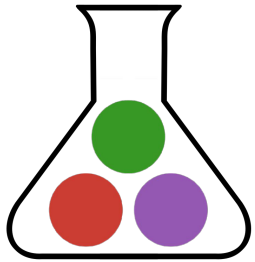
# JuliaLab



https://julia.mit.edu/

Valentin Churavy
vchuravy@mit.edu


# What/Who is the JuliaLab

- PI: Alan Edelman
- Research on/with Julia
- Maintenance and development of Julia in collaboration with the open-source community

Research questions:

- Parallel Computing (GPU/Distributed/Shmem)
- Language runtime/compiler
- Automatic differentiation
- Sparse & Structured Linear Algebra

# What makes a language dynamic?

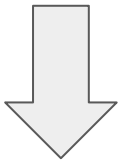
- Commonly: Referring to the type system.
  - **Static:** Types are checked before run-time
  - **Dynamic:** Types are checked on the fly, during execution
  - Also: The type of a **variable** can change during execution
- Closed-world vs open-world semantics
  - The presence of **eval** (Can code be “added” at runtime)
- Struct layout
  - Can one change the fields of a object/class/struct at runtime?

```
x = true
if cond
  x = "String"
end
@show x
```

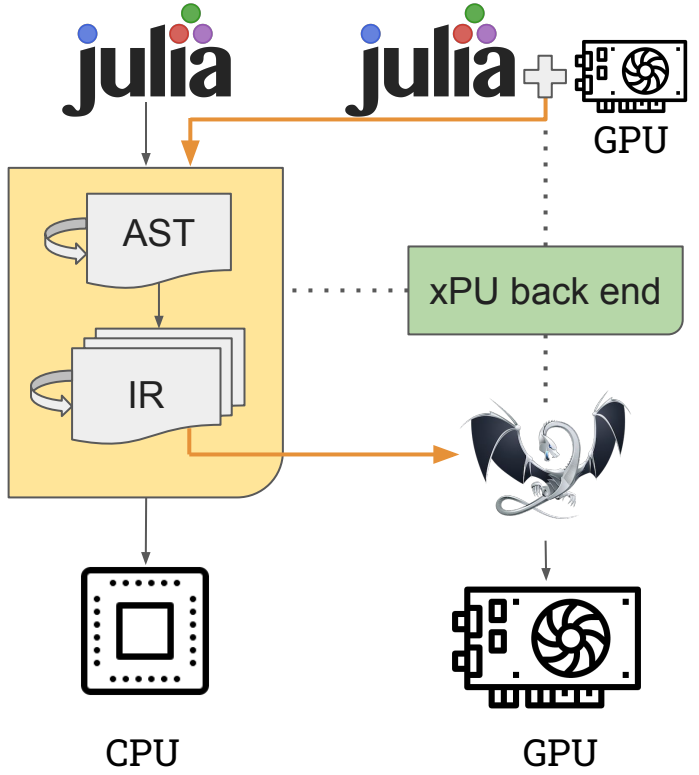
Dynamic semantics are a **spectrum**:
Julia has a dynamic type system and open-world semantics,
but struct layout is static.

# julia gets its Power from Extensible Compiler Design

Language design



Efficient execution




*Julia: Dynamism and Performance Reconciled by Design (doi:10.1145/3276490)*

*Effective Extensible Programming: Unleashing Julia on GPUs (doi:10.1109/TPDS.2018.2872064)*

# Magic of Julia

Abstraction, Specialization, and Multiple Dispatch

Did I really need to move memory for that transpose?

1. **Abstraction** to obtain generic behavior:

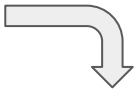
   Encode behavior in the type domain:

```
transpose(A::Matrix{Float64})::Transpose{Float64,Matrix{Float64}}
```

2. **Specialization** of functions to produce optimal code

3. **Multiple-dispatch** to select optimized behavior

```
rand(N, M) * rand(K, M)'
```



compiles to

```
Matrix * Transpose{Matrix}
function mul!(C::Matrix{T}, A::Matrix{T}, tB::Transpose{<:Matrix{T}}, a, b) where {T<:BlasFloat}
    gemm_wrapper!(C, 'N', 'T', A, B, MulAddMul(a, b))
end
```

No I did not! I know $AB^T$ is the dot product of every row of A with every row of B .