (event) => {
        lastEvent = new Date().getTime();
        function scheduleEventHandler(timeout) {
            setTimeout(() => {
                const currentTime = new Date().getTime();
                if (currentTime - lastEvent < TIMEOUT) {
                    scheduleEventHandler(TIMEOUT - (currentTime - lastEvent));
                    return;
                }
                postMessage(channel, eventTitle, [id, event.target.scrollTop]);
                // console.log(eventTitle + " " + id + " " + event.target.scrollTop);
                eventHandlerScheduled = false;
            }, timeout);
        };
        if(!eventHandlerScheduled) {
            eventHandlerScheduled = true;
            scheduleEventHandler(TIMEOUT);
        }
    },{passive: true});
}
```

This is a scrollable that extends to the bottom of the slide -100px to leave some space for the footer.

```
export function getTopAndBottomMargin(e) {
    const style = window.getComputedStyle(e);
```

```
    return parseInt(style.marginTop) + parseInt(style.marginBottom);
}
export function getLeftAndRightMargin(e) {
    const style = window.getComputedStyle(e);
    return parseInt(style.marginLeft) + parseInt(style.marginRight);
}
export function getLeftAndRightPadding(e) {
    const style = window.getComputedStyle(e);
    return parseInt(style.paddingLeft) + parseInt(style.paddingRight);
}
export function getLeftAndRightMarginAndPadding(e) {
    return getLeftAndRightMargin(e) + getLeftAndRightPadding(e);
}

export function postMessage(channel, msg, data) {
    channel.postMessage([msg, data]);
}
```

Simple column-based Layouts

One way to create a very simple multi-column layout consists of a list with the class `columns`.

Example

Default layout:

This first column.

1. a nested list.

The second column.

1. a nested list.

Left-aligned layout:

This first column.

1. a nested list.

The second column.

1. a nested list.

Evenly-spaced layout:

This first column.

1. a nested list.

The second column.

1. a nested list.

Advanced Slide Layouts

Using Grids it is possible to design advanced slide layouts.

When you don't specify a specific layout for a grid
a simple multi-column layout is used.

```
.. grid::
    :class: very-light-gray-background

    .. cell::

        Using Grids it is possible to
        design advanced slide layouts.

    .. cell::

        .. code:: rst
            :class: copy-to-clipboard

            **The Code**
```

Math

Adding math (e.g. \(a^2+b^2=c^2\)) to a slide is done using the math directive or role.

\begin{equation*} e = mc^2 \end{equation*}
Poor Man's Math: $e = mc^2$.

Example

```
Adding math (e.g. :math:`a^2+b^2=c^2`) to a slide is done using the math directive or role.

.. math::

    e = mc^2
```

Tables

.highlight-identical-cells-on-hover

	r	s	t	u	v	w	x	y	z
a	1	2	3	4	5	6	7	8	9
b	4	5	6	7	8	9	1	2	3
c	7	8	9	1	2	3	4	5	6

.highlight-row-on-hover

	r	s	t	u	v	w	x	y	z
a	1	2	3	4	5	6	7	8	9
b	4	5	6	7	8	9	1	2	3