# Howto for TTS with Emacs-Reveal *

Jens Lechtenbörger

April 23, 2024

This presentation serves as example for a presentation that is generated with emacs reveal, where audio is generated via text to speech. Please check out the source file of this presentation for details. Prior knowledge of emacs reveal is required.

Personally, the author of this software does not like learning from videos as they exhibit limited navigation capabilities (no skim reading, limited search, no document structure, no hyperlinks). His students like videos, though.

Maintaining high-quality video and audio over years is a considerable challenge. To overcome this challenge, revealjs presentations can be run in a video mode where slides come with audio explanations and advance automatically. This presentation serves as example.

## 1 General thoughts

- All of this builds on emacs-reveal (Lechtenbörger 2019a, 2019b)
  - Check out its howto first
- Text-To-Speech (TTS) should read notes (`#+begin_notes ... #+end_notes`)
  - Controlled by option `reveal-with-tts`
    - Use customization for available speakers
  - Audio is played with the audio slideshow plugin for Reveal.js
- If slides with audio advance automatically, this is a video mode
  - Then, notes are required for every slides
  - Reveal.js "fragments" (animations) are still possible

This presentation does not aim to explain emacs reveal, which is a free and open source software bundle to create presentations as open educational resources based on the presentation framework reveal js.

This slide lists some general thoughts on tea tea ess.

The source code for this slide uses org properties to specify a break of ten seconds before advancing to the next slide after audio stops. This offers some reading time for the bullet points.

However, the audience does not receive any clue regarding the length of this pause. This is different with `break` tags used in the notes on the next slide.

*This PDF document is an inferior version of an OER HTML page; free/libre Org mode source repository.

## 1.1 Technical Idea

- Implement TTS as two-stage process
  - First, extract notes from presentation
    - Generate a text file for each note
      - Its name is a hash value of the contents
    - Generate one index file that stores names (and other information) for all text files
    - This happens during export/publication of Org files into reveal.js presentations
  - Second, run TTS software on index file to generate audio
    - Implemented in Docker image emacs-reveal/tts
    - Generated audio shares hash value of its text as part of its name, enabling caching of unchanged audio
- Use audio slideshow plugin to play audio

<break time="3s" /> Speech is generated from text in a process with two stages. First, usual presentation notes serve as text input. These notes may embed SSML `break` elements to specify a break with a given duration in seconds between sentences. See the source code of this slide for examples.

While processing the org source code to generate a presentation, each note is extracted into a text file (with some preprocessing). The name of such a text file is the hash value of its contents. Thus, changing contents lead to changing names. <break time="1s" />

In addition, another text file serves as index, collecting the names and positions of texts in a presentation. Besides, this index file also records configuration information, such as the speaker to be used. <break time="1s" /> Second, the index serves as input for the text to speech implementation, which is available as Docker image. Here, names of generated audio again embed the hash values of their input texts, enabling caching of unchanged audio.

In presentations, audio is played with the audio slideshow plugin. <break time="3s" />

## 1.2 Docker image `emacs-reveal/tts`

- Contains two free and open TTS implementations
  - SpeechBrain
  - Microsoft SpeechT5
- For size reasons, without GPU support
- Small wrapper package tts.py
  - Sample invocation shown in .gitlab-ci.yml of this presentation

This slide mentions some technical aspects of the text to speech approach. Please see for yourself if you are interested.

# 2 Slide with notes and fragments

These notes are supposed to be transformed to audio by tea tea ess and read by the audio plugin (if it is enabled). Org-re-reveal converts text to have each sentence on a single line, which is converted to audio by Docker image `tts` of emacs-reveal.

Note that hyphenated words and abbreviations are usually not pronounced correctly, e.g., CPUs. We might rewrite this to use multiple see pea use?

Notes can contain Org markup, such as hyperlinks, **bold**, *emphasis*, `code`, `verbatim`.
Such markup is removed for tea tea ess in org-re-reveal.
Lists can be used in notes as well:

1. This is a first item in a list.
2. Second item.

Note that numbers are currently skipped in text to speech. Thus, the text of enumerations should clarify which point is currently read.

As we aim for text to speech, notes should consist of full sentences, including full stops, question marks etc. Warnings are shown upon export if the code detects this not to be the case.

Notes on this slide clarify some aspects of the text generated by org-re-reveal as basis for TTS.

Besides, for demonstration purposes, this slide contains fragments with separate notes:

- First appearing point, with notes

  Each fragment has its own notes. <break time="1s" /> These ones are meant for the first bullet point.

- Second appearing point

  Explanations continue with this second bullet point.

# 3 Slides with notes, no fragments

- First point
  - Notes only for overall slide
- Second point

These are usual or general notes for the entire slide.

For text to speech, to help the audience, it might be better to animate lists and explain them in separate notes for each bullet point.

## 3.1 Slide on second level of nesting

- Foo
- Bar

Slides can be nested. Audio file names by default follow the schema of dot-separated numbers for horizontal and vertical slides, including fragments.

### 3.1.1 Third level of nesting

My deepest level of nesting is used for this empty slide.

### 3.1.2 Third level of nesting, again

This slide serves as test case for generated audio names.

## 3.2 Another test case

This is another test case for generated audio names.

# 4 The End

Figure 1: The road ahead . . . (“Figure” under CC0 1.0; converted from Pixabay)

`https://gitlab.com/oer/`
The end is near. Or the beginning?

## 4.1 Final slide (silent)

- Here, the notes specify the names for the text and audio file as `another-slide-audio`
  - Thus, the audio plugin will not play this properly with the current configuration

The source code of this slide specifies the name for the text and audio files of the notes. Maybe this is useful for you, maybe not. In this presentation, that audio is not played.

## 4.2 Bibliography

Lechtenbörger, Jens. 2019a. “Emacs-reveal: A software bundle to create OER presentations.” *Journal of Open Source Education (Jose)* 2 (18). `https://doi.org/10.21105/jose.00050`.

———. 2019b. “Simplifying license attribution for OER with emacs-reveal.” In *17. Fachtagung Bildungstechnologien (DELFI 2019)*, edited by Niels Pinkwart and Johannes Konert, 205–16. Bonn: Gesellschaft für Informatik e.V. `https://doi.org/10.18420/delfi2019_280`.

My presentations usually contain a bibliography like this one. Did you notice the references on an earlier slide? Those are hyperlinks into this slide.

# License Information