Codebook

A LaTeX Class for Technical Books

Daniel Heck

To me style is just the outside of content,
and content the inside of style,
like the outside and the inside of the human body—
both go together, they can't be separated.

Jean-Luc Godard

Contents

Introduction

Codebook is a LaTeX package for writing technical books, especially books in computer science, mathematics, and related fields. It is based on the standard book class, but it completely changes the design and layout and adds several new commands and environments for typesetting the following:

- Copyright and dedication pages;
- Epigraphs at the beginning of a chapter;
- Exercises and solutions;
- Wide figures and tables that can reach into the margins;
- Code listings, either embedded into the text or floating;
- Predefined colors.

It is based on the standard book class so it should be compatible with most LaTeX packages out there.

This class grew out of a set of macros and LaTeX definitions I wrote for my personal book projects. My goal was to support a modern design that looks good when printed on paper and when viewed as an eBook. *Codebook* uses three font families: the main text is set in *Palatino*, headings in *Source Sans Pro*, and source code *Noto Mono Condensed*. All three fonts are freely available and are included in current versions of TeXlive.

Visit *http://github.com/dheck/codebook* for the latest version.

Part I

Overview

Parts are traditionally used to structure large documents into groups of related chapters. For a small manual like this, parts are certainly overkill, but it allows you to see how they are formatted. Notice that *Codebook* allows you to include text after the title of a part, like here.

Chapter 1

Writing Books using Codebook

Like any other LaTeX document class, *Codebook* is loaded using the `\documentclass` command:

```
\documentclass[options]{codebook}
```

The following options are supported:

- `minted`: Loads the `minted` package and configures it to work well with *Codebook*.
- `drafting`: Adds the current day to the footer and passes the option on to other packages that support it. Similar to the `draft` option in the standard LaTeX class this option is meant to be used while preparing the document and should be removed upon publication.

1.1 Formatting Text

Since *Codebook* is based on the `book` class, all standard LaTeX commands and environments are supported, although many of them have been tweaked to fit the general design. In addition, *Codebook* provides several new commands and environments that are particularly useful in technical documents and textbooks.

Different font sizes and can be selected using commands such as `\tiny` and `\large`; an overview of all predefined font sizes is shown in Figure 1.1.

Fonts and font sizes

Codebook defines a color palette you may want to use for highlighting text and for things like illustrations and code listings. The full list of predefined colors is shown in Figure 1.1. In addition to the six basic colors cbBlue, cbGreen, cbRed, cbOrange, cbYellow, cbBrown, and cbPurple, there are also six

Predefined colors

<div align="center">

TABLE 1.1

Predefined font sizes.

</div>

\tiny	6 pt	The quick brown fox
\scriptsize	7 pt	The quick brown fox
\footnotesize	8 pt	The quick brown fox
\small	9 pt	The quick brown fox
\normalsize	10 pt	The quick brown fox
\large	11.5 pt	The quick brown fox
\Large	13 pt	The quick brown fox
\LARGE	15 pt	The quick brown fox
\huge	20 pt	The quick brown fox
\Huge	24 pt	The quick brown fox
\HUGE	30 pt	The quick brown fox

light variants like `cbLightBlue` and six dark variants like `cbDarkRed`.[1]

1.1.1 Notes

Codebook provides several ways to add notes and supplementary information to the text. As in all LATEX classes, you can add footnotes[2] using the `\footnote` command. Footnotes are tucked away at the bottom of the page and are therefore (by design!) easy to ignore. Use them for information that is truly optional, or for information that you only include for the sake of completeness.

The note *environment*   To include notes inside the running text, use the "`note`" environment:

```
\begin{note}
Beware of bugs in the above code;
I have only proved it correct, not tried it.
\end{note}
```

This produces the following output:

> Beware of bugs in the above code; I have only proved it correct, not tried it.

Since notes are visually set off from the main text it's clear that they provide supplementary information. They are best used for information that every

---

[1]The color palette is taken from the *Tango Desktop Project*; see *https://en.wikipedia.org/wiki/ Tango_Desktop_Project*

[2]Like this one.

cbLightBlue	cbBlue	cbDarkBlue
cbLightGreen	cbGreen	cbDarkGreen
cbLightRed	cbRed	cbDarkRed
cbLightOrange	cbOrange	cbDarkOrange
cbLightYellow	cbYellow	cbDarkYellow
cbLightBrown	cbBrown	cbDarkBrown
cbLightPurple	cbPurple	cbDarkPurple

Figure 1.1. Predefined colors.

reader should aware of but that can be safely skimmed.

Supplementary information that you want readers should be aware of, even if they don't need to understand all details, us \marginnote command:

Margin notes

```
\marginnote{Margin notes}
```

This example produces the margin note shown at the start of this paragraph. Margin notes generally serve a different purpose than notes embedded in the text and footnotes: They contain information that helps the reader, for example by summarizing the main points of a paragraph or group of paragraphs or by visually dividing a section into smaller pieces.

Finally, the aside environment can be used to present information that is only indirectly related to the contents of the text. For example, Box 1 on page 6 describes a way to use footnotes for representing hyperlinks in documents that are meant to be printed *and* read electronically.

Boxes

1.1.2 Code and Listings

For code and verbatim text, *codebook* loads and configures the *fancyvrb* package.

```
\begin{Verbatim}
verbatim text
\end{Verbatim}
```

produces

```
verbatim text
```

> **Box 1.   Typesetting Hyperlinks**
>
> A common problem when writing technical documents is dealing with links to websites and other online resources. Using hyperlinks is usually the best solution for electronic documents; in LaTeX this can be done using the \href command from the hyperref package.
>
> In documents that are also meant to be printed and read offline, hyperlinks are not useful because the URL is invisible. There are two better solutions. If the web page is considered a primary or secondary source, add it to the bibliography and cite it like a book or an article [**wikipedia:hyperlink**]. Otherwise, simply use a hyperlink[a] followed by a footnote that contains the URL.
>
> ───────────
> [a]*https://en.wikipedia.org/wiki/Hyperlink*

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

Listing 1.2. Hello world in Java.

If you look closely, you will notice that verbatim blocks use a slightly smaller font size than verbatim text inside a paragraph. Code inside Verbatim environments is indented to distinguish it from the surrounding text.

In addition, *codebook* provides two environments for longer listings that can float, similar to figures and tables. The first environment is called listing:

The listing *environment*

```
\begin{listing}[htb]
...
\caption{...}
\label{...}
\end{listing}
```

As with other floats, you can influence the positioning of a particular listing by specifying location specifiers; the default is "[tbp]".

By default, the listing environment sets up fancyvrb to draw thin lines above and below the code, as illustrated in Listing 1.2. Packages that are

based in *fancyvrb* automatically work correctly; Other LATEX packages for displaying code such as *lstlisting* provide their own formatting options and must be configured by hand.

The second environment for embedding code listings is called `lst`. Unlike the normal `listing` environment, `lst`s don't float and are embedded in the surrounding text, similar to the way `equation` or `itemize` can occur in the middle of a normal paragraph. The following example is typeset using `lst`

```
\begin{lst}[lst:hellopython]{Hello World in Python.}
\begin{Verbatim}
print("Hello World")
\end{Verbatim}
\end{lst}
```

The `lst` *environment.*

The caption of `lst` is typeset in the margin:

```
print("Hello World")
```

1.4
Hello World in Python.

Alternatively, you can use the `lst*` environment to produce an unnumbered listing:

```
\begin{lst*}{Hello World in Python.}
\begin{Verbatim}
print("Hello World")
\end{Verbatim}
\end{lst*}
```

The `lst` *environment.*

Since `lst*` is unnumbered it doesn't take the label as an optional argument, and it doesn't output a number in front of the caption.

for interleaving code examples with the main text. Code enclosed in a `lst` environment can be broken across pages.

The `minted` package defines its own `listing` environment that conflicts with the one provided by *Codebook*. To use `minted`, you can load the *Codebook* class as follows:

```
\PassOptionsToPackage{optionlist}{minted}
\documentclass[minted]{codebook}
```

1.1.3 Exercises and Solutions

Codebook provides an easy way to typeset exercises and their solutions. Exercises are enclosed in the `exercise` environment which takes an optional argument

that includes a title. Inside each exercise you can use the answer environment to specify a solution. The following example specifies a single exercise and its solution:

```
\begin{exercise}[Typesetting formulas]
\label{ex:binomial}
Explain how to typeset the following formulas in \LaTeX{}:
\begin{tasks}
  \item $(x+y)^2=x^2 + 2xy + y^2$;
  \item $e^{i\pi}=-1$.
\end{tasks}

\begin{answer}
\begin{tasks}
  \item \verb|$(x+y)^2=x^2 + 2xy + y^2$|
  \item \verb|$e^{i\pi}=-1$|
\end{tasks}
\end{answer}
\end{exercise}
```

The tasks environment used in the example above behaves like itemize, except that it uses alphabetic labels. At the point in the document where the exercise environment appears, only the exercise itself is printed:

Exercise 1.1 *Typesetting formulas*

Explain how to typeset the following formulas in LaTeX:

a) $(x + y)^2 = x^2 + 2xy + y^2$;

b) $e^{i\pi} = -1$.

You can refer to exercises and tasks by labeling them as shown in the previous example. The label of an exercise contains both the number of the chapter and the exercise, so writing Exercise~\ref{ex:binomial} produces "Exercise 1.1." The label of a task contains just its alphabetic tag, so Task~\ref{task:euler} produces "Task b)."

The answers are not typeset immediately but saved to a separate file. You can load all answers defined in the current document using the \inputanswers command:

```
\inputanswers
```

This command outputs the answers of all exercises that were encountered so far.

For the single exercise defined above, \inputanswers produces the following output:

Exercise 1.1

  a) `$(x+y)^2=x^2 + 2xy + y^2$`

  b) `$e^{i\pi}=-1$`

What if you want to print the answers output by one LATEX document in another document, for example when producing a separate solutions manual? Since the lines in an `answer` environment are simply written to a file called "`\jobname.solution`," where the macro `\jobname` is the name of the current document, you can include the answers from a different LATEX document using the `\input` or `\include` commands. For example, the answers defined in a document called `myboook.tex` can be loaded in another document as follows:

`**\input**{mybook.solution}`

1.1.4 Epigraphs

> Real stupidity beats artificial intelligence every time.
>
> —Terry Pratchett, *Hogfather*

Many nonfiction books start each chapter with an *epigraph*, an inspirational or humorous quote that fits the theme the chapter. To typeset such epigraphs, *Codebook* provides the `epigraphs` environment that can contain one or more `\epigraph` commands. The following example shows how the quote at the beginning of this subsection was specified:

```
\begin{epigraphs}
\epigraph[\textsc{Terry Pratchett}, \emph{Hogfather}]
  {Real stupidity beats artificial intelligence every time.}
\end{epigraphs}
\noindent Many nonfiction books start each chapter with an \emph{epigraph},
```

The \noindent command at the beginning of the first paragraph prevents LATEX from indenting the first line after the quote.

The following parameters determine the layout of epigraphs:

- \epigraphwidth (default: 3 in). The width of the box that contains each epigraph.
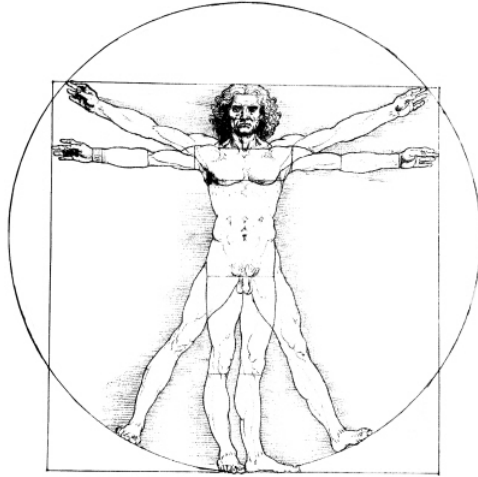
Figure 1.2. A normal figure with a centered caption.

- \beforepigraphskip (default: 0 in). The amount of space to insert before the epigraphs environment.
- \afterepigraphskip (default: 2 pc). The amount of space to insert after the epigraphs environment.

The values of these parameters can be changed using the \setlength command.

1.2 **Floating Material**

1.2.1 Figures

In addition to the standard figure environment, *Codebook* provides a widefigure environment that stretches across the entire width of the page. An example of the standard figure environment is shown in Figure 1.2. Notice that captions are centered, so the contents of the figure should usually be centered as well.

For presenting images or other material that is wider than the body of the page, you can use the widefigure environment that extends into the margin of the page:

```
\begin{widefigure}
  \includegraphics[width=\fulltextwidth]{mandel-wide.png}
  \caption{A wide figure.}
\end{widefigure}
```
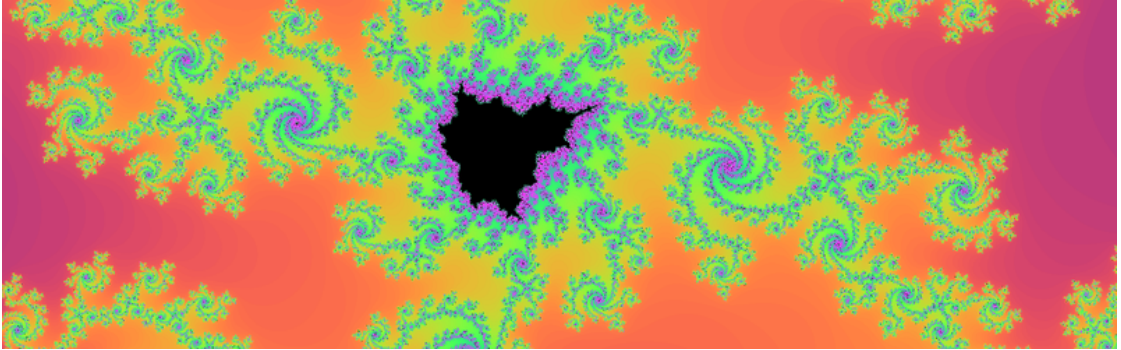
Figure 1.3. A wide figure.

<div align="center">

TABLE 1.2

Quarks

</div>

Generation	Name	Symbol	Spin	Charge [e]	Mass [MeV/c^2]
1	up	u	1/2	+2/3	2.2 ± 0.6
	down	d	1/2	−1/3	4.6 ± 0.5
2	charm	c	1/2	+2/3	1280 ± 30
	strange	s	1/2	−1/3	96 ± 8
3	top	t	1/2	+2/3	173 100 ± 600
	bottom	b	1/2	−1/3	4180 ± 40

The result is shown in Figure 1.3: it looks like a normal `figure` with the caption under the image, but it extends across the text block plus the margin area. The length `\fulltextwidth` used in the example above is defined by *Codebook* as the width of the body plus the width of the margin.

1.2.2 Tables

Table 1.2 illustrates the recommended visual appearance of tables in *Codebook*

- The `\caption` should come before the body of the table.
- The `booktabs` package is used to draw horizontal rules.

The recommended way of formatting tables is to use the `booktabs` package, which is loaded automatically. For example, Table 1.2 is typeset as follows

```
\begin{table}
  \centering\small
```

```
\caption{Quarks}
\label{tab:quarks}
\begin{tabular}{cccccS[separate-uncertainty=true]}
\toprule
  \textbf{Generation} & \textbf{Name} & \textbf{Symbol} & \textbf{Spin} & \textbf{
\midrule
   \multirow{2}{*}{1} & up      & u & $1/2$ & $+2/3$ & 2.2 +- 0.6\\
      & down    & d & $1/2$ & $-1/3$ & 4.6 +- 0.5\\
   \midrule
    ...
\bottomrule
\end{tabular}
\end{table}
```

(This example relies on two additional packages that aren't loaded by *Codebook*:
the siunitx package for formatting columns of numerical data and the multirow
package for the \multirow command.)