

The REPL and Unicode

Keybindings, help mode, how to type Greek

Modes

Key at an empty prompt	Mode	Prompt
—	Julia	julia>
]	Pkg	(env) pkg>
?	Help	help?>
;	Shell	shell>
Backspace	back to Julia	

```
julia> ?sort           # docs, all methods, related functions
julia> ]st             # one-off Pkg command without staying in the mode
julia> ;ls -la         # one-off shell command
```

Help mode also works on operators, macros, types and unicode characters — `?*`, `?@view`, `?Vector`, and pasting a character tells you its name and how to type it.

Keybindings

Key	Does
Ctrl-C	interrupt the running expression
Ctrl-D	exit (at an empty prompt)
Ctrl-L	clear screen
Ctrl-R / Ctrl-S	reverse / forward history search
Up/Down	history, prefix-filtered by what you've typed
Alt-Enter	insert a newline without evaluating
Ctrl-Q	edit the current line in <code>\$EDITOR</code>
Tab	completion — of names, paths, fields, and LaTeX sequences
Tab twice	show all completions, including method signatures

Typing `f(` then Tab shows the applicable methods. Typing `x.` then Tab shows fields and properties.

Useful REPL functions

```
ans                # value of the last expression
varinfo()          # what's defined in Main, with sizes
varinfo(MyPkg)
clipboard("text"); clipboard()    # copy out / paste in
edit("file.jl", 10)              # open in $JULIA_EDITOR at a line
@edit f(1)                      # open the source of the method that call selects
@which f(1)
@doc f
names(Base; all = false)
methods(f); methodswith(MyType)
apropos("sort")                 # full-text search of docstrings
InteractiveUtils.versioninfo()
```

startup.jl

`~/.julia/config/startup.jl` runs before every session. Keep it small and fast.

```

try
    using Revise
catch e
    @warn "Revise failed to load" exception = (e, catch_backtrace())
end

atreplinit() do repl
    try
        @eval using OhMyREPL
    catch
    end
end

ENV["JULIA_EDITOR"] = "zed --wait"

```

Install these into the shared default environment, not into your projects:

```

julia -e 'using Pkg;
    Pkg.activate("v$(VERSION.major).$(VERSION.minor)"; shared=true);
    Pkg.add(["Revise", "OhMyREPL", "BenchmarkTools"])'

```

--startup-file=no skips it — useful for benchmarking and required for reproducible CI.

Command-line flags worth remembering

```

julia --project           # activate the nearest Project.toml
julia -t auto             # all available threads
julia -p 4                # 4 worker processes (Distributed)
julia -e 'code'           # evaluate and exit
julia -E 'code'           # evaluate, print the result, exit
julia -i script.jl        # run then stay interactive
julia --check-bounds=yes  # bounds-check even where @inbounds says not to
julia --startup-file=no
julia --banner=no -q      # quiet
julia --track-allocation=user # write .mem files with per-line allocation counts
julia --code-coverage=user

```

Unicode input

Type the LaTeX name and press Tab. Works in the REPL, and in editors with the Julia extension.

Sequence	Char	Sequence	Char
<code>\alpha</code>	α	<code>\le</code>	$\leq$
<code>\beta</code>	β	<code>\ge</code>	$\geq$
<code>\gamma</code>	γ	<code>\ne</code>	$\neq$
<code>\delta</code>	δ	<code>\approx</code>	$\approx$
<code>\Delta</code>	Δ	<code>\equiv</code>	$\equiv$
<code>\epsilon</code>	ϵ	<code>\in</code>	$\in$
<code>\theta</code>	θ	<code>\notin</code>	$\notin$
<code>\lambda</code>	λ	<code>\subseteq</code>	$\subseteq$
<code>\mu</code>	μ	<code>\cup</code>	$\cup$
<code>\pi</code>	π	<code>\cap</code>	$\cap$
<code>\rho</code>	ρ	<code>\cdot</code>	$\cdot$
<code>\sigma</code>	σ	<code>\times</code>	$\times$
<code>\Sigma</code>	Σ	<code>\div</code>	$\div$
<code>\phi</code>	ϕ	<code>\circ</code>	$\circ$
<code>\omega</code>	ω	<code>\pm</code>	$\pm$
<code>\Omega</code>	Ω	<code>\sqrt</code>	$\sqrt{}$
<code>\infty</code>	∞	<code>\sum</code>	$\sum$
<code>\partial</code>	∂	<code>\prod</code>	$\prod$
<code>\nabla</code>	∇	<code>\int</code>	$\int$
<code>\hbar</code>	$\hbar$	<code>\emptyset</code>	$\emptyset$
<code>\mathrm{e}</code>	e	<code>\mathrm{im}</code>	im (im is the literal)

Sub- and superscripts: `x\_1` gives x_1 , `x^2` gives x^2 , `\_a` through `\_z` and `\^0` through `\^9` mostly exist. These are legal in identifiers, so σ^2 , x_1 , Δt are all valid variable names.

Built-in constants: π (also `pi`), e (also `exp(1)`), im (imaginary unit), `Inf`, `NaN`, `missing`, `nothing`.

Operators you can define yourself, because they're parsed as infix: $\otimes$, $\odot$, $\oslash$, $\star$, $\circ$, $\pm$, $\mp$, $\times$, $\cdot$, $\cup$, $\cap$, $\subseteq$, $\supseteq$, $\approx$, $\neq$, $\equiv$ and many more. Precedence follows the ASCII operator each resembles.

```
 $\otimes(a, b) = a + 2b$ 
3  $\otimes$  4          # 11
```

If you don't want Unicode in your source, every operator has an ASCII spelling: `in`, `issubset`, `isapprox`, `!=`, `<=`, `div`, `dot`, `cross`, `union`, `intersect`, `pi`, `sqrt`.

Finding out what a character is

Paste it into help mode:

```
help?>  $\otimes$ 
" $\otimes$ " can be typed by \otimes<tab>
```

That works for any character, which is the fastest way to read unfamiliar numerical code.