Conical

Conical is a work-in-progress systems language that uses set-theoretic principals to make most invariants unrepresentable. It will also use scope based lifetime resolution to handle resource management.

There are currently 3 operating principles that are being used in the design process:

1. If a specific feature can be implemented using a more general one, the general one should be used.
2. Its better for something to have more granular control with incremental defaults than have it be based on a assumed use-case.
3. The compiler is much better at enforcing assumptions than humans are.

Examples of each principle, in-order, are:

1. Since async functions are essentially just generators + global state, and generators are just compiler-led transformation of a function into a state-machine, both of these should be implemented using macros instead of being a builtin compiler feature.
2. There is no "unsafe" keyword in Conical that just turns off all safety checks. Rather safety should be implemented using the type system; so only the safety mechanisms that are required are used. This makes safety much more granular and controlled than rust which has a single safe/unsafe distinction.
3. If you have a number that is required to be in a non-standard range (e.g. 5-370), in most languages you would have to rely on runtime checks and documentation, in Conical you can exactly define every value allowed and - except in things like external functions or intrinsics - that assumption is guaranteed.
    - This also means that Conical could be partially formally provable.

Documentation

Currently there is no official documentation for the language as it is still very early in design and implementation. However, there is a design ideas / general syntax overview here (N.B. Everything in that document is subject to change).