Von der Präsentation zum Dokument

Die Veröffentlichung von Vortragsfolien ist nicht immer sinnvoll, besser ist es, die Unterlagen so aufzubereiten, dass daraus ein richtiges Dokument entsteht.

1) Fertige Suite oder HTML

Das ist die Frage.

- Impress
- PowerPoint, geht sowieso nicht

Diese Werkzeuge erzeugen Seiten, die nicht für alle Ausgabemedien gleichermaßen geeignet sind. Eine ergänzende Dokumentation zur Ausgabe als PDF-Datei mit zusätzlichen Erläuterungen ist nicht vorgesehen.

HTML hingegen kann auf den verschiedenen Geräten ausgegeben werden. Inhalt und Layout sind getrennte Komponenten. Das bedeutet, dass man sich auf den Inhalt konzentrieren kann. Die Layout-Komponente sorgt dann für das passende Erscheinungsbild.

2) Erstellen der Unterlagen

- **Texteditor**, zb. **geany** oder **ghostwriter**:
  - Format *markdown*
- **Pandoc**:
  - Generierung des html-Codes.
- Shell Script build.sh
  - Automatisierung der Dateierstellung.
- Eigene Templates;
  - für Html- und PDF-Dateien.
- **Weasyprint** um die PDF-Datei zu erstellen, alternativ **prince xml** (Barrierefreie PDF).

3) Folien erstellen

- Eine Reihe von Slide Frames ist verfügbar:
  - slideous
  - s5
  - dzslides
  - revealjs
  - slidy

3.1) Nachteile der vorhandenen Lösungen

- Eine Überarbeitung des Codes auf Markdown oder HTML-Ebene ist erforderlich.
- Unterschiedliche CSS-Dateien sind notwendig.
- Der Slide-Code entspricht nicht immer den Erwartungen.
- Unterstützung für mobile Geräte nicht optimal.

Mobile Geräte haben unterschiedliche Seitenverhältnisse. Dies führt dazu, dass eine ansonsten (auf dem PC) passende Folie fehlerhaft dargestellt wird.

Ein einfaches Umschalten in den Vollbildmodus ist bei den untersuchten Lösungen nicht vorhanden.

Auch ein virtueller Marker (Laserpointer) fehlt, dieser wurde in der Eigenlösung implementiert.

3.2) Eigene Lösung

> Wenn der Berg nicht zum Propheten kommt …
> Erstellt man das Passende!

- Geeignet für mobile Geräte:
  - Vollbildmodus wird unterstützt.
- Überlange Folien können angezeigt werden.
- Einzelne Listenelemente können nacheinander angezeigt werden (Standard).
- Unterlistenelemente können zusammen mit dem Hauptlistenelement angezeigt werden.
- Wenn etwas nicht auf einen kleinen Bildschirm passt, kann gescrollt werden.
- Laserpointer-Funktionalität.

3.3) Anpassung der Vorlagen

- CSS in verschiedenen Dateien:
  - Dadurch können leichter Anpassungen (Farbe, Schriftgröße, …) vorgenommen werden.
  - Angepasste CSS-Dateien können im Präsentationsverzeichnis abgelegt werden, z.B. theme.css für die Farbe.

4) Vorgehensweise

Planung ist notwendig!

Der Aufbau muss durchdacht sein. Zuerst müssen die Folien erstellt werden.

Die Folientitel können auf verschiedenen Ebenen wie in den Gesamtdokumenten vorgesehen sein.

Danach kann die ausführliche Dokumentation (html und PDF) ergänzt werden. Dieser Text kann z.B. aus normalen Absätzen bestehen, diese sind auf den Folien nicht sichtbar.

Das fertige Gesamtdokument, HTML-Seite oder PDF-Ausgabe, wird dank Layout-Regeln (CSS) sauber generiert, inklusive einer Nummerierung, die auf der Folie nicht unbedingt sinnvoll ist.

Danach kann die ausführliche Dokumentation (html und PDF) hinzugefügt werden. Dieser Text kann z.B. aus normalen Absätzen bestehen. Diese auf den Folien nicht sichtbar.

Das fertige Gesamtdokument, HTML-Seite oder PDF-Ausgabe wird durch Layout-Regeln (CSS) sauber zusammengefügt. Eine Nummerierung, die auf der Folie nicht unbedingt sinnvoll ist, wird ebenfalls durchgeführt.

5) Automatisierung

Es werden drei Dokumenttypen erstellt:

- Folien.
- Vollständiges HTML-Dokument.
- Vollständiges PDF-Dokument.

5.1) Build Skript - Anpassung

```
#!/bin/sh
file=slides-zur-doc
template=$HOME/TEMPLATES/mslide

coverPage=true
newPage=true
color=false
titleCount=true
ownSlideCss=ul.css
ownDocCss=docul.css
usePrince=false
```

Die Variablen müssen angepasst werden.

file ist der Basisname der Dateien. Die Markdown-Datei heißt **slides-zur-doc**.md. Die generierten Dokumente sind:

- **slides-zur-doc**-slide.html
- **slides-zur-doc**.html
- **slides-zur-doc**.pdf

coverPage=true Die Titelseite wird als Deckblatt ausgegeben.

newPage=true Jedes Hauptkapitel beginnt auf einer neuen Seite.

color=false Die nummerierten Codeabschnitte sind schwarz auf weiß gedruckt.

titleCount=true Die Kapitel sind nummeriert.

ownSlideCss und **ownDocCss** Name der privaten CSS-Datei für die Präsentation bzw. der PDF-Datei.

usePrince steuert die PDF-Ausgabe über **WeasyPrint** (*usePrince=false*) oder **Prince xml** (*usePrince=true*).

5.2) Scriptauzug - Folien

```
pandoc -L "$template/convertHeader.lua" \
-i "$file.md"  \
--resource-path="$resourcePath" \
--to html5 \
--section-div \
--template mslide.html \
--include-in-header stb.html \
--include-in-header main.css \
--include-in-header theme.css \
--include-in-header color.css \
--include-in-header "$ownslidecss" \
--include-in-header color.css \
--include-in-header anim.css \
--include-in-header ste.html \
--include-in-header jsb.html \
--include-in-header mslide.js \
--include-in-header jse.html \
--include-before-body $help |\
$SED 's/href="#cb[0-9][0-9a-z-]*"//' |\
data-uri -i - -o "$file-slide.html"
```

Die Parameter `--include-in-header` sorgen für einen sauberen HTML-Code. stb.html, ste.html, jsb.html und jse.html deklarieren nur Begin- und Ende der Style- und Javascript-Abschnitte.

Die entsprechenden Style- und Javascript-Dateien müssen zwischen den oben genannten Dateien eingebunden werden.

Um Anpassungen zu erleichtern, werden diese Dateien zunächst im aktuellen Verzeichnis und, falls nicht vorhanden, im Template-Verzeichnis gesucht (Zeile `` `--resource-path ".:$template" ``).

5.3) Scriptauszug - HTML/PDF

```
pandoc -i "$file.md"  \
--section-div \
--resource-path ".:$template" \
--to html5 --template mweb.html \
--section-divs \
--include-in-header stb.html \
--include-in-header docmain.css \
--include-in-header "$titlecount" \
--include-in-header "$colorcss" \
--include-in-header "$owndoccss" \
--include-in-header "$coverpage" \
--include-in-header "$newpage" \
--include-in-header page-$lang.css \
--include-in-header ste.html |\
$SED 's/href="#cb[0-9][0-9a-z-]*"//'|\
$SED 's/><a ><\/a>/> <a><\/a>/' |\
cover | data-uri -i - -o  "$file.html"
weasyprint -s "$template/weasyprint.css" "$file.html" -o "$file.pdf"
```

Auch hier gelten die oben gemachten Aussagen, wobei Dateien, die nicht für eine PDF-Datei bestimmt sind, z.B. anim.css oder javascript-Dateien, nicht aufgeführt werden. Die Layoutdatei main.css wird durch die Datei docmain.css ersetzt.

5.4) Erstellung der Dateien

- Im Terminal *build.sh* aufrufen

6) Template Dateien

6.1) Präsentation

```
build.sh
convertHeader.lua (Untertiteln auf Ebene 1 bringen)
mslide.js
mslide.html       (Template Datei)
main.css
color.css
anim.css
help-de.css
help-en.css
...
```

mslide.js steuert den Ablauf der Präsentation.

mslide.html ist von der Pandoc Templatedatei default.html5 abgeleitet und berücksichtigt die eigenen Variablen (opts.dark, ...

Die separaten CSS-Dateien sollen es erleichtern, eigene Anpassungen vorzunehmen.

6.2) HTML - PDF

```
build.sh
convertHeader.lua (Untertiteln auf Ebene 1 bringen)
mweb.html       (Template Datei)
docmain.css
page-de.css
...
empty.css
```

7) Schriftgrad: CSS

```
body {
    font-size:max(16px,2vw);
}
```

Durch die Angabe von *max(16px,2vw)* erfolgt die Skalierung automatisch, sie wird vom Browser selbstständig stufenlos angepasst. Alle Größenangaben müssen relativ sein *(rem, em, %)*.

1 vw entspricht 1 % der Breite des Ausgabefensters. Für einen Bildschirm mit Breite von 1920px ist die Schriftgröße dementsprechend 19,20*2 = 38,40 px. Für einen Bildschirm mit einer Breite von 1280px würde die Schriftgröße 25,6 px betragen.

Wenn die Breite des Browsers verringert wird, passt sich die Schriftgröße automatisch an, es wird jedoch kein Wert eingestellt, der kleiner als 16 px ist.

7.1) Schriftgrad für Smartphones

```
body {
    font-size:max(12px,3.9vmin);
}
```

Die Berechnung der Schriftgröße basiert hier auf der Höhe des Wiedergabefenster. Bei einem besonders großen Seitenverhältnis wird die Schriftgröße so angepasst, dass eine Folie passend dargestellt wird.

Es gibt jedoch Grenzen: Die Schriftgröße darf nicht kleiner als 12 px sein.

Der “Wide”-Modus kann mit der Taste **[w]** oder Touch-event ein- und ausgeschaltet werden.

8) Schriftgröße der einzelnen Elemente

Selector	CSS
body	line-height: 1.5em;
.titel	font-size: 3em; line-height: 1.3em
h1	font-size: 1.3em;
pre	font-size: .96rem; line-height: 1.3em

Falls eine größere Schriftgröße (weniger Zeilen / Dia) gewünscht wird, können die entsprechenden CSS-Anweisungen in einer eigenen neuen CSS-Datei verankert werden. Zum Beispiel schrift-groesse.css. Die Datei build.sh muss entsprechend ergänzt werden:

```
1 body { font-size: 32px; } /* alles um den Faktor 2 skalieren */
2                           /* (nur HTML-Datei)                */
```

9) Bildgröße

Bilder müssen ebenfalls skaliert werden. Dazu muss die Höhe oder Breite angegeben werden.

Da die Höhe des Ausgabefensters entscheidend ist, ist die Festlegung der Bildhöhe der bessere Weg.

Da die Folien auch eine zweispaltige Darstellung ermöglichen sollen und die Bilder sehr unterschiedliche Größen und Seitenverhältnisse haben können, müssen CSS-Regeln vorhanden sein, die in der Regel eine korrekte Skalierung ermöglichen.

```
img {
    max-height:70vh;
    width:100%;
    object-fit: contain;
}
.h80 img, img.h80 { max-height: 80vh;}
.h75 img, img.h75 { max-height: 75vh;}
.h70 img, img.h70 { max-height: 70vh;}
...
.h20 img, img.h20 { max-height: 20vh;}
.h15 img, img.h15 { max-height: 15vh;}
.h10 img, img.h10 { max-height: 10vh;}
```

In den meisten Fällen sollte dies funktionieren. Bei Bedarf kann die Höhe des Bildes durch die Angabe der Höhe .h10 bis .h80 angegeben werden.

Für die HTML- bzw. PDF-Ausgabe sind die entsprechenden CSS-Eigenschaften enthalten, die Höhenangabe muss entweder in *px*, *em* oder besser *mm',* erfolgen.

Eine Lösung ist die Berechnung der Höhe:

```
@media screen {
img { height: calc(800px * .70); }
.h80 img, img.h80 { height: calc(800px * .80); }
...
}
```

Für die PDF-Ausgabe wird die Berechnung von **weasyprint** nicht unterstützt, daher ist im Abschnitt `@media print {` für jeden Eintrag die entsprechende Höhe in mm angegeben werden.

10) CSS-Farben

```
:root {
    --background: #eeffff;
    --color: #000000;
    --h1-shadow: #ffffff;
...
    --co: hsl(192,33%,53%);
...
}
html[mode=dark]:root {
    --background: #110000;
    --color: #ffffff;
    --h1-shadow: #000000;
...
    --co: hsl(192,33%,47%);
...
}
```

Um Farbanpassungen zu erleichtern, sind die Definitionen in der Datei color.css hinterlegt.

:root ist eine Pseudoklasse, die an das HTML-Element angehängt werden kann. Die Definitionen werden wie üblich an die Unterelemente vererbt.

Die Verwendung des entsprechenden Regelsatzes wird z.B. durch den Selektor **html[mode=dark]:root** bewirkt.

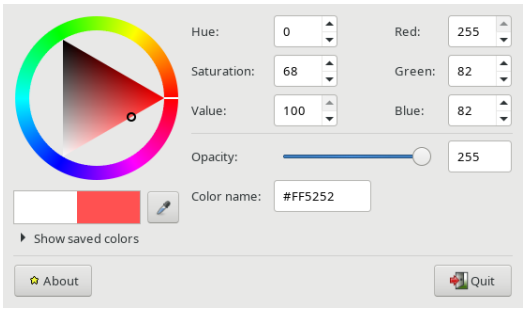
Damit dies funktioniert, muss im HTML-Element ein Attribut “mode” mit dem Wert “dark” per Javascript hinzugefügt werden. Natürlich können auch andere Namen verwendet werden. Dies muss jedoch im Javascript berücksichtigt werden. Es könnten auch viel mehr Darstellungsmodi vorgesehen werden, z.B. “weiß”, “grau”, “schwarz”…

Durch die Zuweisung von Farben über Variablen können die Folien leicht an die der Umgebung und den Wünschen des Betrachters angepasst werden.

10.1) Hilfswerkzeuge

Farbwerte können z.B. unter Linux mit:

KColorchooser



Gcolor2 / Gcolor3

bestimmt werden.

10.2) Dokumentation

Die Anforderungen an die Ausgabe für Printmedien und Bildschirme sind unterschiedlich.

- **web.html** Template Datei zum Aufbau der HTML Datei.
- **docmain.css** CSS definition für HTML und Druckausgabe.
- **page-de.css** Regeln für den Druck (Ausgabe als PDF).

Das PDF-Format eignet sich wesentlich besser für die Papierausgabe als für den Ausdruck über den Browser. CSS kennt Anweisungen zur Steuerung des Seitenaufbaus für die Druckausgabe. Leider ist dies im Browser nicht implementiert. Die Erzeugung der PDF-Datei mittels **weaswyprint** berücksichtigt diese Attribute und sorgt für eine geeignete Druckausgabe.

10.3) Eigene Anpassungen

- Zusätzliche ***.css**
  - müssen in die Datei build.sh eingefügt werden.
- Kopieren Sie die entsprechende ***.css** Datei aus dem Template-Verzeichnis.
  - Style-Änderungen in der lokalen ***.css** vornehmen.

10.4) Beispiel für eigene Anpassungen

Für die PDF-Erzeugung: schwarz/weiß (`docul.css`):

```
li::marker { content: '➪ '; }
li li::marker { font-size: 0.8em ;}
ul a::after { content: " (" attr(href) ")"; }
```

Für die Folienerstellung: farbig (`ul.css`):

Statt eines Punktes wird für Listenelemente ein Sonderzeichen verwendet.

Da die Schriftgröße der Folien den Standardeinzug des -Elements überschreiten kann, wird *padding-inline-start* auf eine Breite von mindestens 1em gesetzt.

```
ul { padding-inline-start: max(1em,40px); }
li::marker { content: '➪'; }
li li::marker { font-size: 0.8em; }
li::marker { color: #057b39;}
html[mode=dark] li::marker { color: gold; }
```

Sowohl **weasyprint** als auch **prince xml** können unterschiedliche CSS-Eigenschaften benötigen.

Diese können gezielt als Parameter übergeben werden. Für **Weasyprint** ist die Datei ``weasyprint.css``` im Template-Verzeichnis enthalten und im Skript build.sh eingetragen.

11) Bedienung

> Ohne Kurzanleitung geht es nicht!

Da die Präsentationen über verschiedene Systeme (PC, Smartphone, Tablet) gezeigt werden sollen, wird die Bedienung über Tastatur, Maus oder Touch-event unterstützt. Touch-events simulieren nur die Bedienung mit der Maus.

11.1) Tastatur

```
Cursor-Tasten: Blättern.
[h] Hilfe ein-/ausblenden.
[v] Mobil: Hilfe ein-/ausschalten.
[m] Darstellung mit hellen/dunklen Farben.
[b] Folien ausblenden/einblenden.
[f] Vollbildmodus ein-/ausschalten.
[a] Animation ein-/ausschalten.
[l] Virtueller Laserpointer ein/aus.
[w] Breitenmodus zur Berechnung der Schriftgröße an/aus.
```

Ein weiterer Modus ist verfügbar. Mit der Taste **P** werden alle (vollständig ausgefüllten) Folien angezeigt. Damit kann ein PDF-Dokument aus dem Browser erzeugt oder ausgedruckt werden.

11.2) Maus / Finger

```
Links oben: rückwärts blättern.
Oben rechts: Vorwärts blättern.
Oben Mitte: Vollbild an/aus.
Unten links: Bedienmenü einblenden.
```

Die Beienfelder zum “verfetten” des Bildschirms, werden nicht angezeigt, eine Ausnahme stellt das eingeblendete Element zur Einstellung des Betriebsmodus dar.

11.3) Bedienung Mobile Geräte.

Fingergerecht?

Die rot umrandeten Rechtecke sind die Bedienelemente für die Steuerung mit Touch-Gesten (Smartphone).

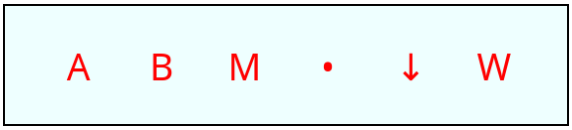
Das mittlere obere Feld steuert den Vollbildmodus (auch über die Taste **[f]** erreichbar).

Die oberen Felder links und rechts ermöglichen den Wechsel zur vorhergehenden bzw. zur nächsten Folie.

Unten links wird das Menü geöffnet, um die Modi einzustellen.

Das Feld rechts unten ermöglicht die Hervorhebung der Schaltflächen.

11.3.1) Einstellung der Darstellungsarten



Hilfmenü

- **A** Animation ein-/ausschalten.
- **B** (Blank) Inhalt sichtbar/unsichtbar.
- **M** Hintergrund hell/dunkel.
- • Virtuelle Zeiger an/aus.
- ↓ Menü schließen.
- **W** Schriftgrößenberechnung abhängig von der Höhe.

Das Menü wird in der linken unteren Ecke angezeigt und ermöglicht einige Einstellungen.

12) Folien als PDF ausdrucken

Der Browser wird benötigt.

Das Drucken der Taste **[p]** fügt dem Element *html* die Klasse **print=“print”** hinzu. Dies hat zur Folge, dass alle Folien sichtbar werden und somit eine Druckausgabe mit dem Browser ausgelöst werden kann.

Die Ausgabe eignet sich eher zur Dokumentation, Listenelemente, die in der HTML-Präsentation nacheinander angezeigt werden, erscheinen alle auf einer Seite, Zwischenstufen werden nicht berücksichtigt.

Die „PDF”-Ausgabe sollte im Querformat und in Farbe erstellt werden.

13) Markdown Datei

```
---
title: Vom Vortragsfolien zur Dokumentation
author: Jean-Jacques Sarton
lang: de
opts:
  dark: true
  anim: true
  incMode: true
---

# Vom Vortragsfolien zur Dokumentation

> Vortragsfolien publizieren ist nicht immer sinnvoll, es wäre besser
die Unterlagen so aufzubereiten, dass ein richtige Dokument entsteht

## Erstellen der Unterlagen

* Texteditor, z.b. **geany**
...
### Eigene Lösung
...
```

Die Überschriften (**#** bis **######**) müssen mit der Dokumentation übereinstimmen. Die Folien sollten nur eine Ebene haben, sonst gibt es Probleme. Als Vorverarbeitung werden im Markdown-Stream alle Titelebenen auf ein einfaches **#** „geschrumpft”.

Der von Pandoc erzeugte HTML-Code verschachtelt die verschiedenen Abschnitte wenn das Dokument verschiedene Überschriftenebenen hat. Dies ist für eine Präsentation nicht sinnvoll. Daher ist es notwendig, alle Titel und Untertitel in <h1>-Titel umzuwandeln. Dies geschieht durch ein ``lua``` Miniscript.

Die Absätze erscheinen nicht auf der Folie, können aber wie folgt eingeblendet werden.

```
:::{.vis}
Paragraf 1

Paragraf 2
:::
```

Es ist auch möglich, spezielle Seiten zu erstellen, die wie die erste Seite aussehen und auf der HTML-Seite und der PDF-Datei nicht angezeigt werden.

```
:::{.noDoc}
# Titel

Paragraf
:::
```

14) Barrierefreie Bilder

Die Bilder sollten eine kurze Beschreibung enthalten (Attribut alt). Das Bild sollte auch einen Untertitel enthalten.

```
:::{.col2 .h50}
![Bild 1](gruen.png){alt="Beschreibung 1" }

![Bild 2](blau.png){alt="Beschreibung 2" }
:::
```

Der “alte” Eintrag enthält die Beschreibung für den Screenreader. Bildschirmleser, der Sehende sieht nur die Bildunterschrift.

Die Leerzeile ist wichtig!

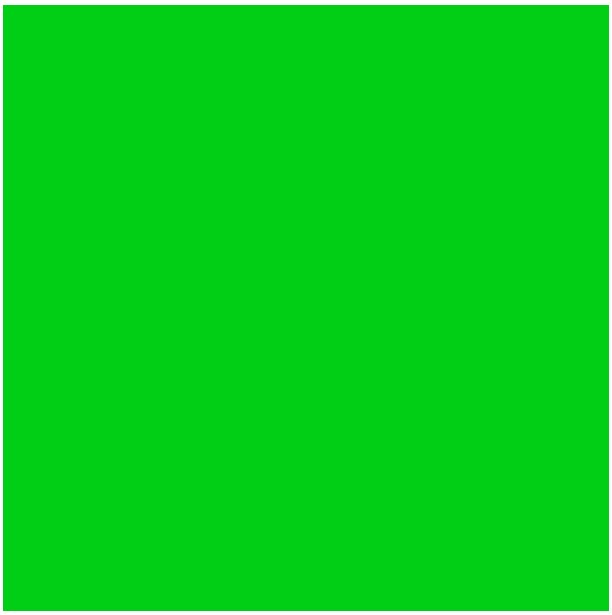


Bild 1

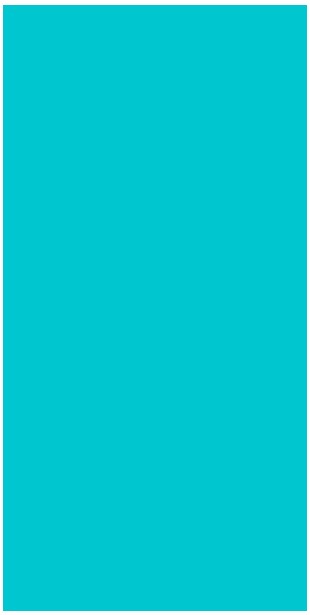


Bild 2

15) 2-spaltiges Layout mit einem Bild

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

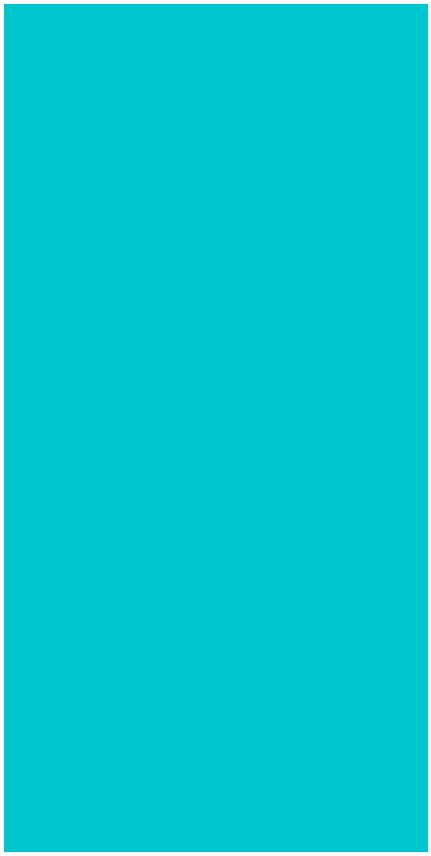


Bild 3

16) Verwendete Programme

sh / *gosh.exe*
: bash, ksh, zsh

sed / *sed-go.exe*
: Korrekturen, Entfernen störender Links, Kode für der Druckausgabe verbessern.

awk / *goawk.exe*
: verschiedene kleine Anpassungen

data-uri / *data-uri.exe*, ein eigenes Binary.
: Ziel ist es, die Bilder als Data-URI in die Präsentation einzubinden. Damit nicht mehrere Dateien übertragen/kopiert werden müssen.

data-uri.exe Wurde mit Pelles C unter Windows kompiliert.

Die ersten 3 **exe** Dateien stammen aus den entsprechenden Projekten auf *github.com*. Sie wurden kompiliert und zu Testzwecken auf einer Windows Installation verwendet.

Die benötigten Programme können in das Template-Verzeichnis kopiert werden. Dadurch wird sichergestellt, dass sie vom Skript gefunden werden.

17) Kompatibilität

- Anzeige:
  - Firefox.
  - Chromium basierten Browser (Chrome, Edge, …).
  - Safari (Fullscreen nicht auf iPhone und Ipad).
- Erzeugung:
  - Linux.
  - MacOS.
  - Windows, die zuvor genannten Programme müssen verfügbar sein.

Je nach Betriebssystem, Distribution und Benutzerpräferenzen können verschiedene Shells verfügbar sein. Das Skript build.sh kann von dash, zsh, ksh, bash und gosh (Windows) ausgeführt werden.

18) Links

- Pandoc (https://pandoc.org/)
- WeasyPrint (https://github.com/Kozea/WeasyPrint)
- Prince XML (https://www.princexml.com/)
- gosh (https://github.com/mvdan/sh)
- goawk (https://github.com/benhoyt/goawk)
- Sed-go (https://github.com/rwtodd/Go.Sed)
- Pelles C (http://www.smorgasbordet.com/pellesc/)