# Julia Fast scientific language

## CONCEPTS

Julia compiles each method on first call, so the first run is slow and later calls are fast. Keep one REPL open.

**Generic function** name: area
↓ call with args
**Dispatch** all argument types
↓ first call
**Method** specialized, compiled
↓ cached
**Fast native code** reused after

## COMMAND LINE

**julia**
Start the interactive REPL

**julia** *<file.jl> <args>*
Run a script; args land in ARGS

**julia -e '***<expr>***'**
Evaluate an expression and exit

**julia --project=***<dir>*
Activate the environment in a folder

**julia -t** *<n|auto>*
Set the number of threads

**julia -i** *<file.jl>*
Run a file, then stay in the REPL

## REPL

**?**
Help mode: docs for a name

**]**
Package mode

**;**
Shell mode: run a shell command

**Backspace**
Leave a special mode

**Ctrl+D / exit()**
Quit Julia

**Ctrl+C**
Interrupt running code

## REPL (cont.)

**@time** *<expr>*
Time an expression and show allocations

**using** *<Pkg>*
Load a package's exports

## PACKAGES

**add** *<pkg>*
Add a package to the environment

**activate** *<dir>*
Switch to the environment in a folder

## BASICS

**println("x = $x")**
Print with interpolation

**for i in 1:10 ... end**
Loop over a range

**function f(x, y=2; k=1) ... end**
Function with default and keyword args

## ARRAYS

**[1, 2, 3]**
Vector literal

**[1 2; 3 4]**
Matrix literal

**A[i, j]**
Element at row i, column j

**push!(v, x)**
Append to a vector

**[f(x) for x in v if** *<cond>***]**
Comprehension

**f.(v)**
Broadcast f over elements

## TYPES & DISPATCH

**struct P x::Float64; y::Float64 end**
Immutable composite type

**area(s::Circle) = pi * s.r^2**
Method for one type