# Manual for `present5`

Michael Schaefer

October 25, 2013

**Abstract**

`present5` is a tool to create presentations like you do using PowerPoint or LaTeX beamer. It is based on HTML5, CSS and JavaScript and is thus platform independent. It also provides LaTeX-to-MathML conversion which makes it a good choice even in a scientific environment. This document provides the reader with the knowledge to create and show presentations. It also aims to enable him to extend the tool or to provide new themes.

# Contents

# 1. About `present5`

Some time ago I had a discussion with a few colleges where we came to the conclusion that modern web technologies offer a nice way to completely platform independent presentations. In contrast to e.g. Microsoft's PowerPoint or PDFs generated with LaTeX it has (among others) the advantage that search engines like Google can directly access the content of the slides. Obviously we were not the first ones with this idea. There is a bunch of different frameworks for this purpose. However, I decided to start working on a new one. One reason for this was to provide a framework that needs as little additional source code around the actual content as possible, but the more important one can be put into the words "for educational purpose".

## 1.1. Features

`present5` combines modern web technologies to a platform independent and standard compliant presentation framework. In detail, we use

- **HTML5** for the content,
- **CSS** for the layout and
- **JavaScript** for the navigation and other necessary program logic

In what follows the term *core features* is used for building blocks of the framework that are realized in JavaScript and therefore available independent of the chosen layout. Core features include

- **sequential navigation** by keyboard, mouse buttons and mouse wheel
- **direct navigation** via user prompt and a special thumbnail view
- **presenter mode** with slide preview, timer and notes on a secondary monitor
- **overlays** for step-by-step slide opening
- **automatic line number** of `<pre>` environments (useful for source code)

As a specialty for people working in science, you can directly put LaTeX code into your presentation which is converted into MathML by a plug-in.

## 1.2. System requirements

The aim of `present5` is to provide a presentation framework that is standard compliant and platform independent. So basically, your presentation should be able to work properly on any computer system that ships with a standard compliant web browser. This is e.g. true for the latest Firefox and Chrome releases. Unfortunately there are some drawbacks if you want to use MathML. To my knowledge, currently (meaning October 25, 2013) Firefox is the only browser with full MathML support.[1]

At last there is of course some requirement on the user itself. In order to make your slides with `present5` you should know at least some basics about HTML5 and the interplay with CSS (e.g. what a tag's `class="..."` attribute does).

## 1.3. Structure of the manual

The rest of the manual will address three different kind of people. *User* are those who want to use `present5` to present slides that were already created by someone else, while *creators* wish to build a new presentation and *developers* aim to extend the framework itself or to provide new themes for it. There will be a separate section for each of them.

[1]Chrome 24 introduced MathML support, but it was deactivated again in Version 25, so Firefox seems to stay the browser of choice.

# 2. User manual

In our terms, a *user* is someone who just wants to give a presentation that is already created. This section will tell you all about the features of `present5`.

## 2.1. Sequential navigation

The most basic feature is sequential navigation and it is achieved as follows:

**forward:** to go to the next slide or overlay, use

- right arrow and spacebar on keyboard
- left click on mouse
- mouse wheel up

**backward:** to go back to the last slide or overlay, use

- left arrow and backspace on keyboard
- right click on mouse
- mouse wheel down

Note that mouse navigation has to be enabled by the presentation, so it might not be available.

## 2.2. Direct navigation

As a contrast, there are ways to navigate directly to a certain slide. The most easy one is to press g on keyboard. This opens a user promt where you can enter the slide number you want to go to. You always jump to the first overlay of that slide.

A more elaborate feature is the *thumbnail view*. It can be open by pressing o on keyboard and shows you a grid of small thumbnails for the slides. You can navigate through the thumbnails using the arrow keys or by moving the mouse cursor. Once you have chosen the proper thumbnail, just press enter or left-click it to jump to the first overlay of that slide. As in the previous section, mouse actions are only possible if enabled by the presentation.

## 2.3. Dual mode

The dual mode is useful in two-monitor environment. It is used to show the plain presentation on one monitor, while the other one is used give the presenter additional information like a clock, custom notes and a preview of the following slide. The most common setup would be to show the presentation on a projector while having the additional information on your laptop display.

The dual mode is started by pressing d on keyboard. This opens another window also showing the presentation, the *secondary presentation*. This presentation can be moved to the secondary display as it is remote controlled by the first one, the *primary presentation*. Once activated, you have the following actions at hand:

**freeze mode:** press f on keyboard to toggle this mode. As long as on freeze mode, navigation only changes the primary presentation. The secondary presentation jumps to the right slide immediately after you deactivate freeze mode.

**start/stop timer:** `present5` has built-in timer functionality. Press s on keyboard to start or stop the timer, depending on if it is running or not. The timer is a countdown, if the presentation provides a variable called `presentationTime`, cf. appendix A. Otherwise it acts as stopwatch.

**reset timer:** If started accidentally, the timer can be reset by pressing r on keyboard. Time is reset according to `presentationTime`, if this value is set, otherwise it start at 0.

The dual mode is closed by pressing d again. This also closes the windows with the secondary presentation.

### 2.4. Additional actions

If for some reason you need to refresh the presentation, just press F5. Note that refreshing in dual mode in not allowed since this would break the link between primary and secondary presentation. You have to deactivate it in advance.

If you change the window size, the presentation has to be adjusted. You can use F4 for that. In dual mode, both primary and secondary presentation are resized simultaneously.

## 3. Creator manual

By the term *creator* we mean people who are creating slides using `present5` "as is". These people will be mainly interested the rest of this chapter only. If you want to provide new layout themes or wish to extend the code basis you are considered a *developer*. In that case you should read section 3.1 and then proceed to section 4. If instead you already have a presentation and just want to present it, you are a *user* and should have a look into section 2 instead.

### 3.1. Directory structure

The `present5` distribution has the following basic directory tree:

→ `css`

→ `doc`

→ `js`

The `css` directory contains the available themes. Currently we provide only one theme. It is called `simple`. Each theme is a collection of CSS and necessary media files such as background images or font files.

The name of the `doc` directory speaks for itself. The manual you are just reading is located there.

The program logic is coded in JavaScript. The neccessary code files are located in `js` together with some third-party scripts (currently just the LaTeX-to-MathML mechanism).

### 3.2. HTML skeleton

In this section we assume that your presentation file is named `presentation.html` and placed in the root directory of the `present5` distribution. The following listing can serve as a template. Important parts will be discussed below, parts in `[...]` have to be changed by you.

Listing 1: HTML skeleton

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8">
  <title>[your title]</title>
  <link rel="stylesheet" type="text/css" href="[path␣to␣your␣theme]/theme.css">
  <script type="text/javascript" src="js/present5.js"></script>
  <script type="text/javascript" src="[path␣to␣your␣theme]/generateSlide.js"></script>
</head>

<body>
  [your content]
</body>

</html>
```

**Line 1** The DOCTYPE advises the browser to render the document as HTML file

**Line 5** In order to display special characters (especially in MathML) font encoding should be set to unicode.

**Line 7,9** Replace `[path to your theme]` with the relative path (i.e. starting with `css/`) of your favorite theme's folder . Have a look into the `css` directory for available themes.

The major work is to fill `[your content]` with the actual presentation. The next section will teach you the basics about this.

## 3.3. Creating the presentation's content

Inside the `<body>` tag you can do basically three things: Specify meta information for the presentation, configure the framework and provide content for slides.

### 3.3.1. Meta information

You start with `<header id="meta-data">`. Inside of this you include one or more tags of the form `<div data-type="[type]">[content]</div>`. For `[type]` you can choose between the following values:

- title
- subtitle
- authors
- date
- language

Most of this should be self-explanatory. Just a few words about language: Its value will be put into the `<body>` tag, i.e. we will end up with `<body data-language="...">`. Themes may use this value to localize e.g. background images for different languages. The value of `[content]` can be arbitray HTML.

### 3.3.2. Framework configuration

Some parameters of the framework can be set by the creator. Therefore, include a block that looks like this:

```
<header id="js-config">
  <div data-variable="key" data-value="value"></div>
</header>
```

Of course you can have multiple `<div>` tags to configure different variables. A complete list of all allowed key-value pairs can be found in appendix A.

### 3.3.3. Slides

Each slide consist of exactly one `<section class="[slideType]">[content]</section>` tag. Depending on `[slideType]`, your `[content]` looks different:

- title: a title page contains only auto-generated meta information, so leave `[content]` empty.
- subject: for a slide with a subject you need to replace `[content]` with exactly two tags: the first one for the subject itself and the second one for the content; they do not need any attributes. It is important to keep this order!
- no-subject: same as above, but without the `<div>` tag for the subject.

### 3.3.4. Summary

To sum up the last three sections, below we give examplary code that you can e.g. put as `[your content]` in listing 1 to get a complete presentation template.

Listing 2: Sample slides

```
<header id="meta-data">
  <div data-type="title">Introducing <b>present5</b></div>
  <div data-type="subtitle">
    An HTML5 conform presentation framework<br/>
    (press <div class="key">&rarr;</div> to continue)
  </div>
  <div data-type="authors">Michael Schaefer</div>
  <div data-type="date">23rd November 2012</div>
  <div data-type="language">en</div>
</header>

<header id="js-config">
  <div data-variable="allowCycling" data-value="false"></div>
  <div data-variable="allowNavigationByMouse" data-value="true"></div>
</header>

<section class="title"></section>

<section class="subject">
  <div>Subject of the slide</div>
  <div>Content</div>
</section>

<section class="no-subject">
  <div>Content</div>
</section>
```

The code starts with meta information for the presentation, followed by some framework configuration. After that, three slides are created using all three default types.

## 3.4. Special features

There are two additional features contained in `present5` that may make it easier for you to create your presentation:

**LaTeX support:** For scientific presentations that need mathematical formulas there is a specialty: You can provide LaTeX code between `\(...\)` for an inline formula or `\[...\]` for an offset formula. `present5` uses MathJax[2] to convert it to MathML. For details we refer to the next section.

**Overlays:** Sometimes it is useful to uncover a slide step by step. To that end, any tag you use inside the slide's content area can be equipped with an attribute `data-overlay=#`, where `#` has to be replaced with a positive natural number.

## 3.5. Conversion and export

This section is dedicated to the task of transforming a presentation into a self-contained HTML document that is ready for publication. This process is pretty easy given you have a Python interpreter running on your system. It consists of the following two steps:

**conversion:** Press c on keyboard. This triggers the conversion process, which replaces the LaTeX code with MathML using MathJax and allows you to save the processed HTML code

**export:** Once you stored the converted file on disk, you can use the Python script `exporter.py` in `present5`'s root directory to create a self-contained presentation. Self-contained means, that the resulting file includes literally everything (including fonts and media files). This single file is

[2]http://www.mathjax.org

anything you need in order to show your presentation. To see the exporter's documentation just run it without further command line arguments.

## 4. Developer manual

Yet to be done

## A. Framework configuration

This section provides a list of variable-value pairs for the framework configuration, cf. section 3.3.2.

`allowCycling` (default: false)

> `true`: last slide is followed by first one again, same in the opposite direction
>
> `false`: presentation stops with first/last slide

`allowMouseInThumbnailView` (default: false)

> `true`: mouse is allowed for selecting slides in thumbnail view
>
> `false`: just keyboard navigation

`allowNavigationByMouse` (default: false)

> `true`: additional navigation via mouse buttons and wheel
>
> `false`: just keyboard navigation

`aspectRatio` (default: 16:9)
> Select the aspect ratio for the presentation. In general, any value can be specified in the format "a:b".

`presentationTime` (default: unset)
> Time for the presentation in minutes (as integer). If set, the timer in dual mode (cf. 2.3) will start as countdown clock with this value.

## B. `simple` theme documentation

This section explains all features of the `simple` theme that is supplied by `present5`.

### B.1. Parts

#### B.1.1. Slides

One slide is represented by a `<section>` tag. There are three different kind of slides, depending on the `class` attribute's value:

- `title`: An auto-generated title slide. Content should be left empty.
- `subject`: Slide with headline. Expects an `<h1>` tag for the headline followed by a `<div>` for the content. Keep this order!
- `no-subject`: Slide without headline. Expects one `<div>` tag for the content.

#### B.1.2. Blocks

In order to highlight certain parts of a slide you can use pre-formatted blocks. In general, a block is nothing more than a `<div class="block">`. There are three different types of blocks which differ in the color of background and border: standard, alert and example. The two latter ones can be specified by adding `alert` resp. `example` as additional argument in the class attribute of the surrounding `<div>`, e.g. `<div class="block alert">`.

The block is supposed to be filled with an `<h1>` tag for the headline, followed by another tag for the content. The content tag is not limited to `<div>`, it can be anything you like. You must keep this order to make it work.

### B.1.3. Figures

To include content with a description below it – like images or tables – `simple` provides a pre-configured `<figure>` tag which allows for providing both content and description.

For the content you are allowed to include as many arbitrary tags as you like. By default, the content will be centered and has a small margin to the description.

To include a description, add a `<figcaption>` tag right below the content. The description will automatically be preceeded by the string “Figure: ”, unless you have a `<table>` in the figure’s content. In that case, the preceeding string will be “Table: ”. Numbering is not supported.

### B.1.4. Margins

The theme is configured in such a way that margins between standard elements like blocks, lists and figures *should* look alright by default. Of course, this will not always work. That’s why we introduced some means to give additional margins by hand. You can use the `class="..."` attribute with the following values: `smallSkip` (0.5em margin), `mediumSkip` (1em) or `bigSkip` (2em).

By default, this produces a margin in all four directions (left, right, top, bottom). If you want to specify the direction of the margin you can add `bottom`, `left`, `right` or `top`.[3] Combinations will not work.

### B.1.5. Tables

`simple` provides pre-formatted tables. By default, tables are horizontally centered and have the following additional properties:

- no outer border
- small transparent spacing between cells
- the rows’ background color alternates between two tones of grey

## B.2. Overview

`<body>`
: can be equipped with an attribute `data-background="..."`. Possible values:
  - `wwu`: Show logos of University of Münster and Applied Mathematics Münster at the bottom of the presentation
  - `wwu_cenos`: Show logos as above an additionally the logo of the Center for Nonlinear Science (CeNoS)

  If omitted, the background is just white.

`<section>`
: **class attributes**: title, subject, no-subject (cf. B.1.1)

`<*>`
: **class attributes**:
  - **alignment**: center, left, right
  - **margins**: smallSkip, mediumSkip, bigSkip (cf. B.1.4)
  - **margin directions**: bottom, left, right, top (cf. B.1.4)

[3]Due to the nature of CSS it is important to write the direction *after* the skip.

`<div class="block">` (cf. B.1.2)
: **additional class attributes**: alert, example
: **supported children**[4]: `<h1>` for title, `<*>` for content. Keep this order!

`<figure>` (cf. B.1.3)
: **area of use**: wrap e.g. images, graphs or tables inside of it
: **supported children**: `<figcaption>` as last tag

`<ol>`
: **specialty**: supports automated numeric numbering of to three layers (e.g. 1.2.4)

`<pre>`
: **specialty**: automated line numbering for enclosed content

`<tr>`
: **specialty**: background color of table rows alternates

[4]i.e. tags that get a special default layout if they follow the specified parent tag