**What works well**

Namespaces – work like normal namespaces - done

Modules – Work as Static classes - done

Values – static field on a static class(mutables allowed) - done

Functions (kind of - FSharpFunc) - done

Records :- Equatable class with All Args constructor, readonly properties - done

Discriminated Unions (if values are ints) :- Compiles to an enum

Discriminated Unions (with types):- get function NewX based on case name

Unit:- if returned uses void, if used as parameter, use null

Sequences :- Goes to IEnumerable - done

Arrays - done

**Medium working**

Tuples – needs to be marked in F# as struct if using C# tuple syntax that’s not Tuple.create().

Discriminated Unions(no values) – Need to use switch to match on tags

Options:- Get a value, if None is equivalent to null and so can still fail.

Lists:- Not the same as C# list, implements IReadOnlyCollection (F# 4.1 , March 2017)

**What doesn’t work well**

Higher Order Functions :- Func takes Func:

FSharpFunc<int, int>.FromConverter(input => input \* 2)