

MANUAL for

THE *jlcode* PACKAGE Version 4.1.1

August 04, 2020

(Version 1.0: January 25, 2018)

<https://github.com/wg030/jlcode>

Copyright 2018–2020 GitHub user wg030

Contents

1	Introduction	2
2	License	2
3	Package Options	3
4	How to Insert Code with <code>autoload=true</code>	4
5	How to Insert Code with <code>autoload=false</code>	5
6	Example with <code>usecolors=true</code> and <code>usebox=true</code>	6
7	Example with <code>nocolors=true</code> and <code>usebox=true</code>	7
8	Example with <code>usecolors=true</code> and <code>nobox=true</code>	8
9	Example with <code>nocolors=true</code> and <code>nobox=true</code>	9
10	Known Managable Issues	10

1 Introduction

The `jlcode` package (*jlcode.sty*) provides a language definition for the programming language Julia for the listings package as well as a style definition. Loading this package is the easiest way to display Julia code within your document when you want to make use of the listings interface.

This package takes especially care of correctly displaying code that contains the most common unicode characters such as greek letters, superscripts or mathematical symbols, which Julia allows as valid characters for identifier names. Moreover all keywords, literals, built-ins, macros and string types that belong to Julia's standard library were generated by a script (*createlists.jl*) and then included in the language definition of this package.

Alongside the actual language definition a nice looking style was defined, too, which will help you to highlight your Julia code with colors and a surrounding box. The style is very similar to the style of the official Julia online documentation, but it can be turned off partly or even completely.

2 License

manual.pdf

Copyright 2018 GitHub user wg030

This work may be distributed and/or modified under the conditions of the L^AT_EX Project Public License, either version 1.3 of this license or (at your option) any later version.

The latest version of this license is in <http://www.latex-project.org/lppl.txt> and version 1.3 or later is part of all distributions of L^AT_EX version 2005/12/01 or later.

This work has the LPPL maintenance status 'maintained'.

The Current Maintainer of this work is GitHub user wg030.

This work consists of the files

jlcode.sty, *createlists.jl*, *testfile1.jl*, *testfile2.jl*, and *manual.pdf*.

3 Package Options

autoload=*(true | false)* **(default: true)**

Load the jlcode style automatically when including the package.

This option is recommended if you just want to display Julia code with the listings package in your document. However if you also want to display code from different programming languages, you should set this option to false because you are likely to experience ugly interferences otherwise.

courierastdflt=*(true | false)* **(default: true)**

Use Courier as the typewriter font in your document.

This option is recommended unless you want to have a different font for your code examples. Note that only the typewriter family is affected here and you must load any font package before the jlcode package since otherwise there will be no effect with this option.

Moreover if you set this option to *false*, make sure your font supports bold typewriter font, like Courier does, because otherwise the keywords will not be displayed in bold, of course.

usecolors=*(true | false)* **(default: true)**

Activate the coloring of keywords, literals, built-ins and macros.

Use this option if you want important aspects of your code to be highlighted with some nice colors.

nocolors=*(true | false)* **(default: false)**

The opposite of usecolors. Deactivate the coloring of keywords, literals, built-ins and macros.

Use this option if you want all parts of your code to be printed in black.

usebox=*(true | false)* **(default: true)**

Activate the code box.

Use this option if you want to display a light grey code box.

nobox=*(true | false)* **(default: false)**

The opposite of usebox. Deactivate the code box.

Use this option if you want your code to be displayed without a code box.

charsperline=*(positive integer)* **(default: 80)**

Control the width of code box.

Use this option in order to specify the exact number of characters per line that can fit into the codebox. If you use the nobox option, this option is without any effect.

4 How to Insert Code with `autoload=true`

Command for Loading the Package:

```
\usepackage[autoload=true]{jlcodel}
```

In-line Code Snippets:

```
\jlinl{@time sort(myarr) # no modification}
```

Display Code:

```
\begin{lstlisting}  
# some julia code  
println( "Here we go with Julia!")  
\end{lstlisting}
```

Listings for Standalone Files:

```
\lstinputlisting{filename.jl}
```

5 How to Insert Code with `autoload=false`

Command for Loading the Package:

```
\usepackage[autoload=false]{jlcode}
```

In-line Code Snippets:

```
\jlinl{@time sort(myarr) # no modification}
```

Display Code:

```
\begin{lstlisting}[language=julia , style=jlcodestyle]  
# some julia code  
println( "Here we go with Julia!")  
\end{lstlisting}
```

Listings for Standalone Files:

```
\lstinputlisting[language=julia , style=jlcodestyle]{filename.jl}
```

6 Example with usecolors=true and usebox=true

Command for Loading the Package:

```
\usepackage[usecolors,usebox]{jlcode}
```

Output of a Useless but Diverse Code Example:

```
#= A comment that consists of several lines.
The following code itself is rather useless unless you want
to test how Julia code is displayed by the jlcode package. =#

# This line will be my reference line, which will contain exactly 80 characters.

# This line is a comment containing operators like &, -, $ and %
# A comment with the German word "Übergrößengeschäft" (store for oversizes)
# This line contains some special unicode characters: €, α, γ, w², Δₓ, Ĩ, x̄, e
# A comment with some numbers: 424, 1.23, 0.2E-5, -9.9e+9, 1_001
# Mathematical characters that are Julia functions:
# |, |>, ~, x, ÷, ∈, ∉, ∃, ∄, ∅, √, ∛, ∩, ∪,
# ≈, ≠, ≡, ≠, ≤, ≥, ⊆, ⊇, ⊂, ⊃, ⊄, ⊅, ∖, ∙
# Other mathematical symbols: ∇, ⊗, ⊕, ∥, ..., ∴, ∵, ∴

# defining a useless testfunction
function Style_4th_Test(x, y)

    myver = v"2.00"
    mystr = "String: \"Übergrößengeschäft\", α, π, ∪, φ and the + operator."
    myset = Set{ [2, 9, 1_200, 2_500, 33]}
    x_in_myset = x ∈ myset
    myset{2} = myset ∪ Set{ [4, 8_000, 12, 33]}
    z1vec = rand{Int8, 3}
    z2vec = Array{Int8}(undef, 3)
    z2vec[1:2] = [x % y, y \ x]
    t = x % 2 == 0 ? x : x + 1
    t̃ = ~(t & x | y) ∨ y
    myβvar = t̃ & t $ t
    α = @time √0.3
    βᵅ = 3.2e+5^α
    myβvar = √0.12E-2 * βᵅ
    z2vec[3] = y^2 + 3.4x*y - (α + myβvar) * t/2
    z2vec = (z2vec + z1vec).^2
    if !(0.1 ≤ norm(z1vec') < norm(z2vec') + e + e ÷ pi + γ + φ)
        mystr = String( mystr, " signed ")
        println( mystr)
        return true;
    elseif 3.2 ≥ norm(z2vec - z1vec) > 2.69
        if norm(z2vec - z1vec) ≠ 3.0
            println( String( "Error in ", myver, "!"))
        end
        return false;
    end
end

end
```

7 Example with nocolors=true and usebox=true

Command for Loading the Package:

```
\usepackage[nocolors,usebox]{jlcode}
```

Output of a Useless but Diverse Code Example:

```
#= A comment that consists of several lines.
The following code itself is rather useless unless you want
to test how Julia code is displayed by the jlcode package. =#

# This line will be my reference line, which will contain exactly 80 characters.

# This line is a comment containing operators like &, -, $ and %
# A comment with the German word "Übergrößengeschäft" (store for oversizes)
# This line contains some special unicode characters: €, α, γ, w², Δₓ, Ĩ, x̃, e
# A comment with some numbers: 424, 1.23, 0.2E-5, -9.9e+9, 1_001
# Mathematical characters that are Julia functions:
# |, |>, ~, x, ÷, ∈, ∉, ∃, ∄, ∅, √, ∛, ∩, ∪,
# ≈, ≠, ≡, ≠, ≤, ≥, ⊆, ⊇, ⊂, ⊃, ⊄, ⊅, ∖, ∙
# Other mathematical symbols: ∇, ⊗, ⊕, ∥, ..., ∴, ∵, ∴

# defining a useless testfunction
function Style_4th_Test(x, y)

    myver = v"2.00"
    mystr = "String: \"Übergrößengeschäft\", α, π, ∪, φ and the + operator."
    myset = Set{ [2, 9, 1_200, 2_500, 33]}
    x_in_myset = x ∈ myset
    myset{2} = myset ∪ Set{ [4, 8_000, 12, 33]}
    z1vec = rand{Int8, 3}
    z2vec = Array{Int8}(undef, 3)
    z2vec[1:2] = [x % y, y \ x]
    t = x % 2 == 0 ? x : x + 1
    t̃ = ~(t & x | y) ∨ y
    myβvar = t̃ & t $ t
    α = @time √0.3
    βᵅ = 3.2e+5^α
    myβvar = ∛0.12E-2 * βᵅ
    z2vec[3] = y^2 + 3.4x*y - (α + myβvar) * t/2
    z2vec = (z2vec + z1vec).^2
    if !(0.1 ≤ norm(z1vec') < norm(z2vec') + e + e ÷ pi + γ + φ)
        mystr = String( mystr, " signed ")
        println( mystr)
        return true;
    elseif 3.2 ≥ norm(z2vec - z1vec) > 2.69
        if norm(z2vec - z1vec) ≠ 3.0
            println( String( "Error in ", myver, "!"))
        end
        return false;
    end
end

end
```

8 Example with usecolors=true and nobox=true

Command for Loading the Package:

```
\usepackage[usecolors,nobox]{jlcode}
```

Output of a Useless but Diverse Code Example:

```
#= A comment that consists of several lines.
The following code itself is rather useless unless you want
to test how Julia code is displayed by the jlcode package. =#

# This line will be my reference line, which will contain exactly 80 characters.

# This line is a comment containing operators like &, -, $ and %
# A comment with the German word "Übergrößengeschäft" (store for oversizes)
# This line contains some special unicode characters: €, α, γ, w², Δₓ, Ÿ, x̄, e
# A comment with some numbers: 424, 1.23, 0.2E-5, -9.9e+9, 1_001
# Mathematical characters that are Julia functions:
# |, |>, ~, ×, ÷, ∈, ∉, ∃, ∄, ∅, √, ∛, ∩, ∪,
# ≈, ≅, ≠, ≡, ≢, ≤, ≥, ⊆, ⊇, ⊂, ⊃, ⊄, ⊅, ∖, ·
# Other mathematical symbols: ∇, ⊗, ⊕, ∥, ..., ∴, ∵, ∴
# defining a useless testfunction
function Style_4th_Test(x, y)

    myver = v"2.00"
    mystr = "String: \"Übergrößengeschäft\", α, π, ∪, φ and the + operator."
    myset = Set{ [2, 9, 1_200, 2_500, 33]}
    x_in_myset = x ∈ myset
    myset^(2) = myset ∪ Set{ [4, 8_000, 12, 33]}
    z1vec = rand{Int8, 3}
    z2vec = Array{Int8}{undef, 3}
    z2vec[1:2] = [x % y, y \ x]
    t = x % 2 == 0 ? x : x + 1
    t̃ = ~(t & x | y) ∨ y
    myβvar = t̃ & t $ t
    α = @time √0.3
    β^α = 3.2e+5^α
    myβvar = ∛0.12E-2 * β^α
    z2vec[3] = y^2 + 3.4x*y - (α + myβvar) * t/2
    z2vec = (z2vec + z1vec).^2
    if !(0.1 ≤ norm(z1vec') < norm(z2vec') + e + e ÷ pi + γ + φ)
        mystr = String( mystr, " signed ")
        println( mystr)
        return true;
    elseif 3.2 ≥ norm(z2vec - z1vec) > 2.69
        if norm(z2vec - z1vec) ≠ 3.0
            println( String( "Error in ", myver, "!"))
        end
        return false;
    end
end

end
```


9 Example with nocolors=true and nobox=true

Command for Loading the Package:

```
\usepackage[nocolors,nobox]{jlcode}
```

Output of a Useless but Diverse Code Example:

```
#= A comment that consists of several lines.
The following code itself is rather useless unless you want
to test how Julia code is displayed by the jlcode package. =#

# This line will be my reference line, which will contain exactly 80 characters.

# This line is a comment containing operators like &, -, $ and %
# A comment with the German word "Übergrößengeschäft" (store for oversizes)
# This line contains some special unicode characters: €, α, γ, w², Δₓ, Ÿ, x̃, e
# A comment with some numbers: 424, 1.23, 0.2E-5, -9.9e+9, 1_001
# Mathematical characters that are Julia functions:
# |, |>, ~, ×, ÷, ∈, ∉, ∃, ∄, ∅, √, ∛, ∩, ∪,
# ≈, ≅, ≠, ≡, ≢, ≤, ≥, ⊆, ⊇, ⊂, ⊃, ⊄, ⊅, ∖, ·
# Other mathematical symbols: ∇, ⊗, ⊕, ∥, ..., ∴, ∵, ∴
# defining a useless testfunction
function Style_4th_Test(x, y)

    myver = v"2.00"
    mystr = "String: \"Übergrößengeschäft\", α, π, ∪, φ and the + operator."
    myset = Set{ [2, 9, 1_200, 2_500, 33]}
    x_in_myset = x ∈ myset
    myset^(2) = myset ∪ Set{ [4, 8_000, 12, 33]}
    z1vec = rand{Int8, 3}
    z2vec = Array{Int8}(undef, 3)
    z2vec[1:2] = [x % y, y \ x]
    t = x % 2 == 0 ? x : x + 1
    t̃ = ~(t & x | y) ∨ y
    myβvar = t̃ & t $ t
    α = @time √0.3
    β^α = 3.2e+5^α
    myβvar = ∛0.12E-2 * β^α
    z2vec[3] = y^2 + 3.4x*y - (α + myβvar) * t/2
    z2vec = (z2vec + z1vec).^2
    if !(0.1 ≤ norm(z1vec') < norm(z2vec') + e + e ÷ pi + γ + φ)
        mystr = String( mystr, " signed ")
        println( mystr)
        return true;
    elseif 3.2 ≥ norm(z2vec - z1vec) > 2.69
        if norm(z2vec - z1vec) ≠ 3.0
            println( String( "Error in ", myver, "!"))
        end
        return false;
    end
end

end
```

10 Known Managable Issues

The following Julia code presents the known issues that can appear due to the nature of the listings package. Right now the jlcode package is not able to handle theses cases. However the issues can be fixed by the user himself.

Output Without Fixing the Issues:

```
function KnownManageableIssues(x, y)

    # KNOWN MANAGEABLE ISSUES:

    # identifier name with a number that follows
    # directly behind a special unicode character:
    myβ2ndvar = 2 * 0.12E-2 * xy

    # identifier name, which contains a γ, π or φ:
    myφvar+ = sqrt(2)
    approx4π = 3.142

    # numbers in E-notation without using a + sign:
    evar = 3.99e400
    evar2 = 3.99E400

    # single characters
    mychar = 'W'
    mychar(2) = '€'
    mychar(3) = 'κ'

end
```

Commands for Fixing the Issues:

```
\addlitjlbase{myβ2ndvar}{my$\beta$2ndvar}{9}
\addlitjlbase{myφvar+}{my$\phi$var$\scriptstyle {}_{+}}{7}
\addlitjlbase{approx4π}{approx4$\pi$}{8}
\addlitjlbase{löwe}{l{"o}we}{4}
\addlitjlbase{checkin€}{checkin\euro}{8}
\addlitjlstring{e400}{e400}{4}
\addlitjlstring{E400}{E400}{4}
\addlitjlstring{'W'}{\textquotesingle W\textquotesingle}{3}
\addlitjlstring{'€'}{\textquotesingle \euro\textquotesingle}{3}
\addlitjlstring{'κ'}{\textquotesingle $\varkappa$\textquotesingle}{3}
```

Output After Fixing the Issues:

```
function KnownManageableIssues(x, y)

    # KNOWN MANAGEABLE ISSUES:

    # identifier name with a number that follows
    # directly behind a special unicode character:
    my $\beta$ 2ndvar = 2 * 0.12E-2 * xy

    # identifier name, which contains a  $\gamma$ ,  $\pi$  or  $\phi$ :
    my $\phi$ var+ = sqrt(2)
    approx4 $\pi$  = 3.142

    # numbers in E-notation without using a + sign:
    evar = 3.99e400
    evar2 = 3.99E400

    # single characters
    mychar = 'W'
    mychar(2) = '€'
    mychar(3) = '⌘'

end
```