Modern Beamer Presentations with the **METROPOLIS** package

Matthias Vogelgesang
matthias.vogelgesang@gmail.com

v1.2 — 2017/01/23

Contents

1 Introduction

Beamer is an awesome way to make presentations with LaTeX, but its theme selection is surprisingly sparse. The stock themes share an aesthetic that can be a little cluttered, while the few distinctive custom themes available are often specialized for a particular corporate or institutional brand.

The goal of **METROPOLIS** is to provide a simple, modern Beamer theme suitable for anyone to use. It tries to minimize noise and maximize space for content; the only visual flourish it offers is an (optional) progress bar added to each slide or to the section slides.

By default, **METROPOLIS** uses Fira Sans, a gorgeous typeface commissioned by Mozilla and designed by Carrois. For best results, you will need the Fira typeface installed and use XƎLaTeX to typeset your slides. However, **METROPOLIS** can also be used with other typefaces and LaTeX build systems.

METROPOLIS's codebase is maintained on GitHub. If you have issues, find mistakes in the manual or want to help make the theme even better, please get in touch there. The full list of contributors already contains over a dozen names!

2 Getting Started

2.1 Installing from CTAN

For most users, we recommend installing **METROPOLIS** from CTAN. If you keep your TEX distribution up-to-date, chances are good that **METROPOLIS** is already installed. If it is not, you need to update your packages. If your distribution is TEX Live (or MacTEX on OS X), the following command updates all packages.

```
tlmgr update --all
```

If this results in an error, you may need to run it with administrative privileges:

```
sudo tlmgr update --all
```

MacTEX on OS X also provides a graphical interface for `tlmgr` called TEX Live Utility.

For any other distribution please refer to its documentation on how to update your packages.

To get the most out of the theme you should also install the `Fira` fonts. However, this is not mandatory; **METROPOLIS** also works with the standard fonts.

2.2 Installing from GitHub

If you want to use the cutting-edge development version of **METROPOLIS**, you can install it manually. Like any LATEX package, this involves four easy steps:

Download the source with a `git clone` of the **METROPOLIS** repository or as a zip archive of the latest development version.

Compile the style files by running `make sty` inside the downloaded directory. (Or run LATEX directly on `source/metropolistheme.ins`.)

Move the resulting `*.sty` files to the folder containing your presentation. To use **METROPOLIS** with many presentations, run `make install` or move the `*.sty` files to a folder in your TEX path instead.

Use the theme for your presentation by declaring `\usetheme{metropolis}` in the preamble of your Beamer document.

METROPOLIS uses the Make build system to offer the following installation options for advanced users:

`make sty` builds the theme style files.

`make doc` builds this documentation manual.

`make demo` builds a demo presentation to test the features of **METROPOLIS**.

`make all` builds the theme and manual.

`make clean` removes the files generated by `make all`.

`make install` installs the theme into your local texmf folder.

`make uninstall` removes the theme from your local texmf folder.

2.3 A Minimal Example

The following code shows a minimal example of a Beamer presentation using **METROPOLIS**.

```
\documentclass{beamer}
\usetheme{metropolis}       % Use metropolis theme
\title{A minimal example}
\date{\today}
\author{Matthias Vogelgesang}
\institute{Centre for Modern Beamer Themes}
\begin{document}
  \maketitle
  \section{First Section}
  \begin{frame}{First Frame}
    Hello, world!
  \end{frame}
\end{document}
```

2.4 Dependencies

METROPOLIS depends on the `beamer` class and the following standard packages:

- `tikz`
- `pgfopts`
- `etoolbox`
- `calc`
- `ifxetex`
- `ifluatex`

For best results, we recommend installing the fonts `Fira Sans` and `Fira Mono` and compiling with **METROPOLIS** using XeLaTeX or LuaTeX. These are optional dependencies; **METROPOLIS** is compatible with (e.g.) pdfLaTeX and will fall back to standard fonts if `Fira Sans` or `Fira Mono` is not installed.

The packaged name of `Fira Sans` is `Fira Sans OT` in some Linux distributions; this case is automatically handled by **METROPOLIS**.

2.5 Pandoc

To use this theme with Pandoc-based presentations, you can run the following command

```
$ pandoc -t beamer --latex-engine=xelatex -V theme:
   metropolis -o output.pdf input.md
```

3 Customization

3.1 Package options

The theme provides a number of options, which can be set using a key=value interface. The primary way to set options is to provide a comma-separated list of option-value pairs when loading **METROPOLIS** in the preamble:

```
\usetheme[option1=value1, option2=value2, ...]{metropolis}
```

Options can be changed at any time — even mid-presentation! — with the `\metroset` macro.

```
\metroset{option1=newvalue1, option2=newvalue2, ...}
```

The list of options is structured as shown in the following example.

`option key` *list of possible values* default

A short description of the option.

3.1.1 Main theme

`titleformat` *regular, smallcaps, allsmallcaps, allcaps* regular

Changes the format of titles, subtitles, section titles, frame titles, and the text on "standout" frames. The available options produce Regular, SmallCaps, allsmallcaps, or ALLCAPS titles. Please refer to Section 6.1 for known issues with these options.

`titleformat plain` *regular, smallcaps, allsmallcaps, allcaps* regular

Changes the format of "standout" frames (see `titleformat`, above).

3.1.2 Inner theme

`sectionpage` *none, simple, progressbar* progressbar

Adds a slide at the start of each section (`simple`) with an optional thin progress bar below the section title (`progressbar`). The `none` option disables the section page.

`subsectionpage` *none, simple, progressbar* none

Optionally adds a slide at the start of each subsection. If enabled with the `simple` or `progressbar` options, the style of the `section page` will be updated to match the style of the `subsection page`. Note that section slides and subsection slides can appear consecutively if both are enabled; you may want to use this option together with `sectionpage=none` depending on the section structure of your presentation.

3.1.3 Outer theme

`numbering` *none, counter, fraction* . . . . counter

Controls whether the frame number at the bottom right of each slide is omitted (`none`), shown (`counter`) or displayed as a fraction of the total number of frames (`fraction`).

`progressbar` *none, head, frametitle, foot* . . . . none

Optionally adds a progress bar to the top of each frame (`head`), the bottom of each frame (`foot`), or directly below each frame title (`frametitle`).

3.1.4 Color theme

`block` *transparent, fill* . . . . transparent

Optionally adds a light grey background to block environments like `theorem` and `example`.

`background` *dark, light* . . . . light

Provides the option to have a dark background and light foreground instead of the reverse.

3.1.5 Font theme

`titleformat title`
`titleformat subtitle`
`titleformat section`
`titleformat frame`

regular, smallcaps, allsmallcaps, allcaps . . . . regular

Individually controls the format of titles, subtitles, section titles, and frame titles (see `titleformat`, above).

3.2 Color Customization

The included METROPOLIS color theme is used by default, but its colors can be easily changed to suit your tastes. All of the theme's styles are defined in terms of three beamer colors:

- `normal text` (dark fg, light bg)

- `alerted text` (colored fg, should be visible against dark or light)
- `example text` (colored fg, should be visible against dark or light)

An easy way to customize the theme is to redefine these colors using

```
\setbeamercolor{ ... }{ fg= ... , bg= ... }
```

in your preamble. For greater customization, you can redefine any of the other stock beamer colors. In addition to the stock colors the theme defines a number of METROPOLIS specific colors, which can also be redefined to your liking.

```
\setbeamercolor{progress bar}{ ... }
\setbeamercolor{title separator}{ ... }
\setbeamercolor{progress bar in head/foot}{ ... }
\setbeamercolor{progress bar in section page}{ ... }
```

3.3 Font Customization

The default font for METROPOLIS is `Fira`. This can be easily changed using the standard font selection commands of the fontspec package. So if you prefer, for example, the `Ubuntu` font family, just add the following two commands after loading the METROPOLIS theme.

```
\setsansfont{Ubuntu}
\setmonofont{Ubuntu Mono}
```

If you are expecting to present in a large room or with an underpowered projector, you may want to change the font to a heavier weight of Fira to maximize readability.

```
\setsansfont[BoldFont={Fira Sans SemiBold}]{Fira Sans Book}
```

3.3.1 Old style figures

The regular fontspec mechanism for changing glyph appearance applies also to this theme. If you want to have old style figures in the text but regular lined figures for math, you could add the following to your preamble:

```
\usefonttheme{professionalfonts}   % required for mathspec
\usepackage{mathspec}
\setsansfont[BoldFont={Fira Sans},
             Numbers={OldStyle}]{Fira Sans Light}
\setmathsfont(Digits)[Numbers={Lining, Proportional}]{Fira
   Sans Light}
```

3.4 Commands

3.4.1 Standout frames

The METROPOLIS inner theme offers a custom frame format with large, centered text and an inverted background — perfect for focusing attention on single sentence or image. To use it, add the key `standout` to the frame:

```
\begin{frame}[standout]
    Thank you!
\end{frame}
```

4 `pgfplots` integration

METROPOLIS comes with a set of pre-defined pgfplots styles and a color theme based on Paul Tol's color scheme.

4.1 Styles

Pass the following style keys to the axis environment to get the appropriate effect:

`mlineplot`	Plot regular line charts with reduced axis frames, less intrusive legend and subdued grid.
`mbarplot`	Plot vertical bar charts in a similar way as `mlineplot` but reduce grid usage.

`horizontal mbarplot`	Plot horizontal bar charts.
`disable thousands separator`	Helper style to remove thousands separator.

4.2 Paul Tol colors

A good presentation uses colors that are distinct from each other as much as possible as well as from black and white, can be discerned item under different lighting and display environments and by color-blind viewers, while matching well together.

In a technical note for SRON, Paul Tol proposed a palette of colors satisfying these constraints. The sub-package `pgfplotsthemetol` defines palettes for `pgfplots` charts based on Tol's work.

5 Tips & Tricks

5.1 Backup Slides

Speakers will often include extra slides at the end of their presentation to refer to during audience questions. One easy way to do this is to include the `appendixnumberbeamer` package in your preamble and call `\appendix` before your backup slides.

METROPOLIS will automatically turn off slide numbering and progress bars for slides in the appendix.

6 Known Issues

6.1 Title formats

Be aware that not every font supports small caps, so the `smallcaps` or `allsmallcaps` options may not work if you use a font other than `Fira Sans`. In particular, the Computer Modern sans-serif typeface, which is used when METROPOLIS is compiled with pdfLaTeX, does not have a small-caps variant.

The title format options `allsmallcaps` and `allcaps` are quite nice from an aesthetic point of view, but their use of `\MakeLowercase` and `\MakeUppercase` can cause unexpected problems. For example:

- Some commands, like `\\`, do not work inside `\MakeLowercase` and `\MakeUppercase`. (See #125)
- Only alphabetic characters are affected by `\MakeLowercase`, so numerals and punctuation remain at full height. This can spoil some of the aesthetic benefits of `allsmallcaps`. (See #33)
- `\MakeLowercase` and `\MakeUppercase` apply to math mode and `\scshape` does not. This can easily introduce mathematical errors that are hard to catch.
- It is impossible to typeset symbols which are encoded as uppercase letters in a different font. In particular, `\mathbb` and `\mathcal` letters will be replaced by other math glyphs. (See #153)

The `allsmallcaps` and `allcaps` options are safe to use if your titles contain only alphabetic characters and do not require the expansion of any macros.

6.2 Interactions with other color themes

METROPOLIS can be used along with any other Beamer color theme, such as `crane` or `seahorse`. If you wish to do this, it is usually best to include the METROPOLIS subpackages individually so the METROPOLIS color theme is never loaded. This will prevent conflicts between the METROPOLIS color theme and your preferred theme.

For example, overriding the color theme as follows may not work as expected because `\usetheme{metropolis}` loads the METROPOLIS color theme, which defines a relationship between the frametitle background and the primary palette of the theme. Since `seahorse` assumes a different relationship between its palettes, the result is a grey, rather than periwinkle, frametitle background.

```
\usetheme{metropolis}
\usecolortheme{seahorse}
```

The correct colors are chosen if the METROPOLIS outer, inner, and font themes are loaded seperately:

```
\useoutertheme{metropolis}
\useinnertheme{metropolis}
\usefonttheme{metropolis}
\usecolortheme{seahorse}   % or your preferred color theme
```

Please note that **METROPOLIS** may not use all the colors defined in your favourite Beamer color theme. In particular, **METROPOLIS** does not set a background color for the title; this will cause issues when using color themes like `whale` which set a white foreground for the title.

6.3 Notes on second screen

If you use the `[show notes on second screen]` option built in to Beamer and compile with XeLaTeX, text on slides following the first section slide may be rendered in white instead of the regular colour. This is due to a bug in Beamer or XeLaTeX itself. You can work around it either by compiling with LuaTeX or by adding the following code to your preamble to reset the text color on each slide.

```
\makeatletter
\def\beamer@framenotesbegin{% at beginning of slide
     \usebeamercolor[fg]{normal text}
      \gdef\beamer@noteitems{}%
      \gdef\beamer@notes{}%
}
\makeatother
```

6.4 Standout frames with labels

Because the `standout` frame option creates a group to restrict the colour change to a single slide, labels defined after calling `standout` will stay local to the group. In other words, the following may result in a “label undefined” error.

```
\begin{frame}[standout, label=conclusion]{Conclusion}
  Awesome slide
```

```
\end{frame}
```

To fix this problem, change the order of the keys in the frame.

```
\begin{frame}[label=conclusion, standout]{Conclusion}
    Awesome slide
\end{frame}
```

This error can be unwittingly triggered if you export your slides from Emacs Org mode, which automatically adds labels after frame options. Alex Branham offers the following solution for Org mode users, using `org-set-property`.

```
* Start of a frame
   :PROPERTIES:
   :BEAMER_opt: label=conclusion,standout
   :END:
```

6.5 Standout frames with Pandoc

With Pandoc versions prior 1.17.2 it was not possible to create standout frames because Pandoc only supported a specific list of frame attributes thus ignoring additional attributes such as `{.standout}`.

7 License

METROPOLIS is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License. This means that if you change the theme and re-distribute it, you must retain the copyright notice header and license it under the same CC-BY-SA license. This does not affect any presentations that you create with the theme.

8 Implementation

8.1 METROPOLIS parent theme

The primary job of this package is to load the component sub-packages of the METROPOLIS theme and route the theme options accordingly. It also provides some custom commands and environments for the user.

8.1.1 Package dependencies

```
\RequirePackage{etoolbox}
\RequirePackage{pgfopts}
```

8.1.2 Options

Most options are passed off to the component sub-packages.

```
\pgfkeys{/metropolis/.cd,
  .search also={
    /metropolis/inner,
    /metropolis/outer,
    /metropolis/color,
    /metropolis/font,
  }
}
```

`titleformat plain` Controls the formatting of the text on standout “plain” frames.

```
\pgfkeys{
  /metropolis/titleformat plain/.cd,
    .is choice,
    regular/.code={%
      \let\metropolis@plaintitleformat\@empty%
      \setbeamerfont{standout}{shape=\normalfont}%
    },
    smallcaps/.code={%
      \let\metropolis@plaintitleformat\@empty%
      \setbeamerfont{standout}{shape=\scshape}%
    },
  allsmallcaps/.code={%
   \let\metropolis@plaintitleformat\MakeLowercase%
   \setbeamerfont{standout}{shape=\scshape}%
   \PackageWarning{beamerthememetropolis}{%
    Be aware that titleformat plain=allsmallcaps can lead to problems%
   }
  },
  allcaps/.code={%
   \let\metropolis@plaintitleformat\MakeUppercase%
   \setbeamerfont{standout}{shape=\normalfont}%
   \PackageWarning{beamerthememetropolis}{%
    Be aware that titleformat plain=allcaps can lead to problems%
   }
  },
}
```

`titleformat` Sets a standard format for titles, subtitles, section titles, frame titles, and the text on standout “plain” frames.

```
\pgfkeys{
 /metropolis/titleformat/.code=\pgfkeysalso{
   font/titleformat title=#1,
   font/titleformat subtitle=#1,
   font/titleformat section=#1,
   font/titleformat frame=#1,
   titleformat plain=#1,
  }
}
```

For backwards compatibility with earlier betas of the theme, we implement deprecated option names as aliases to the corresponding `key=value` options.

```
\pgfkeys{/metropolis/.cd,
 usetitleprogressbar/.code=\pgfkeysalso{outer/progressbar=frametitle},
 noslidenumbers/.code=\pgfkeysalso{outer/numbering=none},
 usetotalslideindicator/.code=\pgfkeysalso{outer/numbering=fraction},
 nosectionslide/.code=\pgfkeysalso{inner/sectionpage=none},
 darkcolors/.code=\pgfkeysalso{color/background=dark},
 blockbg/.code=\pgfkeysalso{color/block=fill, inner/block=fill},
}
```

Set default values for options.

```
\newcommand{\metropolis@setdefaults}{
  \pgfkeys{/metropolis/.cd,
    titleformat plain=regular,
  }
}
```

8.1.3 Component sub-packages

Having processed the options, we can now load the component sub-packages of the theme.

```
\useinnertheme{metropolis}
\useoutertheme{metropolis}
\usecolortheme{metropolis}
\usefonttheme{metropolis}
```

The `tol` theme for `pgfplots` is only loaded if `pgfplots` is used.

```
\AtEndPreamble{%
  \@ifpackageloaded{pgfplots}{%
    \RequirePackage{pgfplotsthemetol}
  }{}
}
```

8.1.4 Custom commands

The parent theme defines custom commands as their proper usage may depend on multiple sub-packages.

`\metroset` Allows the user to change options midway through a presentation.

```
\newcommand{\metroset}[1]{\pgfkeys{/metropolis/.cd,#1}}
```

`\plain` Creates a plain frame with dark background, suitable for displaying images or a few words. The format of the text can be set with the `titleformat plain` option.

```
\def\metropolis@plaintitleformat#1{#1}
\newcommand{\plain}[2][]{%
  \PackageWarning{beamerthememetropolis}{%
    The syntax ‘\plain’ may be deprecated in a future version of Metropolis.
    Please use a frame with [standout] instead.
  }
  \begin{frame}[standout]{#1}
    \metropolis@plaintitleformat{#2}
  \end{frame}
}
```

`\mreducelistspacing`

```
\newcommand{\mreducelistspacing}{\vspace{-\topsep}}
```

8.1.5 Process package options

```
\metropolis@setdefaults
\ProcessPgfOptions{/metropolis}
```

8.2 METROPOLIS inner theme

A `beamer` inner theme dictates the style of the frame elements traditionally set in the “body” of each slide. These include:

- title, part, and section pages;
- itemize, enumerate, and description environments;
- block environments including theorems and proofs;
- figures and tables; and
- footnotes and plain text.

8.2.1 Package dependencies

```
\RequirePackage{etoolbox}
\RequirePackage{keyval}
\RequirePackage{calc}
\RequirePackage{pgfopts}
\RequirePackage{tikz}
```

8.2.2 Options

`sectionpage` Optionally add a slide marking the beginning of each section.

```
\pgfkeys{
  /metropolis/inner/sectionpage/.cd,
    .is choice,
    none/.code=\metropolis@disablesectionpage,
    simple/.code={\metropolis@enablesectionpage
                  \setbeamertemplate{section page}[simple]},
    progressbar/.code={\metropolis@enablesectionpage
                       \setbeamertemplate{section page}[progressbar]},
}
```

`subsectionpage` Optionally add a slide marking the beginning of each subsection.

```
\pgfkeys{
  /metropolis/inner/subsectionpage/.cd,
    .is choice,
    none/.code=\metropolis@disablesubsectionpage,
    simple/.code={\metropolis@enablesubsectionpage
                  \setbeamertemplate{section page}[simple]},
    progressbar/.code={\metropolis@enablesubsectionpage
                       \setbeamertemplate{section page}[progressbar]},
}
```

`etropolis@inner@setdefaults` Set default values for inner theme options.

```
\newcommand{\metropolis@inner@setdefaults}{
  \pgfkeys{/metropolis/inner/.cd,
    sectionpage=progressbar,
    subsectionpage=none
  }
}
```

8.2.3 Title page

`title page` Template for the title page. Each element is only typeset if it is defined by the user. If `\subtitle` is empty, for example, it won't leave a blank space on the title slide.

```
\setbeamertemplate{title page}{
  \begin{minipage}[b][\paperheight]{\textwidth}
    \ifx\inserttitlegraphic\@empty\else\usebeamertemplate*{title graphic}\fi
    \vfill%
    \ifx\inserttitle\@empty\else\usebeamertemplate*{title}\fi
    \ifx\insertsubtitle\@empty\else\usebeamertemplate*{subtitle}\fi
    \usebeamertemplate*{title separator}
```

Beamer's definition of `\insertauthor` is always nonempty, so we have to test another macro initialized by `\author{...}` to see if the user has defined an author. This solution was suggested by Enrico Gregorio in an answer to this Stack Exchange question.

```
    \ifx\beamer@shortauthor\@empty\else\usebeamertemplate*{author}\fi
    \ifx\insertdate\@empty\else\usebeamertemplate*{date}\fi
    \ifx\insertinstitute\@empty\else\usebeamertemplate*{institute}\fi
    \vfill
    \vspace*{1mm}
  \end{minipage}
}
```

Normal people should use `\maketitle` or `\titlepage` instead of using the `title page` beamer template directly. Beamer already defines these macros, but we patch them here to make the title page `[plain]` by default, remove `\@thanks`, and ensure the title frame number doesn't count.

`\maketitle`
`\titlepage`

Inserts the title frame, or causes the current frame to use the `title page` template.

```
\def\maketitle{%
  \ifbeamer@inframe
    \titlepage
  \else
    \frame[plain,noframenumbering]{\titlepage}
  \fi
}
\def\titlepage{%
  \usebeamertemplate{title page}
}
```

`title graphic` Set the title graphic in a zero-height box, so it doesn't change the position of other elements.

```
\setbeamertemplate{title graphic}{
  \vbox to 0pt {
    \vspace*{2em}
    \inserttitlegraphic%
  }%
  \nointerlineskip%
}
```

`title` Set the title on the title page.

```
\setbeamertemplate{title}{
  \raggedright%
  \linespread{1.0}%
  \inserttitle%
  \par%
  \vspace*{0.5em}
}
```

`subtitle` Set the subtitle on the title page.

```
\setbeamertemplate{subtitle}{
  \raggedright%
  \insertsubtitle%
  \par%
  \vspace*{0.5em}
}
```

`title separator` Template to set the title graphic in a zero-height box. (It won't change the position of other elements.)

```
\newlength{\metropolis@titleseparator@linewidth}
\setlength{\metropolis@titleseparator@linewidth}{0.4pt}
\setbeamertemplate{title separator}{
  \begin{tikzpicture}
    \fill[fg] (0,0) rectangle (\textwidth, \metropolis@titleseparator@linewidth);
  \end{tikzpicture}%
  \par%
}
```

`author` Set the author on the title page.

```
\setbeamertemplate{author}{
  \vspace*{2em}
  \insertauthor%
  \par%
  \vspace*{0.25em}
}
```

`date` Set the date on the title page.

```
\setbeamertemplate{date}{
  \insertdate%
  \par%
}
```

`institute` Set the institute on the title page.

```
\setbeamertemplate{institute}{
  \vspace*{3mm}
  \insertinstitute%
  \par%
}
```

8.2.4 Section page

`section page` Template for the section title slide at the beginning of each section.

```
\defbeamertemplate{section page}{simple}{
  \begin{center}
    \usebeamercolor[fg]{section title}
    \usebeamerfont{section title}
    \insertsectionhead\par
    \ifx\insertsubsectionhead\@empty\else
      \usebeamercolor[fg]{subsection title}
      \usebeamerfont{subsection title}
      \insertsubsectionhead
    \fi
  \end{center}
}
```

```
\defbeamertemplate{section page}{progressbar}{
  \centering
  \begin{minipage}{22em}
    \raggedright
    \usebeamercolor[fg]{section title}
    \usebeamerfont{section title}
    \insertsectionhead\\[-1ex]
    \usebeamertemplate*{progress bar in section page}
    \par
    \ifx\insertsubsectionhead\@empty\else%
      \usebeamercolor[fg]{subsection title}%
      \usebeamerfont{subsection title}%
      \insertsubsectionhead
    \fi
  \end{minipage}
  \par
  \vspace{\baselineskip}
}
\newcommand{\metropolis@disablesectionpage}{
  \AtBeginSection{
    % intentionally empty
  }
}
\newcommand{\metropolis@enablesectionpage}{
  \AtBeginSection{
    \ifbeamer@inframe
      \sectionpage
    \else
      \frame[plain,c,noframenumbering]{\sectionpage}
    \fi
  }
}
```

`subsection page` Template for the subsection title slide that can optionally be added to at the beginning of each subsection.

```
\setbeamertemplate{subsection page}{%
  \usebeamertemplate*{section page}
}
\newcommand{\metropolis@disablesubsectionpage}{
  \AtBeginSubsection{
    % intentionally empty
  }
}
\newcommand{\metropolis@enablesubsectionpage}{
  \AtBeginSubsection{
    \ifbeamer@inframe
      \subsectionpage
    \else
      \frame[plain,c,noframenumbering]{\subsectionpage}
    \fi
  }
}
```

`rogress bar in section page` Template for the progress bar displayed by default on the section page. This code is duplicated in large part in the outer theme's template `progress bar in head/foot`.

```
\newlength{\metropolis@progressonsectionpage}
\newlength{\metropolis@progressonsectionpage@linewidth}
\setlength{\metropolis@progressonsectionpage@linewidth}{0.4pt}
\setbeamertemplate{progress bar in section page}{
  \setlength{\metropolis@progressonsectionpage}{%
    \textwidth * \ratio{\insertframenumber pt}{\inserttotalframenumber pt}%
  }%
  \begin{tikzpicture}
    \fill[bg] (0,0) rectangle (\textwidth, \metropolis@progressonsectionpage@linewi
    \fill[fg] (0,0) rectangle (\metropolis@progressonsectionpage, \metropolis@progressonsectionpage@linewidth);
  \end{tikzpicture}%
}
```

The above code assumes that `\insertframenumber` is less than or equal to `\inserttotalframenumber`. However, this is not true on the first compile; in the absence of an `.aux` file, `\inserttotalframenumber` defaults to 1. This behaviour could cause fatal errors for long presentations, as `\metropolis@progressonsectionpage` would exceed TEX's maximum length (16383.99999pt, roughly 5.75 metres or 18.9 feet). To avoid this, we increase the default value for `\inserttotalframenumber`; presentations with over 4000 slides will still break on first compile, but users in that situation likely have deeper problems to solve.

```
\def\inserttotalframenumber{100}
```

8.2.5 Block environments

`block`
`block alerted`
`block example`

The three different block environments differ only in their colours. Rather than repeat the essentially the same template three times, we use the auxiliary macro `\metropolis@block` to define all three templates.

```
\newlength{\metropolis@blocksep}
\newlength{\metropolis@blockadjust}
\setlength{\metropolis@blocksep}{0.75ex}
\setlength{\metropolis@blockadjust}{0.25ex}
\providecommand{\metropolis@strut}{%
  \vphantom{ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz()}%
}
\newcommand{\metropolis@block}[1]{
  \par\vskip\medskipamount%
  \setlength{\parskip}{0pt}
```

If a background color is defined for the block title or body, we need to add a little bit of padding to the corresponding box. Ideally, this would be accomplished by setting `colsep=0.75ex`, which is intended to add “color separation space” only when the box has a colored background. Unfortunately, `colsep` also adds this separation if the background color is inherited, even if the inherited color is actually empty. (The technical reason for this boils down to the fact that the `\ifx` directive does not expand macros.)

To achieve the correct spacing for `alertblock`s and `exampleblock`s as well as for normal blocks, we have to begin the `beamercolorbox` differently based on whether `block title` has an empty background.

If the `block title` background is empty, or the user has explicitly removed the background from (e.g.) `block title alerted`, we just need to set a rightskip for a nice ragged-right block title.

```
  \ifbeamercolorempty[bg]{block title#1}{%
    \begin{beamercolorbox}[rightskip=0pt plus 4em]{block title#1}}{%
  \ifbeamercolorempty[bg]{block title}{%
```

```
    \begin{beamercolorbox}[rightskip=0pt plus 4em]{block title#1}%
  }%
%    \end{macrocode}
%
%    Otherwise, if the |block title| has a background, we set the padding based
%    on |\metropolis@blockskip|. However, we have to visually compensate for
%    the |\metropolis@strut| added to the block title (see below) by
%    subtracting |\metropolis@blockadjust| from the top and bottom padding.
%
%    \begin{macrocode}
  {%
    \begin{beamercolorbox}[
      sep=\dimexpr\metropolis@blocksep-\metropolis@blockadjust\relax,
      leftskip=\metropolis@blockadjust,
      rightskip=\dimexpr\metropolis@blockadjust plus 4em\relax
    ]{block title#1}%
  }}%
%    \end{macrocode}
%
%    We can now set the contents of the |block title|. The zero-width but
%    positive-height box |\metropolis@strut| ensures that the block title box
%    has a consistent height, even if it lacks punctuation, ascenders, or
%    descenders.
%
%    \begin{macrocode}
      \usebeamerfont*{block title#1}%
      \metropolis@strut%
      \insertblocktitle%
      \metropolis@strut%
  \end{beamercolorbox}%
%    \end{macrocode}
%
%    Next, we typeset the |block body|. This the code is similar to, but simpler
%    than, the |block title| code since we don’t need to adjust for any struts.
%
%    \begin{macrocode}
```

```
  \nointerlineskip%
  \ifbeamercolorempty[bg]{block body#1}{%
    \begin{beamercolorbox}[vmode]{block body#1}}{
  \ifbeamercolorempty[bg]{block body}{%
    \begin{beamercolorbox}[vmode]{block body#1}%
  }{%
    \begin{beamercolorbox}[sep=\metropolis@blocksep, vmode]{block body#1}%
    \vspace{-\metropolis@parskip}
  }}%
      \usebeamerfont{block body#1}%
      \setlength{\parskip}{\metropolis@parskip}%
}
```

This concludes the auxiliary macro `\metropolis@block`. Finally, we define the block beamer templates using this macro.

```
\setbeamertemplate{block begin}{\metropolis@block{}}
\setbeamertemplate{block alerted begin}{\metropolis@block{ alerted}}
\setbeamertemplate{block example begin}{\metropolis@block{ example}}
\setbeamertemplate{block end}{\end{beamercolorbox}\vspace*{0.2ex}}
\setbeamertemplate{block alerted end}{\end{beamercolorbox}\vspace*{0.2ex}}
\setbeamertemplate{block example end}{\end{beamercolorbox}\vspace*{0.2ex}}
```

8.2.6 Lists and floats

```
\setbeamertemplate{itemize items}{\textbullet}
\setbeamertemplate{caption label separator}{: }
\setbeamertemplate{caption}[numbered]
```

8.2.7 Footnotes

```
\setbeamertemplate{footnote}{%
  \parindent 0em\noindent%
  \raggedright
  \usebeamercolor{footnote}\hbox to 0.8em{\hfil\insertfootnotemark}\insertfootnotet
}
```

8.2.8 Text and spacing settings

```
\newlength{\metropolis@parskip}
```

```
\setlength{\metropolis@parskip}{0.5em}
\setlength{\parskip}{\metropolis@parskip}
\linespread{1.15}
```

By default, Beamer frames offer the `c` option to *almost* vertically center the text, but the placement is a little too high. To fix this, we redefine the `c` option to equalize `\beamer@frametopskip` and `\beamer@framebottomskip`. This solution was suggested by Enrico Gregorio in an answer to this Stack Exchange question.

```
\define@key{beamerframe}{c}[true]{% centered
  \beamer@frametopskip=0pt plus 1fill\relax%
  \beamer@framebottomskip=0pt plus 1fill\relax%
  \beamer@frametopskipautobreak=0pt plus .4\paperheight\relax%
  \beamer@framebottomskipautobreak=0pt plus .6\paperheight\relax%
  \def\beamer@initfirstlineunskip{}%
}
```

8.2.9 Standout frames

METROPOLIS offers a custom frame format with large, centered text and an inverted background. To use it, add the key `standout` to the frame: `\begin{frame}[standout] ... \`

`standout` Optional arguments to Beamer's frames are implemented using `\define@key` from the `keyval` package, which will execute code when the defined option is called. For the `standout` option, we begin a group, change the colors and fonts, and set a alignment.

```
\providebool{metropolis@standout}
\define@key{beamerframe}{standout}[true]{%
  \booltrue{metropolis@standout}
  \begingroup
    \setkeys{beamerframe}{c}
    \setkeys{beamerframe}{noframenumbering}
    \ifbeamercolorempty[bg]{palette primary}{
      \setbeamercolor{background canvas}{
        use=palette primary,
        bg=-palette primary.fg
      }
    }{
      \setbeamercolor{background canvas}{
        use=palette primary,
        bg=palette primary.bg
      }
    }
  \centering
  \usebeamercolor[fg]{palette primary}
  \usebeamerfont{standout}
}
```

Then we just have to close the group after the standout slide is finished in order to restore the colours and fonts for the rest of the presentation. Unfortunately, we cannot use or this (see http://tex.stackexchange.com/questions/226319/). Instead, we add the `\endgroup` to `\beamer@reseteecodes`, which is run exactly once at the end of each slide.

```
  \apptocmd{\beamer@reseteecodes}{%
    \ifbool{metropolis@standout}{
      \endgroup
      \boolfalse{metropolis@standout}
    }{}
  }{}{}
```

8.2.10 Process package options

```
\metropolis@inner@setdefaults
\ProcessPgfPackageOptions{/metropolis/inner}
```

8.3 METROPOLIS outer theme

A **beamer** outer theme dictates the style of the frame elements traditionally set outside the body of each slide: the head, footline, and frame title.

8.3.1 Package dependencies

```
\RequirePackage{etoolbox}
\RequirePackage{calc}
\RequirePackage{pgfopts}
```

8.3.2 Options

`numbering` Adds slide numbers to the bottom right of each slide.

```
\pgfkeys{
  /metropolis/outer/numbering/.cd,
    .is choice,
    none/.code=\setbeamertemplate{frame numbering}[none],
    counter/.code=\setbeamertemplate{frame numbering}[counter],
    fraction/.code=\setbeamertemplate{frame numbering}[fraction],
}
```

`progressbar` Adds a progress bar to the top, bottom, or frametitle of each slide.

```
\pgfkeys{
  /metropolis/outer/progressbar/.cd,
    .is choice,
    none/.code={%
      \setbeamertemplate{headline}[plain]
      \setbeamertemplate{frametitle}[plain]
      \setbeamertemplate{footline}[plain]
    },
    head/.code={\pgfkeys{/metropolis/outer/progressbar=none}
      \addtobeamertemplate{headline}{}{%
        \usebeamertemplate*{progress bar in head/foot}
      }
    },
    frametitle/.code={\pgfkeys{/metropolis/outer/progressbar=none}
      \addtobeamertemplate{frametitle}{}{%
        \usebeamertemplate*{progress bar in head/foot}
      }
    },
    foot/.code={\pgfkeys{/metropolis/outer/progressbar=none}
      \addtobeamertemplate{footline}{}{%
        \usebeamertemplate*{progress bar in head/foot}%
      }
    },
}
```

`etropolis@outer@setdefaults` Sets default values for outer theme options.

```
\newcommand{\metropolis@outer@setdefaults}{
  \pgfkeys{/metropolis/outer/.cd,
    numbering=counter,
    progressbar=none,
  }
}
```

8.3.3 Head and footline

All good `beamer` presentations should already remove the navigation symbols, but METROPOLIS removes them automatically (just in case).

```
\setbeamertemplate{navigation symbols}{}
```

`frame numbering`

Templates for the frame number. Can be omitted, shown or displayed as a fraction of the total frames.

```
\defbeamertemplate{frame footer}{none}{}
\defbeamertemplate{frame footer}{custom}[1]{ #1 }
```

```
\defbeamertemplate{frame numbering}{none}{}
\defbeamertemplate{frame numbering}{counter}{\insertframenumber}
\defbeamertemplate{frame numbering}{fraction}{
  \insertframenumber/\inserttotalframenumber
}
```

`headline`
`footline`

Templates for the head- and footline at the top and bottom of each frame.

```
\defbeamertemplate{headline}{plain}{}
\defbeamertemplate{footline}{plain}{%
  \begin{beamercolorbox}[wd=\textwidth, sep=3ex]{footline}%
    \usebeamerfont{page number in head/foot}%
    \usebeamertemplate*{frame footer}
    \hfill%
    \usebeamertemplate*{frame numbering}
  \end{beamercolorbox}%
}
```

8.3.4 Frametitle

`frametitle` Templates for the frame title, which is optionally underlined with a progress bar.

```
\newlength{\metropolis@frametitle@padding}
\setlength{\metropolis@frametitle@padding}{2.2ex}
\newcommand{\metropolis@frametitlestrut@start}{
  \rule{0pt}{\metropolis@frametitle@padding +%
    \totalheightof{%
      \ifcsdef{metropolis@frametitleformat}{\metropolis@frametitleformat X}{X}%
    }%
  }%
}
\newcommand{\metropolis@frametitlestrut@end}{
  \rule[-\metropolis@frametitle@padding]{0pt}{\metropolis@frametitle@padding}
}
\defbeamertemplate{frametitle}{plain}{%
  \nointerlineskip%
  \begin{beamercolorbox}[%
      wd=\paperwidth,%
      sep=0pt,%
      leftskip=\metropolis@frametitle@padding,%
      rightskip=\metropolis@frametitle@padding,%
    ]{frametitle}%
  \metropolis@frametitlestrut@start%
  \insertframetitle%
  \nolinebreak%
  \metropolis@frametitlestrut@end%
  \end{beamercolorbox}%
}
\setbeamertemplate{frametitle continuation}{%
  \usebeamerfont{frametitle}
  \romannumeral \insertcontinuationcount
}
```

`progress bar in head/foot` Template for the progress bar optionally displayed below the frame title on each page. Much of this code is duplicated in the inner theme's template `progress bar in section page`.

```
\newlength{\metropolis@progressinheadfoot}
```

```
\newlength{\metropolis@progressinheadfoot@linewidth}
\setlength{\metropolis@progressinheadfoot@linewidth}{0.4pt}
\setbeamertemplate{progress bar in head/foot}{
  \nointerlineskip
  \setlength{\metropolis@progressinheadfoot}{%
    \paperwidth * \ratio{\insertframenumber pt}{\inserttotalframenumber pt}%
  }%
  \begin{beamercolorbox}[wd=\paperwidth]{progress bar in head/foot}
    \begin{tikzpicture}
      \fill[bg] (0,0) rectangle (\paperwidth, \metropolis@progressinheadfoot@linewi
      \fill[fg] (0,0) rectangle (\metropolis@progressinheadfoot, \metropolis@progressinheadfoot@linewidth);
    \end{tikzpicture}%
  \end{beamercolorbox}
}
```

`appendix` Removes page numbering and per-slide progress bars when `\appendix` is called. This makes it easier to include additional “backup slides” at the end of the presentation, especially in conjunction with the package `appendixnumberbeamer`.

```
\AtBeginDocument{%
  \apptocmd{\appendix}{%
    \pgfkeys{%
      /metropolis/outer/.cd,
      numbering=none,
      progressbar=none}
    }{}{}
}
```

8.3.5 Process package options

```
\metropolis@outer@setdefaults
\ProcessPgfPackageOptions{/metropolis/outer}
```

8.4 METROPOLIS font theme

A `beamer` font theme sets the style of the font used in the document.

8.4.1 Package dependencies

```
\RequirePackage{etoolbox}
\RequirePackage{ifxetex}
\RequirePackage{ifluatex}
\RequirePackage{pgfopts}
```

8.4.2 Load Fira fonts

If the presentation is compiled with XeLaTeX or LuaLaTeX, the fontspec package is loaded and we search for the `Fira` fonts.

```
\ifboolexpr{bool {xetex} or bool {luatex}}{
  \@ifpackageloaded{fontspec}{
    \PassOptionsToPackage{no-math}{fontspec}
  }{
    \RequirePackage[no-math]{fontspec}
  }
```

`\checkfont` Checks if a font is installed; if not, `fontsnotfound` is increased.

```
  \newcounter{fontsnotfound}
  \newcommand{\checkfont}[1]{%
    \suppressfontnotfounderror=1%
    \font\x = ”#1” at 10pt
    \selectfont
    \ifx\x\nullfont%
      \stepcounter{fontsnotfound}%
    \fi%
    \suppressfontnotfounderror=0%
  }

```

`\iffontsavailable` Resets the `fontsnotfound` counter and calls `\checkfont` for each font in the comma separated list in the first argument.

```
  \newcommand{\iffontsavailable}[3]{%
    \setcounter{fontsnotfound}{0}%
    \expandafter\forcsvlist\expandafter%
    \checkfont\expandafter{#1}%
    \ifnum\value{fontsnotfound}=0%
    #2%
  \else%
    #3%
  \fi%
}
```

We search for regular, italic, light, light italic, mono, and mono bold fonts under the default `Fira Sans` and `Fira Mono` names. If this fails, the suffix OT — used by some Linux distributions — will be tried. If this also fails, a warning will be displayed and the standard fonts will be used.

```
\iffontsavailable{Fira Sans Light,%
                  Fira Sans Light Italic,%
                  Fira Sans,%
                  Fira Sans Italic}%
{%
  \setsansfont[ItalicFont={Fira Sans Light Italic},%
               BoldFont={Fira Sans},%
               BoldItalicFont={Fira Sans Italic}]%
              {Fira Sans Light}%
}{%
  \iffontsavailable{Fira Sans Light OT,%
                    Fira Sans Light Italic OT,%
                    Fira Sans OT,%
                    Fira Sans Italic OT}%
  {%
    \setsansfont[ItalicFont={Fira Sans Light Italic OT},%
                 BoldFont={Fira Sans OT},%
                 BoldItalicFont={Fira Sans Italic OT}]%
                {Fira Sans Light OT}%
  }{%
    \PackageWarning{beamerthememetropolis}{%
      Could not find Fira Sans fonts%
    }
  }
}
\iffontsavailable{Fira Mono, Fira Mono Bold}{%
  \setmonofont[BoldFont={Fira Mono Medium}]{Fira Mono}%
}{%
  \iffontsavailable{Fira Mono OT, Fira Mono Bold OT}{%
```

```
      \setmonofont[BoldFont={Fira Mono Medium OT}]{Fira Mono OT}%
    }{%
      \PackageWarning{beamerthememetropolis}{%
        Could not find Fira Mono fonts%
      }
    }
  }
  \AtBeginEnvironment{tabular}{%
    \addfontfeature{Numbers={Monospaced}}%
  }
}{%
  \PackageWarning{beamerthememetropolis}{%
    You need to compile with XeLaTeX or LuaLaTeX to use the Fira fonts%
  }
}
```

This concludes the portion of the code which is only run when compiled with XeLaTeX or LuaLaTeX. The remainder of this package applies regardless of the compiling engine.

8.4.3 General font definitions

```
\setbeamerfont{title}{size=\Large,%
                      series=\bfseries}
\setbeamerfont{author}{size=\small}
\setbeamerfont{date}{size=\small}
\setbeamerfont{section title}{size=\Large,%
                              series=\bfseries}
\setbeamerfont{block title}{size=\normalsize,%
                            series=\bfseries}
\setbeamerfont{block title alerted}{size=\normalsize,%
                                    series=\bfseries}
\setbeamerfont*{subtitle}{size=\large}
\setbeamerfont{frametitle}{size=\large,%
                           series=\bfseries}
\setbeamerfont{caption}{size=\small}
\setbeamerfont{caption name}{series=\bfseries}
\setbeamerfont{description item}{series=\bfseries}
\setbeamerfont{page number in head/foot}{size=\scriptsize}
\setbeamerfont{bibliography entry author}{size=\normalsize,%
```

```
                                          series=\normalfont}
\setbeamerfont{bibliography entry title}{size=\normalsize,%
                                         series=\bfseries}
\setbeamerfont{bibliography entry location}{size=\normalsize,%
                                            series=\normalfont}
\setbeamerfont{bibliography entry note}{size=\small,%
                                        series=\normalfont}
\setbeamerfont{standout}{size=\Large,%
                         series=\bfseries}
```

8.4.4 Title format options

`titleformat title` Controls the format of the title.

```
\pgfkeys{
  /metropolis/font/titleformat title/.cd,
    .is choice,
    regular/.code={%
      \let\metropolis@titleformat\@empty%
      \setbeamerfont{title}{shape=\normalfont}%
    },
    smallcaps/.code={%
      \let\metropolis@titleformat\@empty%
      \setbeamerfont{title}{shape=\scshape}%
    },
    allsmallcaps/.code={%
      \let\metropolis@titleformat\lowercase%
      \setbeamerfont{title}{shape=\scshape}%
      \PackageWarning{beamerthememetropolis}{%
        Be aware that titleformat title=allsmallcaps can lead to problems%
      }
    },
    allcaps/.code={%
      \let\metropolis@titleformat\uppercase%
      \setbeamerfont{title}{shape=\normalfont}
      \PackageWarning{beamerthememetropolis}{%
        Be aware that titleformat title=allcaps can lead to problems%
      }
  },
}
```

`titleformat subtitle` Control the format of the subtitle.

```
\pgfkeys{
  /metropolis/font/titleformat subtitle/.cd,
    .is choice,
    regular/.code={%
      \let\metropolis@subtitleformat\@empty%
      \setbeamerfont{subtitle}{shape=\normalfont}%
    },
    smallcaps/.code={%
      \let\metropolis@subtitleformat\@empty%
      \setbeamerfont{subtitle}{shape=\scshape}%
    },
    allsmallcaps/.code={%
      \let\metropolis@subtitleformat\lowercase%
      \setbeamerfont{subtitle}{shape=\scshape}%
      \PackageWarning{beamerthememetropolis}{%
        Be aware that titleformat subtitle=allsmallcaps can lead to problems%
      }
    },
    allcaps/.code={%
      \let\metropolis@subtitleformat\uppercase%
      \setbeamerfont{subtitle}{shape=\normalfont}%
      \PackageWarning{beamerthememetropolis}{%
        Be aware that titleformat subtitle=allcaps can lead to problems%
      }
    },
}
```

`titleformat section` Controls the format of the section title.

```
\pgfkeys{
  /metropolis/font/titleformat section/.cd,
    .is choice,
    regular/.code={%
      \let\metropolis@sectiontitleformat\@empty%
      \setbeamerfont{section title}{shape=\normalfont}%
    },
    smallcaps/.code={%
      \let\metropolis@sectiontitleformat\@empty%
      \setbeamerfont{section title}{shape=\scshape}%
    },
    allsmallcaps/.code={%
      \let\metropolis@sectiontitleformat\MakeLowercase%
      \setbeamerfont{section title}{shape=\scshape}%
      \PackageWarning{beamerthememetropolis}{%
        Be aware that titleformat section=allsmallcaps can lead to problems%
      }
    },
    allcaps/.code={%
      \let\metropolis@sectiontitleformat\MakeUppercase%
      \setbeamerfont{section title}{shape=\normalfont}%
      \PackageWarning{beamerthememetropolis}{%
        Be aware that titleformat section=allcaps can lead to problems%
      }
    },
}
```

`frametitleformat` Control the format of the frame title.

```
\pgfkeys{
  /metropolis/font/titleformat frame/.cd,
    .is choice,
    regular/.code={%
      \let\metropolis@frametitleformat\@empty%
      \setbeamerfont{frametitle}{shape=\normalfont}%
    },
    smallcaps/.code={%
      \let\metropolis@frametitleformat\@empty%
      \setbeamerfont{frametitle}{shape=\scshape}%
    },
    allsmallcaps/.code={%
      \let\metropolis@frametitleformat\MakeLowercase%
      \setbeamerfont{frametitle}{shape=\scshape}%
      \PackageWarning{beamerthememetropolis}{%
        Be aware that titleformat frame=allsmallcaps can lead to problems%
      }
    },
    allcaps/.code={%
      \let\metropolis@frametitleformat\MakeUppercase%
      \setbeamerfont{frametitle}{shape=\normalfont}
      \PackageWarning{beamerthememetropolis}{%
        Be aware that titleformat frame=allcaps can lead to problems%
      }
    },
}
```

`titleformat aliases` Allows `titleformat title` et al. to be used in the `\usetheme` declaration, where LaTeX automatically removes all spaces.

```
\pgfkeys{
  /metropolis/font/.cd,
  titleformattitle/.code=\pgfkeysalso{titleformat title=#1},
  titleformatsubtitle/.code=\pgfkeysalso{titleformat subtitle=#1},
  titleformatsection/.code=\pgfkeysalso{titleformat section=#1},
  titleformatframe/.code=\pgfkeysalso{titleformat frame=#1},
}
```

`metropolis@font@setdefaults` Sets default values for font theme options.

```
\newcommand{\metropolis@font@setdefaults}{
  \pgfkeys{/metropolis/font/.cd,
    titleformat title=regular,
    titleformat subtitle=regular,
    titleformat section=regular,
    titleformat frame=regular,
  }
}
```

We first define hooks to change the case format of the titles.

```
\def\metropolis@titleformat#1{#1}
\def\metropolis@subtitleformat#1{#1}
```

```
\def\metropolis@sectiontitleformat#1{#1}
\def\metropolis@frametitleformat#1{#1}
```

To make the uppercase and lowercase macros work in the title, subtitle, etc., we have to patch the appropriate `beamer` commands that set their values. This solution was suggested by Enrico Gregorio in an answer to this StackExchange question.

```
\patchcmd{\beamer@title}%
  {\def\inserttitle{#2}}%
  {\def\inserttitle{\metropolis@titleformat{#2}}}%
  {}%
  {\PackageError{beamerfontthememetropolis}{Patching title failed}\@ehc}
\patchcmd{\beamer@subtitle}%
  {\def\insertsubtitle{#2}}%
  {\def\insertsubtitle{\metropolis@subtitleformat{#2}}}%
  {}%
  {\PackageError{beamerfontthememetropolis}{Patching subtitle failed}\@ehc}
\patchcmd{\sectionentry}
  {\def\insertsectionhead{#2}}
  {\def\insertsectionhead{\metropolis@sectiontitleformat{#2}}}
  {}
  {\PackageError{beamerfontthememetropolis}{Patching section title failed}\@ehc}
\@tempswafalse
\patchcmd{\beamer@section}
  {\def\insertsectionhead{\hyperlink{Navigation\the\c@page}{#1}}}
  {\def\insertsectionhead{\hyperlink{Navigation\the\c@page}{%
    \metropolis@sectiontitleformat{#1}}}}
  {\@tempswatrue}
  {}
\patchcmd{\beamer@section}
  {\protected@edef\insertsectionhead{\noexpand\hyperlink{Navigation\the\c@page}{#1}
  {\protected@edef\insertsectionhead{\noexpand\hyperlink{Navigation\the\c@page}{%
    \noexpand\metropolis@sectiontitleformat{#1}}}}
  {\@tempswatrue}
  {}
\if@tempswa\else
  \PackageError{beamerfontthememetropolis}{Patching section title failed}\@ehc
\fi
```

```
\@tempswafalse
\patchcmd{\beamer@subsection}
  {\def\insertsubsectionhead{\hyperlink{Navigation\the\c@page}{#1}}}
  {\def\insertsubsectionhead{\hyperlink{Navigation\the\c@page}{%
    \metropolis@sectiontitleformat{#1}}}}
  {\@tempswatrue}
  {}
\patchcmd{\beamer@subsection}
  {\protected@edef\insertsubsectionhead{\noexpand\hyperlink{Navigation\the\c@page}{
  {\protected@edef\insertsubsectionhead{\noexpand\hyperlink{Navigation\the\c@page}{
    \noexpand\metropolis@sectiontitleformat{#1}}}}
  {\@tempswatrue}
  {}
\if@tempswa\else
  \PackageError{beamerfontthememetropolis}{Patching section title failed}\@ehc
\fi
```

Similarly, to make the `\MakeLowercase` and `\MakeUppercase` macros work in the frame title we have to patch `\beamer@@frametitle`.

```
\patchcmd{\beamer@@frametitle}
  {{%
      \gdef\insertframetitle{{#2\ifnum\beamer@autobreakcount>0\relax{}\space%
      \usebeamertemplate*{frametitle continuation}\fi}}%
    \gdef\beamer@frametitle{#2}%
    \gdef\beamer@shortframetitle{#1}%
    }}
  {{%
      \gdef\insertframetitle{{\metropolis@frametitleformat{#2}\ifnum%
      \beamer@autobreakcount>0\relax{}\space%
      \usebeamertemplate*{frametitle continuation}\fi}}%
    \gdef\beamer@frametitle{#2}%
    \gdef\beamer@shortframetitle{#1}%
    }}
  {}
  {\PackageError{beamerfontthememetropolis}{Patching frame title failed}\@ehc}
```

8.4.5 Process package options

```
\metropolis@font@setdefaults
```

```
\ProcessPgfPackageOptions{/metropolis/font}
```

8.5 METROPOLIS color theme

8.5.1 Package dependencies

```
\RequirePackage{pgfopts}
```

8.5.2 Options

`block` Optionally adds a light grey background to block environments like `theorem` and `example`.

```
\pgfkeys{
  /metropolis/color/block/.cd,
    .is choice,
    transparent/.code=\metropolis@block@transparent,
    fill/.code=\metropolis@block@fill,
}
```

`colors` Provides the option to have a dark background and light foreground instead of the reverse.

```
\pgfkeys{
  /metropolis/color/background/.cd,
    .is choice,
    dark/.code=\metropolis@colors@dark,
    light/.code=\metropolis@colors@light,
}
```

`etropolis@color@setdefaults` Sets default values for color theme options.

```
\newcommand{\metropolis@color@setdefaults}{
  \pgfkeys{/metropolis/color/.cd,
    background=light,
    block=transparent,
  }
}
```

8.5.3 Base colors

```
\definecolor{mDarkBrown}{HTML}{604c38}
\definecolor{mDarkTeal}{HTML}{23373b}
\definecolor{mLightBrown}{HTML}{EB811B}
\definecolor{mLightGreen}{HTML}{14B03D}
```

8.5.4 Base styles

All colors in METROPOLIS are derived from the definitions of `normal text`, `alerted text`, and `example text`.

```
\newcommand{\metropolis@colors@dark}{
  \setbeamercolor{normal text}{%
    fg=black!2,
    bg=mDarkTeal
  }
  \usebeamercolor[fg]{normal text}
}
\newcommand{\metropolis@colors@light}{
  \setbeamercolor{normal text}{%
    fg=mDarkTeal,
    bg=black!2
  }
}
\setbeamercolor{alerted text}{%
  fg=mLightBrown
}
\setbeamercolor{example text}{%
  fg=mLightGreen
}
```

8.5.5 Derived colors

The titles and structural elements (e.g. `itemize` bullets) are set in the same color as `normal text`. This would ideally done by setting `normal text` as a parent style, which we do to set `titlelike`, but this doesn't work for `structure` as its foreground is set explicitly in `beamercolorthemedefault.sty`.

```
\setbeamercolor{titlelike}{use=normal text, parent=normal text}
\setbeamercolor{author}{use=normal text, parent=normal text}
\setbeamercolor{date}{use=normal text, parent=normal text}
```

```
\setbeamercolor{institute}{use=normal text, parent=normal text}
\setbeamercolor{structure}{use=normal text, fg=normal text.fg}
```

The “primary” palette should be used for the most important navigational elements, and possibly of other elements. METROPOLIS uses it for frame titles and slides.

```
\setbeamercolor{palette primary}{%
  use=normal text,
  fg=normal text.bg,
  bg=normal text.fg
}
\setbeamercolor{frametitle}{%
  use=palette primary,
  parent=palette primary
}
```

The METROPOLIS inner or outer themes optionally display progress bars in various locations. Their color is set by `progress bar` but the two different kinds can be customized separately. The horizontal rule on the title page is also set based on the progress bar color and can be customized with `title separator`.

```
\setbeamercolor{progress bar}{%
  use=alerted text,
  fg=alerted text.fg,
  bg=alerted text.fg!50!black!30
}
\setbeamercolor{title separator}{
  use=progress bar,
  parent=progress bar
}
\setbeamercolor{progress bar in head/foot}{%
  use=progress bar,
  parent=progress bar
}
\setbeamercolor{progress bar in section page}{
  use=progress bar,
  parent=progress bar
}
```

Block environments such as `theorem` and `example` have no background color by default. The option `block=fill` sets a background color based on the background and foreground of `normal text`. The option `block=transparent` reverts the block environments to an empty background, which can be useful if changing colors mid-presentation.

```
\newcommand{\metropolis@block@transparent}{
  \setbeamercolor{block title}{%
    use=normal text,
    fg=normal text.fg,
    bg=
  }
  \setbeamercolor{block body}{
    bg=
  }
}
\newcommand{\metropolis@block@fill}{
  \setbeamercolor{block title}{%
    use=normal text,
    fg=normal text.fg,
    bg=normal text.bg!80!fg
  }
  \setbeamercolor{block body}{
    use={block title, normal text},
    bg=block title.bg!50!normal text.bg
  }
}
\setbeamercolor{block title alerted}{%
    use={block title, alerted text},
    bg=block title.bg,
    fg=alerted text.fg
}
\setbeamercolor{block title example}{%
    use={block title, example text},
    bg=block title.bg,
    fg=example text.fg
}
\setbeamercolor{block body alerted}{use=block body, parent=block body}
\setbeamercolor{block body example}{use=block body, parent=block body}
```

Footnotes

```
\setbeamercolor{footnote}{fg=normal text.fg!90}
\setbeamercolor{footnote mark}{fg=.}
```

8.5.6 Process package options

```
\metropolis@color@setdefaults
\ProcessPgfPackageOptions{/metropolis/color}
```

```
\mode<all>
```

8.6 Tol `pgfplots` theme

Paul Tol’s 12-color palette[1] is as follows:

```
\definecolor{TolDarkPurple}{HTML}{332288}
\definecolor{TolDarkBlue}{HTML}{6699CC}
\definecolor{TolLightBlue}{HTML}{88CCEE}
\definecolor{TolLightGreen}{HTML}{44AA99}
\definecolor{TolDarkGreen}{HTML}{117733}
\definecolor{TolDarkBrown}{HTML}{999933}
\definecolor{TolLightBrown}{HTML}{DDCC77}
\definecolor{TolDarkRed}{HTML}{661100}
\definecolor{TolLightRed}{HTML}{CC6677}
\definecolor{TolLightPink}{HTML}{AA4466}
\definecolor{TolDarkPink}{HTML}{882255}
\definecolor{TolLightPurple}{HTML}{AA4499}
```

To use these colors, we describe “cycle lists” from which PGF chooses styles for the different series in a chart.

`mbarplot cycle` Colors and styles intended for bar charts with up to 12 series.

```
\pgfplotscreateplotcyclelist{mbarplot cycle}{%
  {draw=TolDarkBlue,    fill=TolDarkBlue!70},
  {draw=TolLightBrown,  fill=TolLightBrown!70},
  {draw=TolLightGreen,  fill=TolLightGreen!70},
  {draw=TolDarkPink,    fill=TolDarkPink!70},
  {draw=TolDarkPurple,  fill=TolDarkPurple!70},
  {draw=TolDarkRed,     fill=TolDarkRed!70},
```

---

[1] Tol actually describes several palettes; these colours are taken from the bottom row of Figure 3 in his technical note.

```
  {draw=TolDarkBrown,   fill=TolDarkBrown!70},
  {draw=TolLightRed,    fill=TolLightRed!70},
  {draw=TolLightPink,   fill=TolLightPink!70},
  {draw=TolLightPurple, fill=TolLightPurple!70},
  {draw=TolLightBlue,   fill=TolLightBlue!70},
  {draw=TolDarkGreen,   fill=TolDarkGreen!70},
}
```

`mlineplot cycle` Colors and styles intended for line charts with up to 4 series.

```
\pgfplotscreateplotcyclelist{mlineplot cycle}{%
  {TolDarkBlue, mark=*, mark size=1.5pt},
  {TolLightBrown, mark=square*, mark size=1.3pt},
  {TolLightGreen, mark=triangle*, mark size=1.5pt},
  {TolDarkBrown, mark=diamond*, mark size=1.5pt},
}
```

However, the above cycle lists are not applied automatically. We still need to define styles — `mlineplot` and `mbarplot` — that the user can apply to the axis of a `pgfplots` chart to use the colors. We'll also take the opportunity to adjust the display of chart axes when these styles are used.

```
\pgfplotsset{
  compat=1.9,
```

`mlineplot` A style to apply to the axis of a PGF line plot.

```
  mlineplot/.style={
    mbaseplot,
    xmajorgrids=true,
    ymajorgrids=true,
    major grid style={dotted},
    axis x line=bottom,
    axis y line=left,
    legend style={
      cells={anchor=west},
      draw=none
    },
    cycle list name=mlineplot cycle,
  },
```

`mbarplot`
`horizontal mbarplot`

A style to apply to the axis of a PGF bar chart. `mbarplot` uses vertical bars by default, while `horizontal mbarplot` has horizontal bars as the name implies. Their shared properties are factored out into the internal style `mbarplot base`.

```
mbarplot base/.style={
  mbaseplot,
  bar width=6pt,
  axis y line*=none,
},
mbarplot/.style={
  mbarplot base,
  ybar,
  xmajorgrids=false,
  ymajorgrids=true,
  area legend,
  legend image code/.code={%
    \draw[#1] (0cm,-0.1cm) rectangle (0.15cm,0.1cm);
  },
  cycle list name=mbarplot cycle,
},
horizontal mbarplot/.style={
  mbarplot base,
  xmajorgrids=true,
  ymajorgrids=false,
  xbar stacked,
  area legend,
  legend image code/.code={%
    \draw[#1] (0cm,-0.1cm) rectangle (0.15cm,0.1cm);
  },
  cycle list name=mbarplot cycle,
},
```

`mbaseplot`

Adjusts the appearance of the axes in a PGF chart.

```
mbaseplot/.style={
  legend style={
    draw=none,
    fill=none,
    cells={anchor=west},
  },
    x tick label style={
      font=\footnotesize
    },
    y tick label style={
      font=\footnotesize
    },
    legend style={
      font=\footnotesize
    },
    major grid style={
      dotted,
    },
    axis x line*=bottom,
  },
  disable thousands separator/.style={
    /pgf/number format/.cd,
      1000 sep={}
  },
}
```