Journal of Object Technology
Published by AITO — Association Internationale pour les Technologies Objets
http://www.jot.fm/

Typesetting guidelines for the Journal of Object Technology

Damien Pollet[a] Oscar Nierstrasz[b] Stéphane Ducasse[a]
Lynn V. Siebel[ab]

a. RMoD, Inria Lille Nord Europe, France
http://rmod.lille.inria.fr

b. Software Composition Group, University of Bern, Switzerland
http://scg.unibe.ch

Abstract This short manual documents the `jot.cls` LaTeX document class, and provides guidelines and advice on how to use it to prepare and typeset article manuscripts for submission to JOT, the Journal of Object Technology.

Keywords typography, guideline, manual

1 Installation and compatibility

The `jot.cls` project is hosted online at `http://github.com/jotfm/jot`. You can download stable versions from there, or directly clone the development sources from the version control repository. To install the package, simply copy the files somewhere where TeX can find them.

The class is developed and tested using the pdfLaTeX toolchain, which is readily available in any modern TeX distribution; the author uses TeXlive on a Mac. Besides pdfTeX, other TeX engines or backends like `dvips` or `dvipdfm` *should* work but are not supported.[1] The `jot.cls` class requires the following packages, which are all part of the standard TeXlive contents:

`booktabs`	`calc`	`caption`	`eso-pic`
`geometry`	`graphicx`	`hyperref`	`ifthen`
`keyval`	`listings`	`placeins`	`ragged2e`
`refcount`	`soul`	`wrapfig`	`xcolor`

[1] The code does nothing to break them, at least not intentionally, so authors are free to work; nevertheless, in case the journal editors have to recompile articles from the LaTeX sources, it's best if all submissions are guaranteed to compile cleanly with a single engine.

Damien Pollet, Oscar Nierstrasz, Stéphane Ducasse, Lynn V. Siebel. *Typesetting guidelines for the Journal of Object Technology*. Licensed under . In Journal of Object Technology, vol. 0, 2010, pages 0:1–8. Available at http://github.com/jotfm/jot

```
\documentclass{jot}
⟨packages and preamble declarations⟩

\title{ ⟨paper title⟩ }
\author[⟨info⟩]{⟨name⟩}{ ⟨bio text⟩ }
⟨more authors...⟩
\affiliation{⟨identifier⟩}{ ⟨description⟩ }
⟨more affiliations...⟩
\jotdetails{ ⟨publication information⟩ }

\begin{document}
    \begin{abstract}
        ⟨···⟩
    \end{abstract}
    \keywords{⟨comma-separated list⟩}

    ⟨manuscript content...⟩

    \backmatter
    ⟨appendices...⟩
    \bibliographystyle{alphaurl}
    \bibliography{⟨bib files⟩}

    \abouttheauthors
    \begin{acknowledgments}
        ⟨···⟩
    \end{acknowledgments}
\end{document}
```

Listing 1 – Template for a new article main source file.

2 General document structure

The `jot.cls` class builds on the standard `article.cls` from LaTeX, so the document structure is pretty standard. The main differences concern how to declare the title, authors, affiliations, and publication information, and the end of the document. See Listing 1, as well as the `template.tex` file in the `jot.cls` distribution for a more complete, reusable starting point.

We describe the syntax of the commands in details in the next section. For now, here is a summary of the differences:

- title and author information is declared in the preamble and is automatically typeset; there is no need to call the `\maketitle` macro at the beginning of the document;
- authors are declared independently, using one `\author` declaration each, and similarly for affiliations;
- the `\jotdetails` command specifies journal publication information such as year, volume, number...

The main body of the article is organized just like with the standard `article` class, until we get to `\backmatter`. This declaration marks the end of the manuscript text and the beginning of reference material; floating figures or tables that were postponed from the article body will be typeset at that point. If you need appendices, they should go just after `\backmatter`; the bibliographic references and the author biographies should always end the article.

3 Preamble, title, author, and publication data

3.1 Title

Define the title the usual way, using `\title`; if your title spans multiple lines, you can use \\ to split it at better points.

`\title{`⟨*text*⟩`}`

The main title is automatically used in page headers and PDF metadata, but you can override it using the optional declarations `\runningtitle` or `\pdftitle`. Usually only the `\runningtitle` override will be necessary because `\pdftitle` takes the same value by default.

`\runningtitle{`⟨*text*⟩`}`
`\pdftitle{`⟨*text*⟩`}`

3.2 Author information

In contrast with the standard LaTeX classes, authors are declared separately, using an `\author` declaration each. Authors will appear in the order they were declared.

`\author[`⟨*options*⟩`]{`⟨*name*⟩`}{`⟨*bio text*⟩`}`

The first mandatory argument ⟨*name*⟩ defines the author's name. Nothing else should go there, as this value is used in several points in the typesetting; in particular, the `\thanks` macro is inactive: use the affiliation, acknowledgments, or biography texts instead.

The second mandatory parameter ⟨*bio text*⟩ defines the biography and contact paragraphs that appear at the end of the article, in the *About the authors* section; leaving ⟨*bio text*⟩ completely empty will suppress this author's entry there.

The optional parameter ⟨*options*⟩ is a list of comma-separated `key=value` definitions:

- `affiliation=lab`, or `affiliation={lab1,lab2}`

  Attach affiliations with the given identifiers to the author.

- `photo=filename`

  Point to the image file with the author's portrait. No need to specify the file extension. The photo should be of proper definition and proportions, however; a square or a squarish vertical rectangle about 200–300 pixels wide is good.

- `nowrap`

  Specify this option to adjust the layout of the biography text, if it does not flow under the picture by at least one or a couple lines.

Key	Type	Value
`licence`	string	Licence the authors wish to publish under (choose from: `ccby`, `ccbynd`),
`year`	number	Publication year,
`volume`	number	... volume,
`articleno`	number	... article number.
`received`	date	Dates of initial submission,
`published`	date	... final publication,
`revised`	date	... and latest revision of the paper.
`doisuffix`	string	DOI identifier for the paper, without the resolver prefix URL.

Table 1 – Option keys for `jotdetails`.

Finally, as with the title, you can override the authors list in the headings or PDF metadata. Both can take either the final text or an `\and`-separated list of authors.

`\runningauthor{`⟨*names*⟩`}`
`\pdfauthor{`⟨*names*⟩`}`

3.3 Affiliations

Affiliations are typeset in an list below the names of the authors; this allows for any ordering convention between authors and affiliations, and for authors that have multiple affiliations. The ⟨*identifier*⟩ makes the link between the `affiliation` value in the author declaration and the affiliation information. Keep the ⟨*description text*⟩ compact vertically, two or three lines at most.

`\affiliation{`⟨*identifier*⟩`}{`⟨*description text*⟩`}`

3.4 Publication information

The `\jotdetails` declaration defines publication and indexing information about the article, in `key=value` form (see Table 1). Usually, you will just specify the article-specific part of the DOI identifier with `doisuffix`, since all JOT articles will have a DOI of the form `10.5381/doisuffix`. The `url` key only serves as a fallback in the page footers when no DOI was specified.

`\jotdetails{`⟨*key-value info*⟩`}`

3.5 Appendices and bibliography

Any appendices immediately follow the `\backmatter` declaration; you don't need to call `\appendix`.

Be sure to include DOIs (Digital Object Identifiers) for all cited articles, where available.

```
@article{JOT:issue2010-09/editorial,
  author = {Oscar Nierstrasz},
  title = {Ten Things I Hate About Object-Oriented Programming},
  journal = {Journal of Object Technology},
  volume = {9},
  number = {5},
  issn = {1660-1769},
  year = {2010},
  month = sep,
  doi = {10.5381/jot.2010.9.5.e1}
}
```

In the bibliography, use the `alphaurl` style for reference keys, to ensure that DOIs and URLs will appear as links in the bibliography.

```
\bibliographystyle{alphaurl}
```

The bibliographic references follow the appendices; you can either adopt the BibTEX way as shown here, or use the `thebibliography` environment directly.

3.6 Author biographies and acknowledgments

The `\abouttheauthors` declaration will typeset the authors' bibliographies from the data given in the preamble. For the camera-ready version, think of adjusting the layout to the quantity of text for each author, by toggling the `nowrap` option in the `\author` declarations.

Following this, you can use the `acknowledgments` environment to mention collaborations, grants, etc.

4 Floats and program code

4.1 Very wide floats

Usually you want figures and tables to occupy their natural width. When a figure is quite large, however, you should prevent it from extending into the margin, by scaling the graphics to the text width:

```
\begin{figure}
    \includegraphics[width=\linewidth]{⟨···⟩}
    \caption{⟨···⟩}
\end{figure}
```

For tables, you can control the table width using the `tabularx` package.

Exceptionally, if a figure or a table really needs to be wider than the text to be legible, you can wrap the graphics or tabular in a `fullwidth` environment, to temporarily reduce the margins. The effect of `fullwidth` is shown in Figure 1.

4.2 Displaying code

The `jot.cls` sets up the `listings` package with generic defaults for simple, clean-looking listings. There are three cases where you might want to display code. The first is to mention program entities in the middle of a sentence; for this, use the `\lstinline`

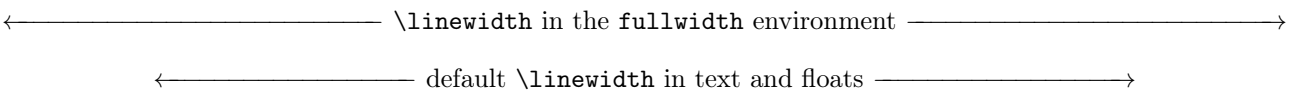


Figure 1 – Changing the available horizontal space using the `fullwidth` environment. Pay attention to not include the caption inside `fullwidth`, else it could produce very long lines in small size, which is difficult to read.

command and associated facilities from `listings`. Note however that this command will not work in some places like float captions, so we advise you to define an alias to the basic font-changing command, like so:

```
\newcommand\code[1]{\texttt{#1}}
```

The second case is to display a large piece of code; this requires a floating listing, which you will get by adding the `style=float` option to either the `lstlisting` environment or the `\lstinputlisting` command. Since this is a float, remember to set the `caption` and `label` options as well. Line numbers are pre-configured, and can be activated by adding `style=numbers` either to the `\lstset` declaration or to the options of individual listings, as needed.

Finally, this should be rarely needed in most articles, but if you need to display short code excerpts in the flow of paragraphs, like most examples in this document, you can just use the regular `lstlisting` environment without special options.

You will probably need some additional configuration to indicate the language of your listings, and e.g. to highlight parts of code; please refer to the documentation of `listings` for more precisions, but keep the number of different text styles to a minimum.[2]

5 Various recommendations, good practices

5.1 Encodings and language

Even though JOT is an English publication, it's best to embrace internationalization, and have the correct `inputenc` and `babel` package declarations in your article. We recommend writing your LaTeX source code in an UTF-8 editor, but other encodings are fine, as long as they are correctly explicited in the document preamble.

```
\usepackage[utf8]{inputenc}
\usepackage[english]{babel}
```

5.2 Referring to sections and floats

The `hyperref` package provides the `\autoref` command, that replaces `\ref` but will automatically format the reference as needed, while making the whole reference a link, instead of just the number. So, instead of typing `see Figure~\ref{foo}` by hand, just use `see \autoref{foo}` instead. This works for other kinds of floats as well.

[2]Highlighting is based on visual contrast and thus relies on scarcity to be effective.

5.3 Better typography

In typography, attention to detail pays, but also visual simplicity and homogeneity. With LaTeX it is often tempting to use different fonts for similar concepts like classes and files, or mathematical properties or propositions. It's best to keep the number of different text styles to a minimum: besides the default text font, `\texttt` and `\emph` should cover most needs.

Tables or graphics with too many lines hamper legibility [Tuf01, chi03]; to help minimize *chartjunk* and maximize *data ink*, `jot.cls` loads the `booktabs` package for you. To take advantage of it:

- do not specify vertical rules in tables;
- use `\toprule`, `\bottomrule`, and few `\middlerule` in between, instead of `\hline`;
- rely on spacing and column alignment to visually separate columns (use `@{\quad}` as a column separator).

The `microtype` package can help LaTeX layout more compact paragraphs with fewer hyphenations.

References

[Bri04] Robert Bringhurst. *The Elements of Typographic Style*. Hartley & Marks, 3.1 edition, 2004.

[chi03] *The Chicago Manual of Style*. The University of Chicago Press, 15th edition, 2003.

[GM04] Michel Goossens and Frank Mittelbach. *The LaTeX Companion*. Addison-Wesley, second edition, 2004.

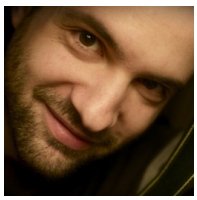
[HK96] Jost Hochilu and Robin Kinross. *Designing books: practice and theory*. Hyphen Press, 1996.

[Knu86] Donald Erwin Knuth. *The TeXbook*. Addison-Wesley, 1986.

[Nie10] Oscar Nierstrasz. Ten things I hate about object-oriented programming. *Journal of Object Technology*, 9(5), September 2010. `doi:10.5381/jot.2010.9.5.e1`.

[Tuf01] Edward R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 2nd edition, 2001.

About the authors



Damien Pollet is an assistant professor at the Université de Lille 1, France.

When he's not busy hacking the LaTeX document class for JOT and maintaining various web servers, he teaches software engineering or does research in the RMoD group, on better constructs and tools for dynamic programming languages, as well as on program visualization and reengineering.

Contact him at damien.pollet@inria.fr, or visit http://people.untyped.org/damien.pollet.

Oscar Nierstrasz is a professor of computer science at the Institute of Computer Science (IAM) of the University of Bern, where he founded the Software Composition Group in 1994.
http://scg.unibe.ch/staff/oscar.

Stéphane Ducasse is a research director at Inria Lille, where he founded the RMoD group in 2007.
http://stephane.ducasse.free.fr.

Lynn V. Siebel is a fictitious author who kindly accepted to demonstrate how the jot.cls class handles authors with multiple affiliations, but whose smile shall remain unseen.

Acknowledgments The style and code of jot.cls was partially inspired from the previous jotarticle.cls developed at ETH Zurich by Susanne Cech, and from the class toc.cls for the *Theory of Computing* journal.