# Syntax Philosophy

## Operators should always mean the same thing.

In C#, the “(“ character can represent order of operations, casting, function recipes, or any other number of things.

To help with this, function recipes in nala will be written using the “<” character. This is because in C#, generics are used with the “<” character in a similar way to function recipes, which works well.

*C#:*

void main(string[] args)  
{

}

nala:

void main<string[] args>  
{

}

## The developer should have more control over extensions and interfaces.

### Extending static classes

In C#, the developer is limited to non-static classes if he wants to extend a class. In nala, the developer should be able to extend static classes.

Requiring properties in interfaces.  
In C#, the developer can not place properties in an interface and can not require that an implementation of an interface have those properties. In nala, the developer should have that option.

Better syntax for virtual and abstract.  
In C#, it’s not readily clear for new users the distinction between virtual and abstract. In nala, classes which cannot be initialized should be identified by template, rather than abstract class.

C#:

abstract class Object  
{

}

nala:

template Object  
{

}

Classes are custom types  
And as such should be referred to not as classes at all, but as type instances.

C#:

class Object  
{

}

nala:

type Object  
{

}

## Functions should be easy to write and should make sense.

### Return types should not be separate keywords.

It does not makes sense to say a function *is* a boolean. It *is* a function, it *returns* a boolean.

C#:

int DoThing(string thing)  
{

}

nala?:

func DoThing(string thing : int)  
{

}

Now that the keyword refers to what it is, not what it returns, func can be replaced dependent on override context.

C#:

virtual void DoThing()  
{

}

override int DoThing()  
{  
  
}

abstract void DoThing();

nala?:

virtual DoThing()  
{

}  
  
override DoThing(:int)  
{

}

template DoThing();

Operators should be able to be used as literals.

Arithmetic and relational operators should be valid literals to store as values. The following code (or something with the same philosophy) should be valid Nala:

operator op = +;  
int a = 5;  
int b = 8;  
echo op(a,

The above code should write out “13”.