**What works well**

Namespaces – work like normal namespaces – done – blog done

Modules – Work as Static classes – done – blog done

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

Functions (kind of - FSharpFunc) – done – blog done

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

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

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

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

Sequences :- Goes to IEnumerable – done – blog done

Arrays – done – blog done

**Medium working**

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

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

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

**What doesn’t work well**

Higher Order Functions :- Func takes Func:

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