

DAS OSGLISTINGS PAKET

VO.1.0 2026-08-28

Ein auf `minted` aufbauendes Frontend für Code-Listings

Matthias WERNER¹

<https://github.com/werner-matthias/osglecture>

Das Paket `OSGLISTINGS` stellt ein Frontend für `minted` bereit, welches Farben und Rahmenstile leicht konfigurierbar macht und Markierungen, externe Editoren und Codequellen (gists) unterstützt.

Außerdem wird der Quelltext als Code getaggt, falls `\DocumentMetadata{tagging=on}` gesetzt ist.

Das Paket ist Teil des `osglecture`-Bundles, aber von den anderen Paketen weitgehend unabhängig.

Inhaltsverzeichnis

1 Nutzung	1	1.6 Barrierefreiheit	6
1.1 Rahmenstile	2	1.7 Tagging	7
1.2 Farbpalette	2		
1.3 Links in Online-Editoren	3	2 Implementation	7
1.4 Remote-Quellen	3		
1.5 Zeichengenaue Markierungen	4	Index	8

1 Nutzung

Das Paket wird auf dem üblichen Weg mit `\usepackage` geladen und lädt seinerseits (u.a.) `minted`.

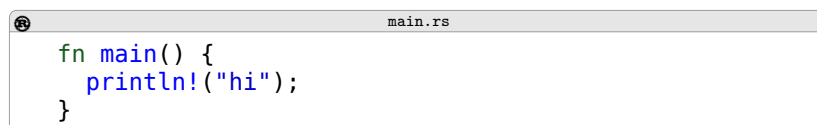
`osglistings` kennt vier Paketoptionen: `cache`, `cachedir`, `frozen` und `cacheignorefilecontents`. Diese werden unverändert an `minted`s eigene, gleichnamige Ladezeit-Optionen weitergereicht und steuern dessen Highlighting-Cache. Ohne Angabe verhält sich `minted` wie mit seinen eigenen Voreinstellungen. Andere `minted`-Paketoptionen lassen sich derzeit nicht über `osglistings` setzen.

¹. matthias.werner@informatik.tu-chemnitz.de

```
1 \usepackage[cachedir=build/minted-cache]{osglistings}
```

Ein Quelltext, der direkt im Dokument steht, wird mit der Umgebung `osglisting` gesetzt; ein Quelltext aus einer externen Datei mit `\osglistinginput`. Beide akzeptieren dieselben Schlüssel, dazu jede von `minted` selbst verstandene Option (`firstline`, `breaklines`, ...), die unverändert durchgereicht wird.

```
1 \documentclass{article}
2 \usepackage{osglistings}
3 \begin{document}
4   \begin{osglisting}[style=editor, title=main.rs]{rust}
5     fn main() {
6       println!("hi");
7     }
8   \end{osglisting}
9 \end{document}
```



```
main.rs
fn main() {
  println!("hi");
}
```

1.1 Rahmenstile

Der Schlüssel `style` wählt zwischen drei Darstellungen: `plain` (ein schlichter Rahmen), `paper` („Druckerpapier“ mit gerissener Kante, ohne Kopfzeile) und `editor` (ein „Editorfenster“ mit Titelzeile, Sprachsymbol und, falls `link` gesetzt ist, einem Bearbeiten-Symbol). Der Stil `paper` unterstützt keinen Seitenumbruch innerhalb der Box.

1.2 Farbpalette

Die Syntaxfarben kommen aus einer benannten Palette, aktiviert mit `\UseOsgListingsPalette` bzw. für einen einzelnen Aufruf mit dem Schlüssel `palette`. Eine eigene Palette wird mit `\DeclareOsgListingsPalette` deklariert; nicht angegebene Slots übernehmen den Wert der eingebauten Standardpalette `osglistings-default`.

```

1 \DeclareOsgListingsPalette{my-theme}{
2   keyword = {model=HTML, value=8A2BE2},
3   string  = {alias=keyword},
4 }
5 \UseOsgListingsPalette{my-theme}

```

Jeder Slot ist entweder eine *Ressource* (model+value, in jedem von [xcolor](#) unterstützten Modell) oder ein *Alias* (alias=<other slot>, höchstens zwei Ebenen tief aufgelöst).

1.3 Links in Online-Editoren

Der Schlüssel `link` hinterlegt eine URL; `link-display` legt fest, wo sie erscheint: `titlebar` (ein Symbol in der Titelzeile, nur für editor), `label` (ein klickbarer Text unterhalb der Box), `qrcode` (ein QR-Code unterhalb der Box) oder `none` (kein sichtbarer Hinweis).

1.4 Remote-Quellen

Das dritte Argument von `\osglistinginput` akzeptiert statt eines lokalen Pfads auch einen `gist:-` oder `github:-`Lokator, z. B. `gist:<owner>/<gist-id>/<filename>` bzw. `github:<owner>/<repo>@<ref>/<path>`. Der Quelltext wird beim ersten Kompilieren geladen und unter `\jobname.osglistings-cache/` gecacht; `cache` steuert die Wiederverwendung: `cache=true` (Voreinstellung) nutzt einen vorhandenen Cache- Eintrag, `cache=refresh` lädt immer neu und aktualisiert ihn.

```

1 \osglistinginput[style=editor, title={fizzbuzz.rs}]
2 {rust}{gist:werner-matthias/3af1e0.../fizzbuzz.rs}

```

Fehlt der Schlüssel `link`, leitet das Paket automatisch einen Editor-Link aus dem Sprachregister ab: aus `editor-url-gist`, falls für die Sprache hinterlegt und die Quelle ein Gist ist (Vorlage typischerweise mit `{GIST_ID}` statt `{CODE}`), sonst aus `editor-url`. Der Platzhalter `{CODE}` wird durch den URL-codierten Quelltext ersetzt, `{GIST_ID}` durch die Gist-ID.

Ein einmal gecachter Eintrag wird beim Kompilieren nicht automatisch gegen die Quelle validiert – `remote-check>manual` (Voreinstellung) prüft gar nicht; `remote-check=compile` löst am Dokumentende genau eine Anfrage pro *unterschiedlichem* Repository/Gist aus (nicht pro Datei) und warnt bei veralteten Einträgen; `remote-refresh=true` aktualisiert veraltete Einträge dabei zusätzlich sofort. Außerhalb von L^AT_EX übernimmt das Kommandozeilenskript [osglistings-check-updates.lua](#) dieselbe Prüfung für einen oder mehrere Cache-Ordner.

```
1 texlua osglistings-check-updates.lua --refresh myfile.osglistings-cache
```

Achtung, Sicherheitsrisiko!

Ein `gist:-` oder `github:-`Lokator benötigt zur Übersetzung `--shell-escape` und ruft dabei intern `curl` mit den Rechten des `TeX`-Prozesses auf. Verwenden Sie diese Lokatoren nur in Dokumenten, deren Inhalt (und deren Lokatoren) Sie vertrauen – ein böswillig gewählter Lokator könnte sonst beliebige Netzwerkanfragen unter Ihrer Identität auslösen.

1.5 Zeichengenaue Markierungen

`\osglistingsmark{name}` legt an der Stelle, an der es im Quelltext steht, einen unsichtbaren, benannten `tikz`-Knoten ab; ein späteres `tikzpicture` kann von oder zu ihm zeichnen. Für Inline-Quelltext (Umgebung `osglisting`) muss dazu `escapeinside=§§` selbst gesetzt werden (z. B. `escapeinside=§§`), damit `\osglistingsmark{name}` überhaupt als `LATEX`-Code statt als Verbatim-Text erkannt wird.

```
1 \documentclass{article}
2 \usepackage{tikz}
3 \usepackage{osglistings}
4 \begin{document}
5   \begin{osglisting}[style=plain, escapeinside=§§]{rust}
6     let code = ExitCode::from(42§\osglistingsmark{here}§);
7   \end{osglisting}
8   \begin{tikzpicture}[remember picture, overlay]
9     \draw[->, red, thick] (osgm-here) ++(0,1) node[above]{Exit-Code}
10      -- (osgm-here);
11 \end{tikzpicture}
12 \end{document}
```

Exit-Code



```
let code = ExitCode::from(42);
```

Für externe Quellen übernimmt der Schlüssel `marks` von `\osglistinginput` dieselbe Aufgabe, ohne dass `escapeinside` manuell gesetzt werden muss – ist `marks` nicht leer, schaltet das Paket `escapeinside` selbst ein (Trennzeichen `$`, änderbar über `marks-escape`). Jeder durch Komma getrennte Eintrag adressiert entweder zeichengenau über `<name>` at `<line>:<col>` (`<col>` zählt in Zeichen, nicht Bytes) oder über eine wörtliche Textsuche mit `<name>` after "`<text>`" bzw. `<name>` before "`<text>`", optional gefolgt von `occurrence=<n>` (Voreinstellung 1).

```

1 \osglistinginput[
2   marks = {
3     exitc after "ExitCode",
4     semi before ";" occurrence=2,
5   }
6 ]{rust}{src/example.rs}

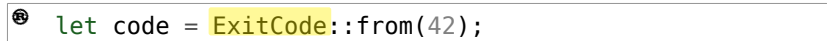
```

`around` verhält sich wie `after/before`, markiert aber keine Einzelstelle, sondern grenzt den gefundenen Text mit einem Knotenpaar `<name>-begin/<name>-end` ein. `\osglistingshighlight` verbindet ein solches Paar anschließend zu einer Markierungsbox – dank der in `\osglistingsmark` mitgemessenen Zeichenhöhe/-tiefe ganz ohne manuelle `xshift/yshift`-Korrekturen. Für Inline-Quelltext lässt sich derselbe Effekt auch ohne `marks` erzielen, indem `\osglistingsmark` einfach zweimal (mit den Namen `<name>-begin/<name>-end`) direkt im Quelltext gesetzt wird.

```

1 \documentclass{article}
2 \usepackage{tikz}
3 \usepackage{osglistings}
4 \begin{document}
5   \begin{osglisting}[style=plain, escapeinside=$$]{rust}
6     let code = $\osglistingsmark{kw-begin}$ExitCode$\osglistingsmark{kw-end}$$::from
7       (42);
8     \end{osglisting}
9     \osglistingshighlight{kw}
10  \end{document}

```



```
let code = ExitCode::from(42);
```

Für externe Quellen genügt dagegen `around` im `marks`-Eintrag, gefolgt von `\osglistingshighlight`.

```
1 \osglistingsinput[
2   marks = { exitc around "ExitCode" }
3 ]{rust}{src/example.rs}
4 \osglistingshighlight{exitc}
```

`marks` (wie jede Nutzung von `\osglistingsmark`) benötigt Lua^AT_EX.

1.6 Barrierefreiheit

Trägt ein `around`-Eintrag zusätzlich `desc="⟨text⟩"`, öffnet die Ersetzung statt eines bloßen Punktpaars eine getaggte `Span`-Struktur mit diesem Text als `alt`-Attribut (Voraussetzung: `\DocumentMetadata{tagging=on}`, sonst bleibt es beim Punktpaar) – assistive Technologien lesen dann `desc="⟨text⟩"` statt des markierten Quelltexts selbst. Für Inline-Quelltext leisten `\osglistingsmarkspanbegin`/`\osglistingsmarkspanend` dasselbe von Hand. `\osglistingshighlight` selbst kapselt sein Overlay dabei automatisch als `Artifact`; assistive Technologien überspringen es, weil die eigentliche Information ja schon über `desc="⟨text⟩"` am Quelltext hängt, nicht am Pfeil oder der Box drumherum.

```
1 \osglistingsinput[
2   marks = {
3     exitc around "ExitCode" desc="the process exit status",
4   }
5 ]{rust}{src/example.rs}
6 \osglistingshighlight{exitc}
```

Nur getaggte Spannen tragen Text

Der übrige Quelltext landet zwar in einer getaggtten Code-Struktur, aber (Stand dieser Implementierung) ohne für Screenreader auslesbaren Text – desc ist für eine markierte Spanne also meist nicht Zusatz, sondern die einzige tatsächlich vorgelesene Information.

Zweiter Kompilierlauf nötig

remember picture löst sich, wie jedes seitenübergreifende tikz-Overlay, erst nach einem zweiten Kompilierlauf korrekt auf – vor dem ersten Lauf (oder nach Änderungen an den markierten Positionen) zeigen Pfeile/Markierungen noch auf ihre vorherige Position oder gar nicht.

1.7 Tagging

Ist L^AT_EX mit `\DocumentMetadata{tagging=on}` gestartet, wird der Quelltext als Struktur Code getaggt (Slotname `osglistings/code`, umsetzbar mit `\NewStructureName/\AssignStructureRole`). Tagging wird pro Aufruf mit `tag=false` abgeschaltet.

2 Implementation

Um die Implementationsbeschreibung zu erhalten, kommentieren Sie bitte `\OnlyDescription` in `osglistings.dtx` aus.

Index

A		O	
\AssignStructureRole	7	\OnlyDescription	7
D		\osglistinginput	2 f., 5
\DeclareOsgListingsPalette	2	\osglistingshighlight	5 f.
\DocumentMetadata	1, 6 f.	\osglistingsmark	4 ff.
J		\osglistingsmarkspanbegin	6
\jobname	3	\osglistingsmarkspanend	6
N		U	
\NewStructureName	7	\UseOsgListingsPalette	2
		\usepackage[<i>(Optionen)</i>]{osglistings}	1