


# Documentation in Julia

Adin Schmahmann

`adin@mit.edu`

# Why Documentation?



How do I add an item to the end of a list?

## Current Solution

- Hope it's like other languages
  - Even the languages Julia is based on have different syntax
- Google “Julia add item to list”
  - Not enough Julia users
- For core functionality read online documentation
- Search through source code

# Julia Documentation

- Allow each function, macro, etc. to have documentation describing what they do
- Allow interfaces to be defined
  - Ex: Define the list of functions required to be compatible with an abstract type
- Create functions that allow searching of the documentation by types, text, etc.

# Help Documentation

```
@doc begin
    "Add element to end of array."
end function push!(arr::Array, item)
    return push(arr, item)
end
```

```
@doc begin
    "Add element to end of tuple."
end function push!(tup::Tuple, item)
    return tuple(tup..., item)
end
```

```
julia> doc(push!)
push!(Array{T,N},::item) : Add element to end of array.
push!((Any...),::item) : Add element to end of tuple.
```

# Interfaces

- Define the behaviors a type should have
  - Ex: All Streams should be able to be read from and written to
- Localize the definitions of the different behaviors a class should have
- Ability to write code for an interface that is type independent
  - Ex: A function that will print an entire stream to the console
- Since abstract types have no implementations they could all be interfaces
  - This effectively changes the inherits `<:` symbol to mean implements

```
abstract String
```

```
## core string functions ##
```

```
length(s::String) = error("you must implement length(", typeof(s), ")")  
next(s::String, i::Int) = error("you must implement next(", typeof(s), ", Int)")  
next(s::String, i::Integer) = next(s, int(i))
```

```
length(s::UTF8String) = length(s.data)  
strlen(s::UTF8String) = ccall(:u8_strlen, Int, (Ptr{UInt8},), s.data)
```

```
type UTF8String <: String  
    data::Array{UInt8,1}  
end
```

```
function next(s::UTF8String, i::Int)  
    if !is_utf8_start(s.data[i])  
        error("invalid UTF-8 character index")  
    end  
    trailing = _jl_utf8_trailing[s.data[i]+1]  
    if length(s.data) < i + trailing  
        error("premature end of UTF-8 data")  
    end  
    c = uint32(0)  
    for j = 1:trailing  
        c += s.data[i]  
        c <=<= 6  
        i += 1  
    end  
    c += s.data[i]  
    i += 1  
    c -= _jl_utf8_offset[trailing+1]  
    char(c), i  
end
```

```
abstract String
```

```
@interface String begin  
  next(s::*, i::Int)  
  length(s::*)  
end
```

```
type UTF8String <: String  
  data::Array{UInt8,1}  
end
```

```
function next(s::UTF8String, i::Int)  
  if !is_utf8_start(s.data[i])  
    error("invalid UTF-8 character index")  
  end  
  trailing = _jl_utf8_trailing[s.data[i]+1]  
  if length(s.data) < i + trailing  
    error("premature end of UTF-8 data")  
  end  
  c = uint32(0)  
  for j = 1:trailing  
    c += s.data[i]  
    c <<= 6  
    i += 1  
  end  
  c += s.data[i]  
  i += 1  
  c -= _jl_utf8_offset[trailing+1]  
  char(c), i  
end
```

```
length(s::UTF8String) = length(s.data)  
strlen(s::UTF8String) = ccall(:u8_strlen, Int, (Ptr{UInt8},), s.data)
```

```
@implements String UTF8String
```

# Searching Documentation

- Given all the metadata now available allow searching
  - Based on type/method signature:
    - `findFunctionsWithSig((Int,String,Array{Float,2}))`
    - `findFunctionsWithType(Int64)`
  - Based on documentation
    - `searchDocs("SVD")`
  - What functions are needed in an interface?
    - `Interface("String")`



# Future Work

- Determine what programmers need most when searching documentation
- Build interfaces and documentation into Julia
- IDE integration
  - Autocomplete interfaces
  - Code completion for functions and objects

# Questions?