A YEAR IN PARADISE. FULL STACK F# FOR SYNTHETIC BIOLOGY

THU 27TH - 12:20 PM (TALK)

We are now about one year into implementing a largely F# environment for designing, building and analyzing engineered microorganisms. This is cloud deployed full stack F# (Fable, React, Elmish, Giraffe, Postgres) with algorithms for designing DNA, and web based systems for controlling robots and managing a lab. I will talk about what has worked well and where there are rough spots. I will touch on some areas of biology but it will be mostly about implementing a system for managing a complex manufacturing environment that is widely applicable to many industries.

DEMETRIX

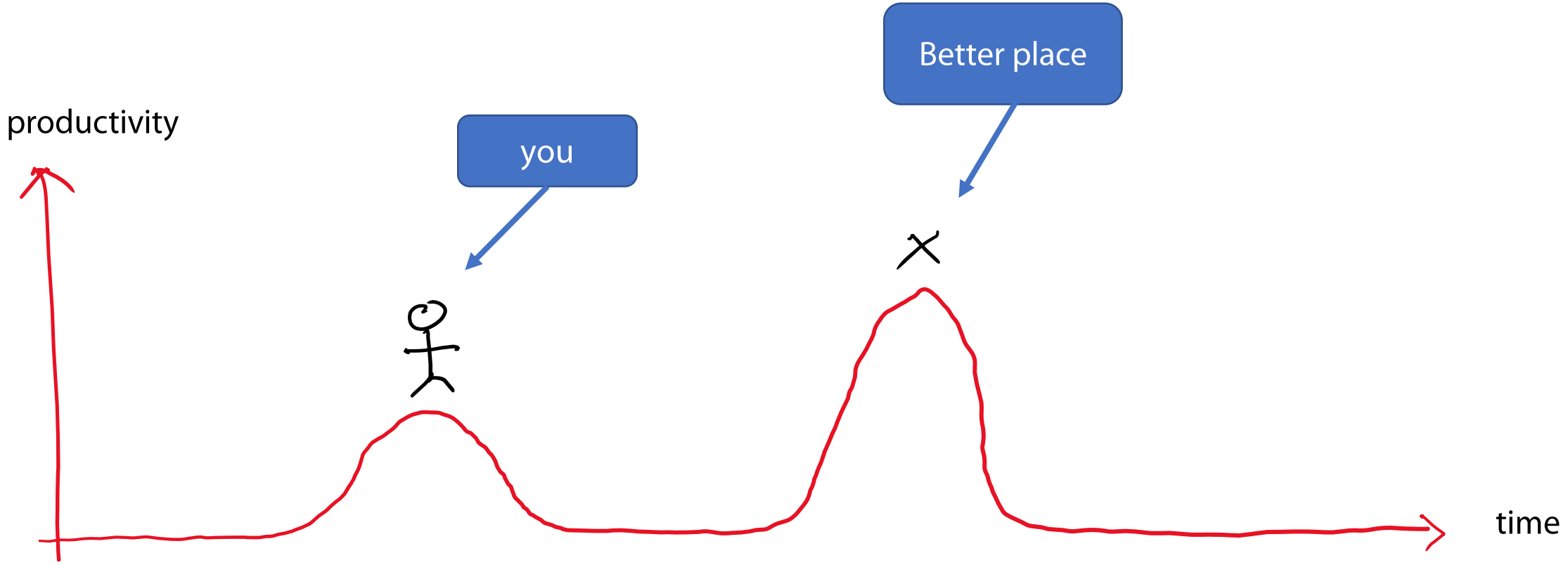
Open F# 20180927

Darren Platt
@dplattsf

Thanks!

- Developers
  - Garrett Birkel
  - Dmitry Morozov
  - Gien Verschatse
- Community
  - Alfonso Garcia [Fable and solid advice]
  - Don Syme [Type provider support]
  - Postgres Npgsql maintainers [improvements in type safety]
  - Dustin Moris Gorski [Giraffe]
  - Jeffrey Rennie [Google Cloud Platform]
  - Elliot Menschik, Paul Underwood [AWS Biotech support]
  - Krzysztof-Cieslak [Ionide]
  - Matthias Brandewinder [Recruiting]
  - Chris Macklin [Genotype Specification Language collaboration]
  - Open Fsharp
  - Microsoft F# team
  - A ton of people I have missed..

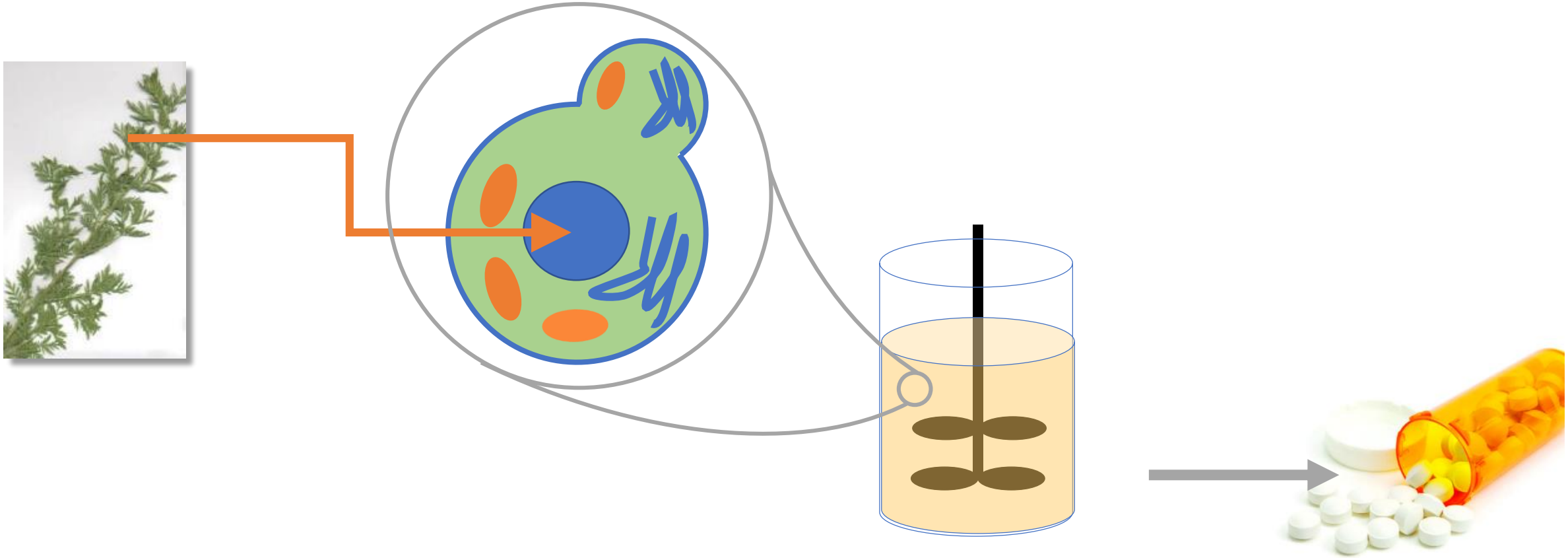
Down in every direction



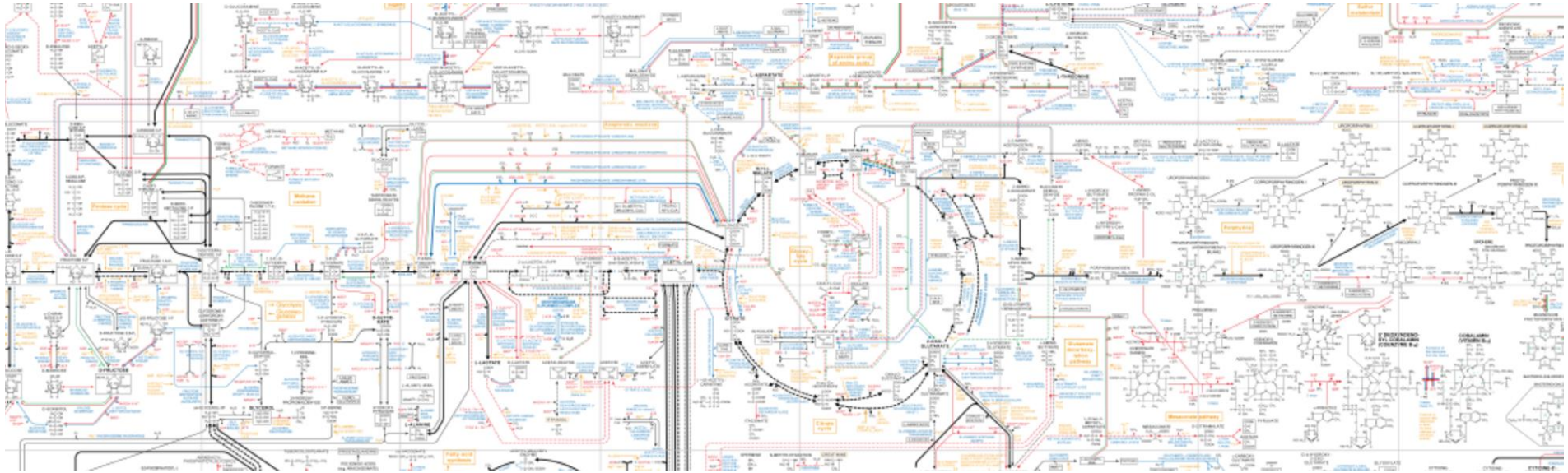
One minute course on genetic engineering

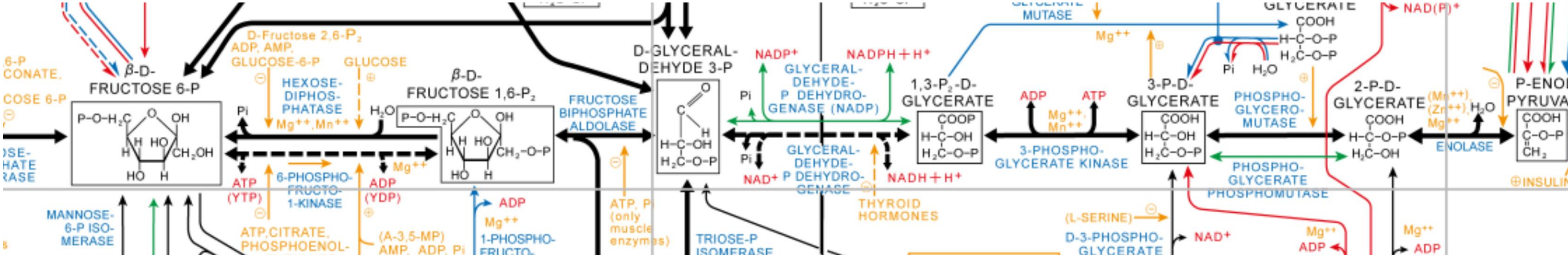
Demetrix

Nature's medicines through synthetic biology



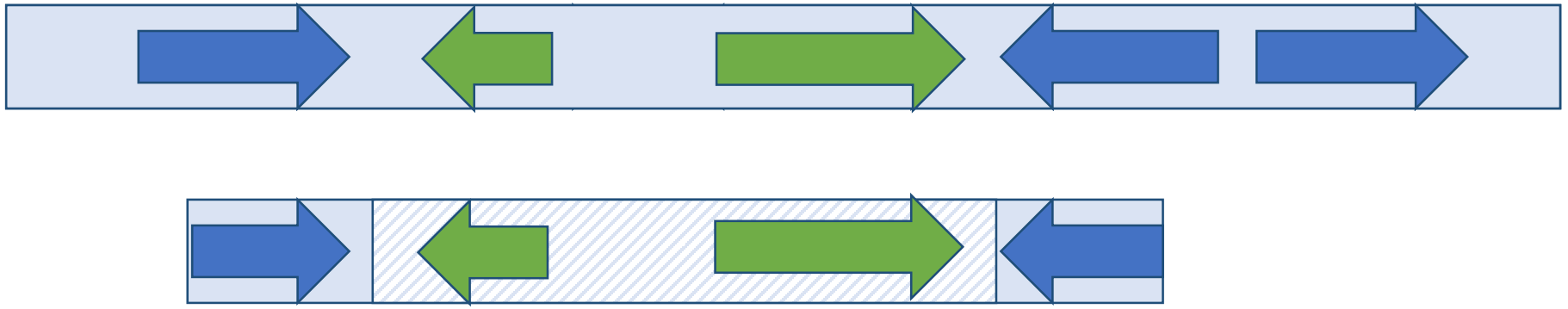
http://biochemical-pathways.com

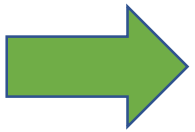




Yeast has a pretty sweet binary patch process

ATGACTAGTCTGATGTGATCTGATCTGATAT….



New gene  ATGCTGTATGCTTGATCGCGTTAGC..

Biotech software development environment

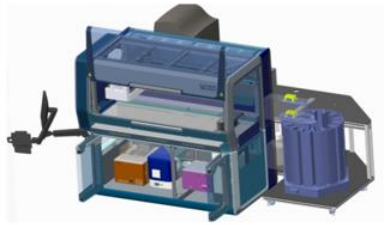
- Complex
- Usability has big impact [doing R&D]
- Exotic and mundane business activities
  - E.g. design a piece of DNA with certain characteristics
  - Where in the freezer would I find X
- Intersection of two complex domains [CS + molecular biology]
- Rapidly evolving requirements
- High app : developer ratio
- In-house users (few can access your software but you know them)
- Messy not Big data,
  - e.g. 10,000 vessels holding liquid in an experiment
  - Typically thousands to millions of data points per experiment set
  - Lots of corner cases
  - sequencing devices routinely emit gigs of data

My minimal shopping list for a programming language

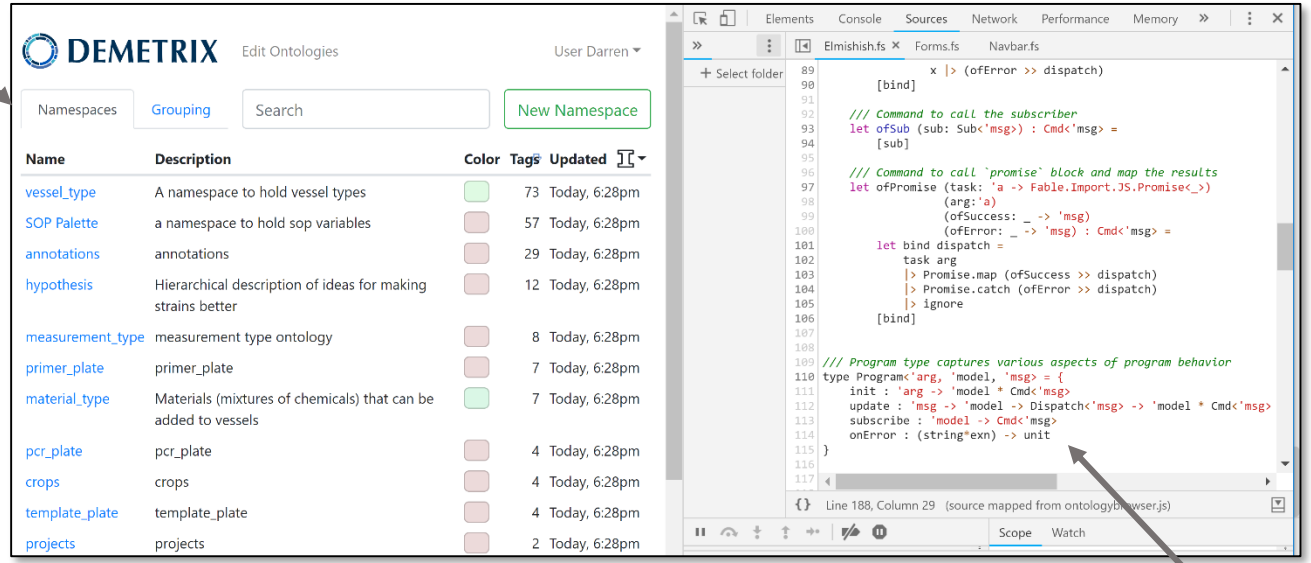
- Strongly typed - reduces errors
- Semantic whitespace - reduces errors
- Functional first - reduces errors
- Option types - reduces errors
- Units of measure - reduces errors
- Type inference - reduces RSI, makes #1 palatable
- Static typing - fast
- Good parallel primitives - lazy fast
- Rich built in data types - reduces development time
- Type providers - reduces development time
- Open source - reduces dependency
- Decent library support - don’t want to roll my own db driver
- Language interop (*) - increases library support
- Mature tooling - make CPU work for developer
- Supportive community - life is too short to hang out with jerks

Platform

User interfaces
Data visualization
Lab / Robot data services



Client



Fable

ASP.NET Core

GIRAFFE

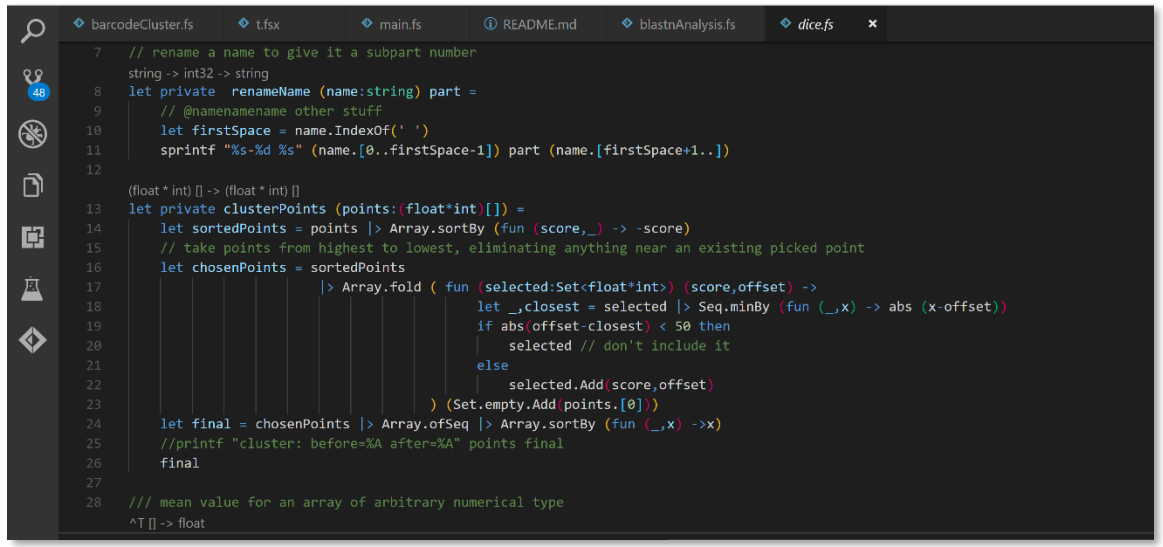
Full stack F#

Shared
code and
data definitions

Server

aws

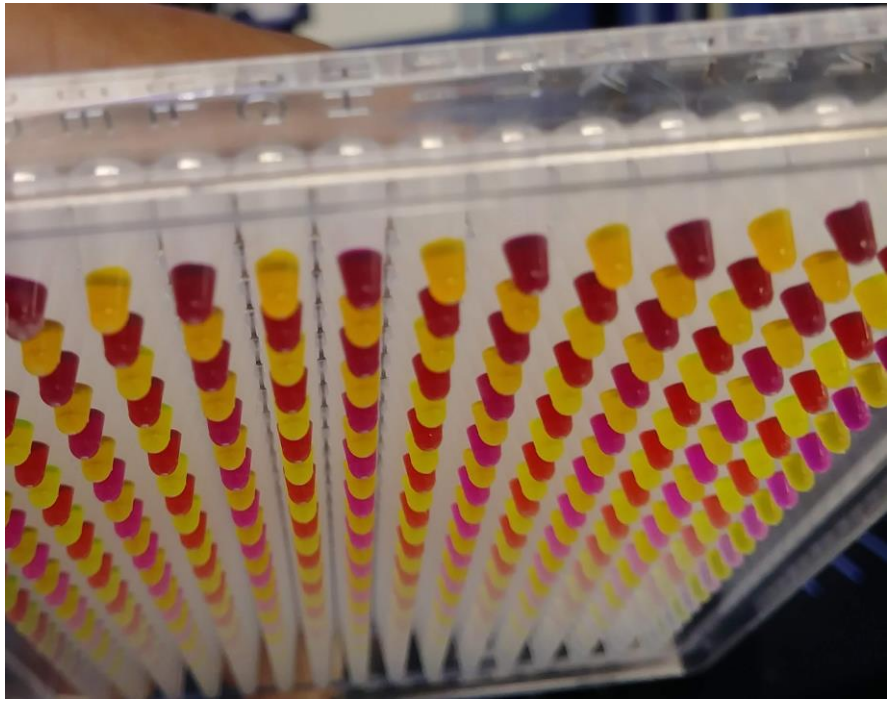
Algorithms
DNA Compilers
Data services / access layer



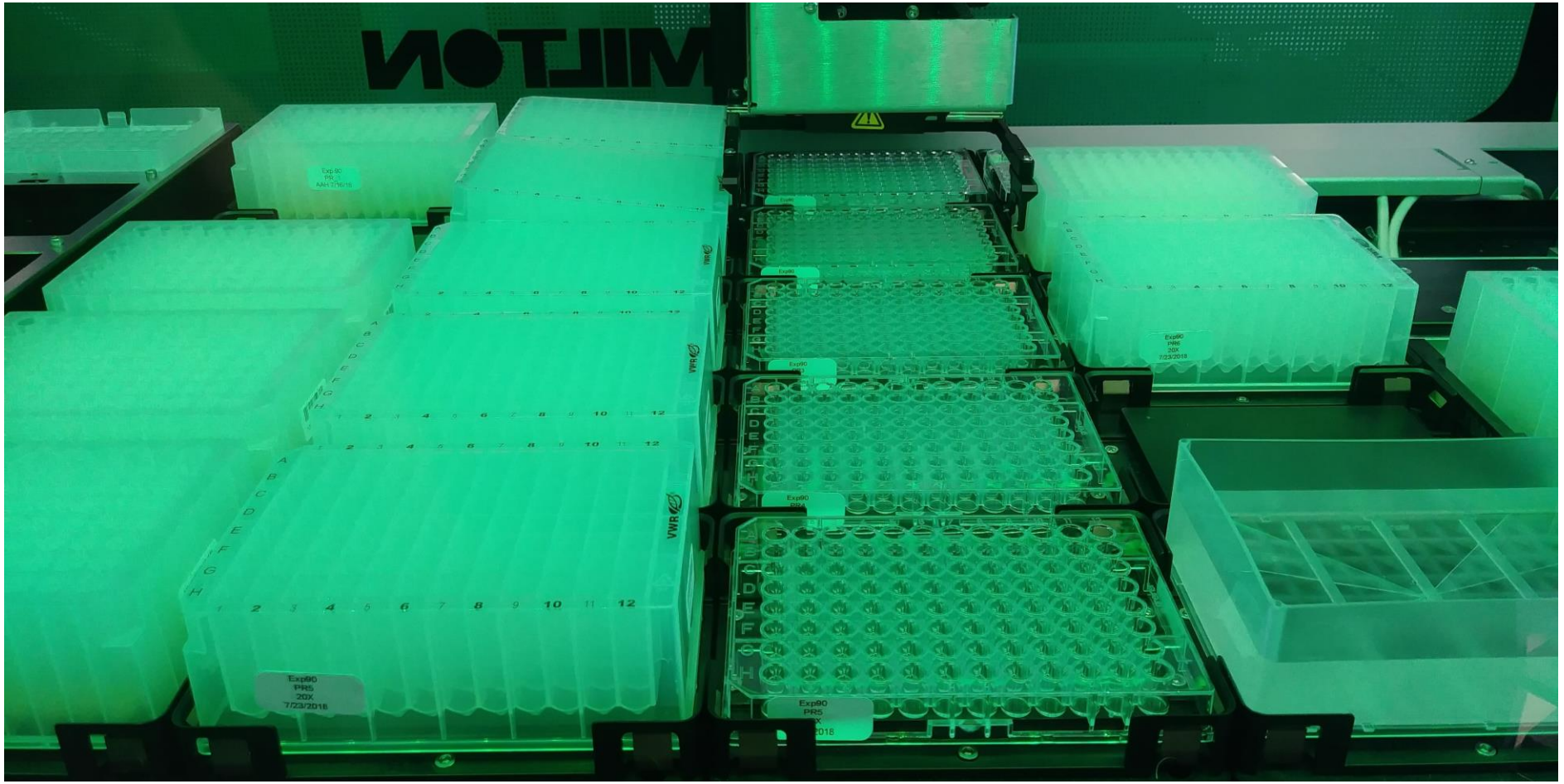
Our other platform..

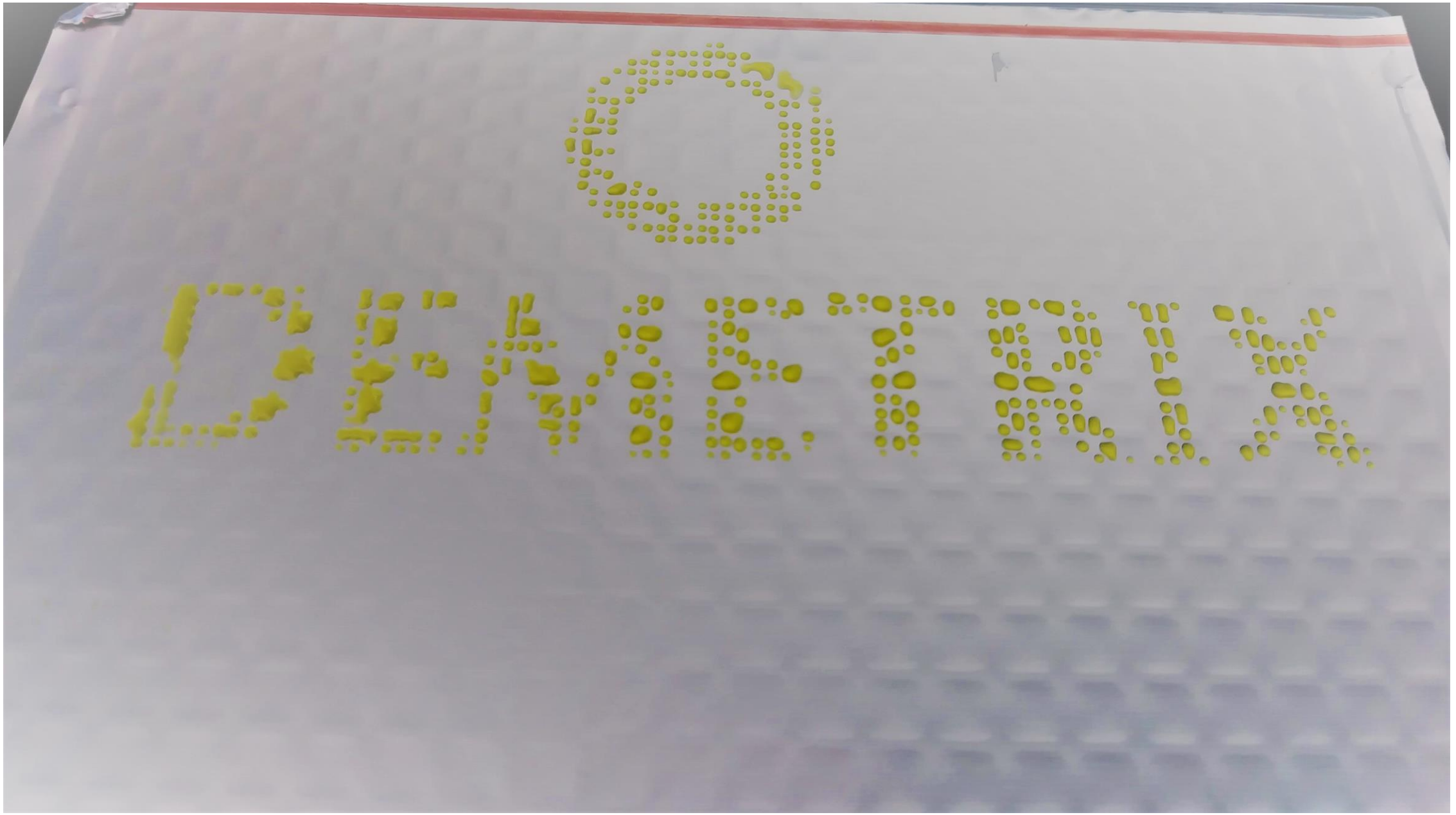
Moving colored liquids

- A lot of molecular biology looks just like precision cooking
  - Think French pastries..
- Moving liquids around



Plenty of tedious, highly automatable tasks





Example applications

DNA Design environment

DEMETRIX GSL Docs gsl paper demo [copy] User Darren

#49 gsl paper demo

```
// Yeast terpene design from Paddon et al,  Wilson et al.
// Level 1 Syntax
// Part definitions
#refgenome cenpk
let pGAL1_10 = gGAL1[-668:-1]
let truncHMGR = /ATGG/ {#rabitstart } ; gHMG1[1586:~200E]

let cassette(locus,gene1,gene2) =
    &locus.up ; ### ; !&gene1 ; &pGAL1_10 ; &gene2 ; &locus[-500:~500]
end


// Locus engineering
uERG9 ; ### ; pMET3 ; gERG9[1:~500]
cassette(gLEU2,mERG8,mMVD1)
cassette(gHIS3,mERG10,mERG12)
cassette(gADE1,mIDI1,&truncHMGR)
cassette(gURA3,mERG13,&truncHMGR)
cassette(gTRP1,&truncHMGR,mERG20)
```

2018 09/26 15:59.06 Success!

Flags Save And Compile Download

Genotype Specification Language

Erin H. Wilson, Shiori Sagawa, James W. Weis, Max G. Schubert, Michael Bissell, Brian Hawthorne, Christopher D Reeves, Jed Dean, and Darren Platt*

Amyris, Inc., 5885 Hollis Street, Suite 100, Emeryville, California 94608, United States

Letter | Published: 10 April 2013

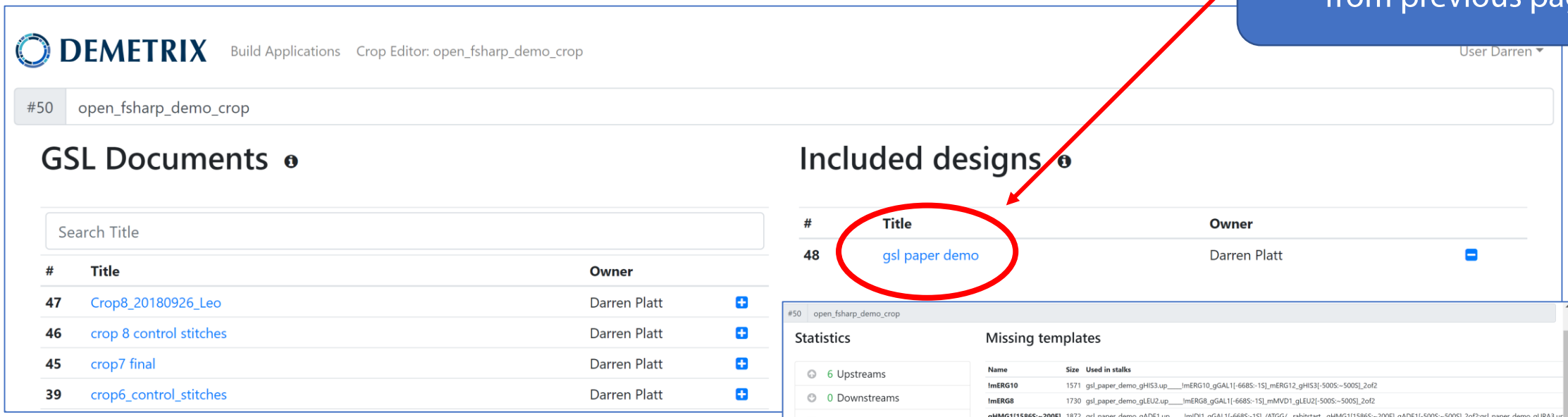
High-level semi-synthetic production of the potent antimalarial artemisinin

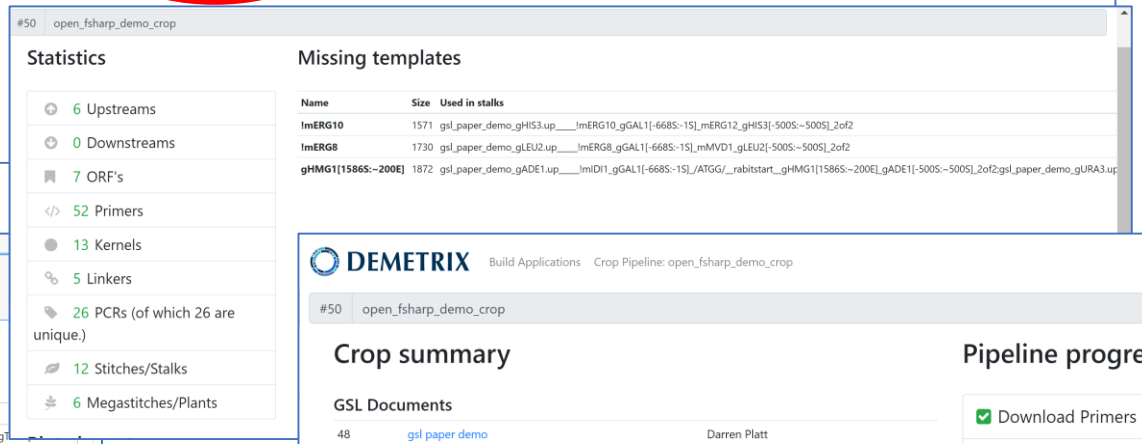
C. J. Paddon, P. J. Westfall […] J. D. Newman

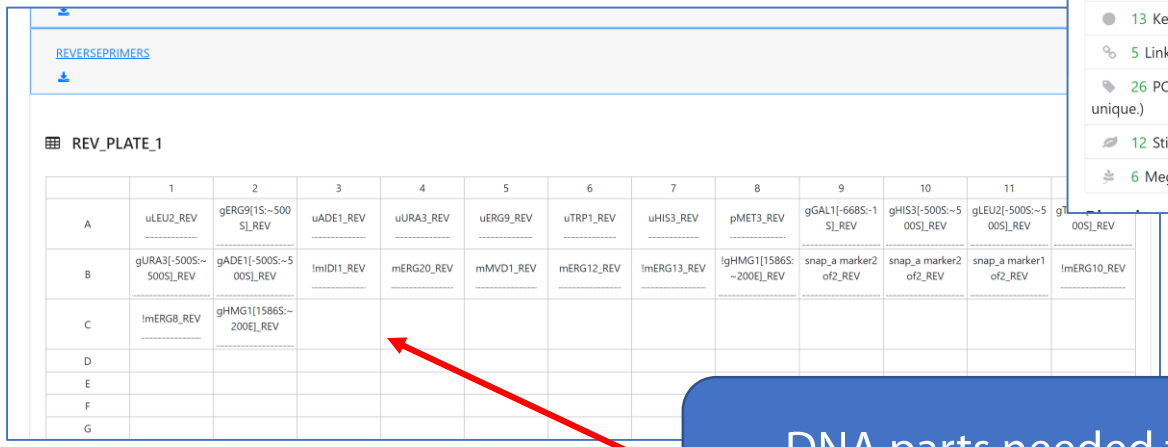
Nature **496**, 528–532 (25 April 2013) | Download Citation

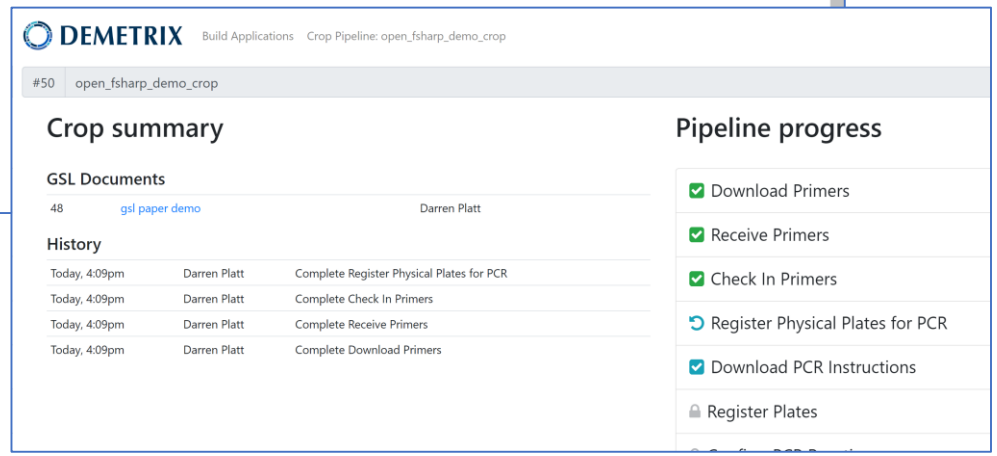
DNA Manufacturing Environment

Compiled design from previous page



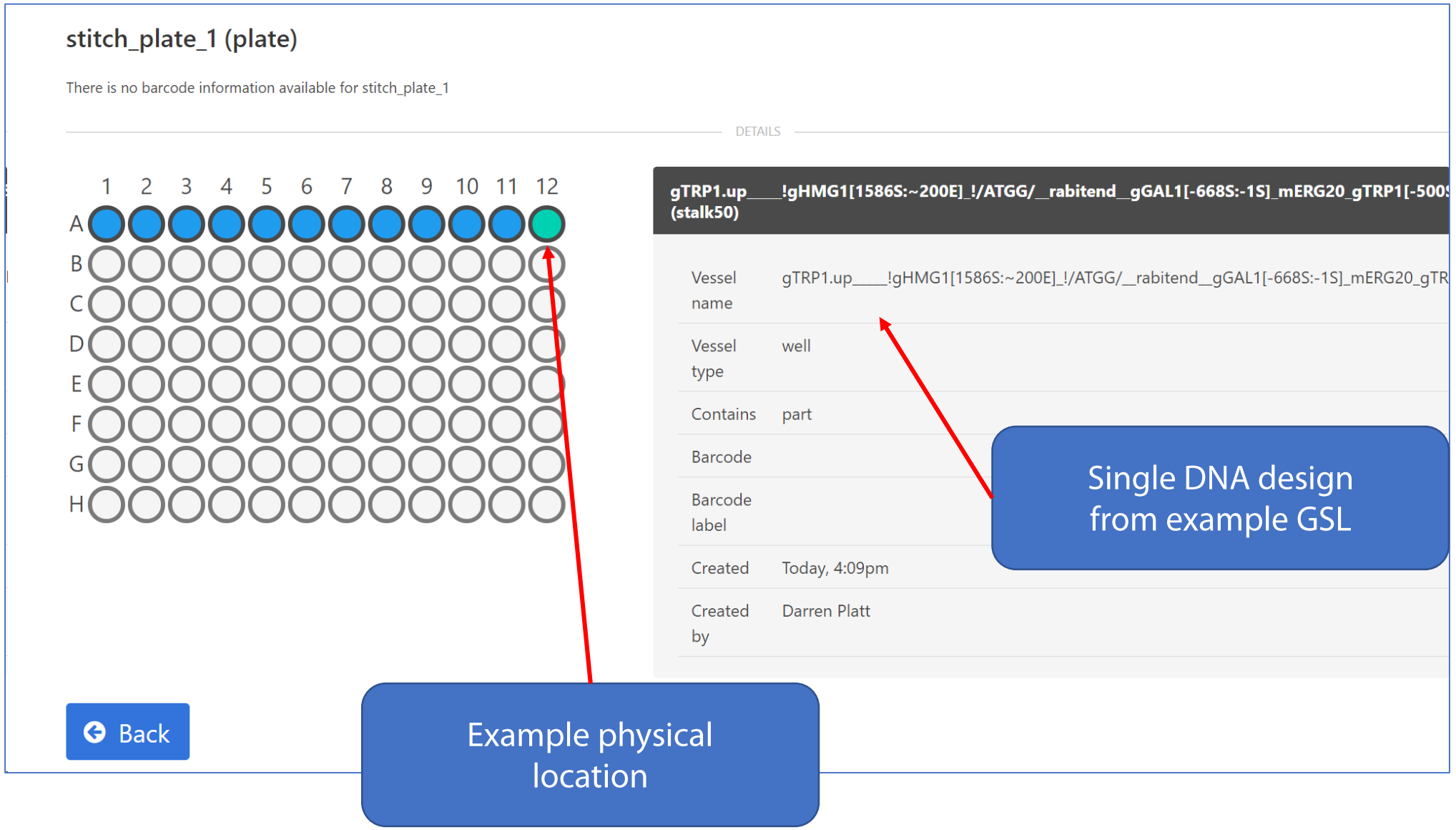




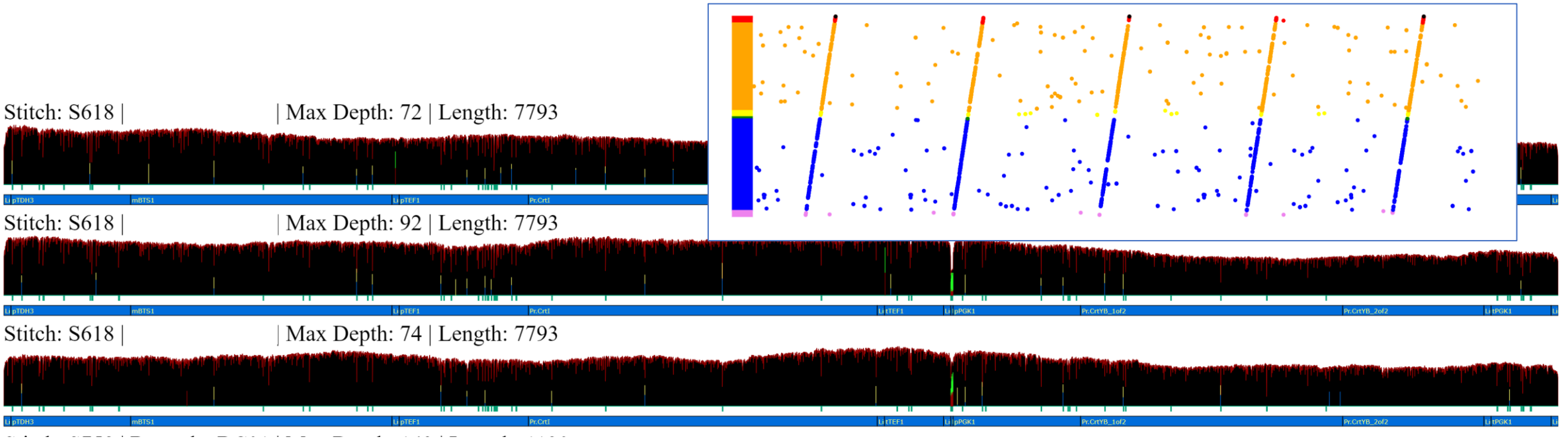


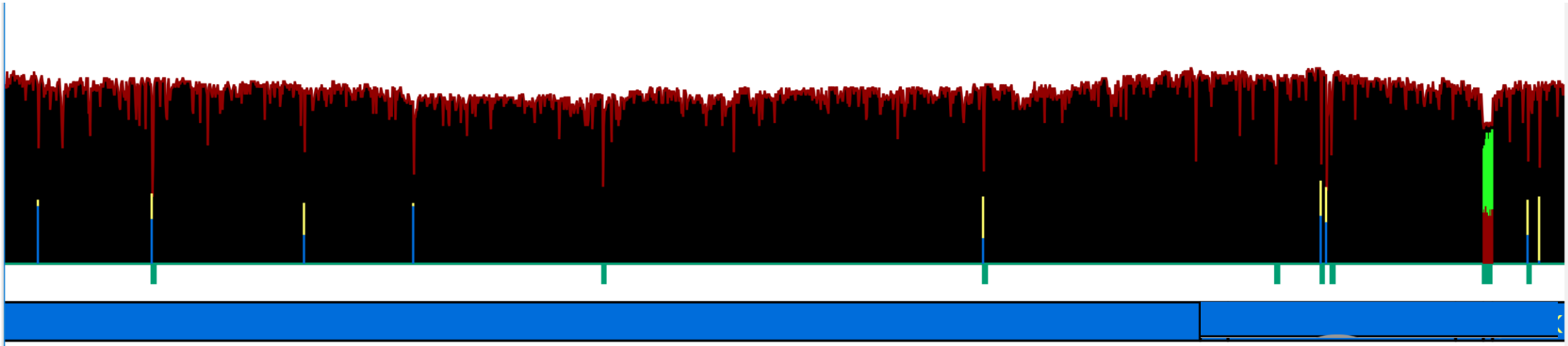
DNA parts needed for construction of design

Inventory

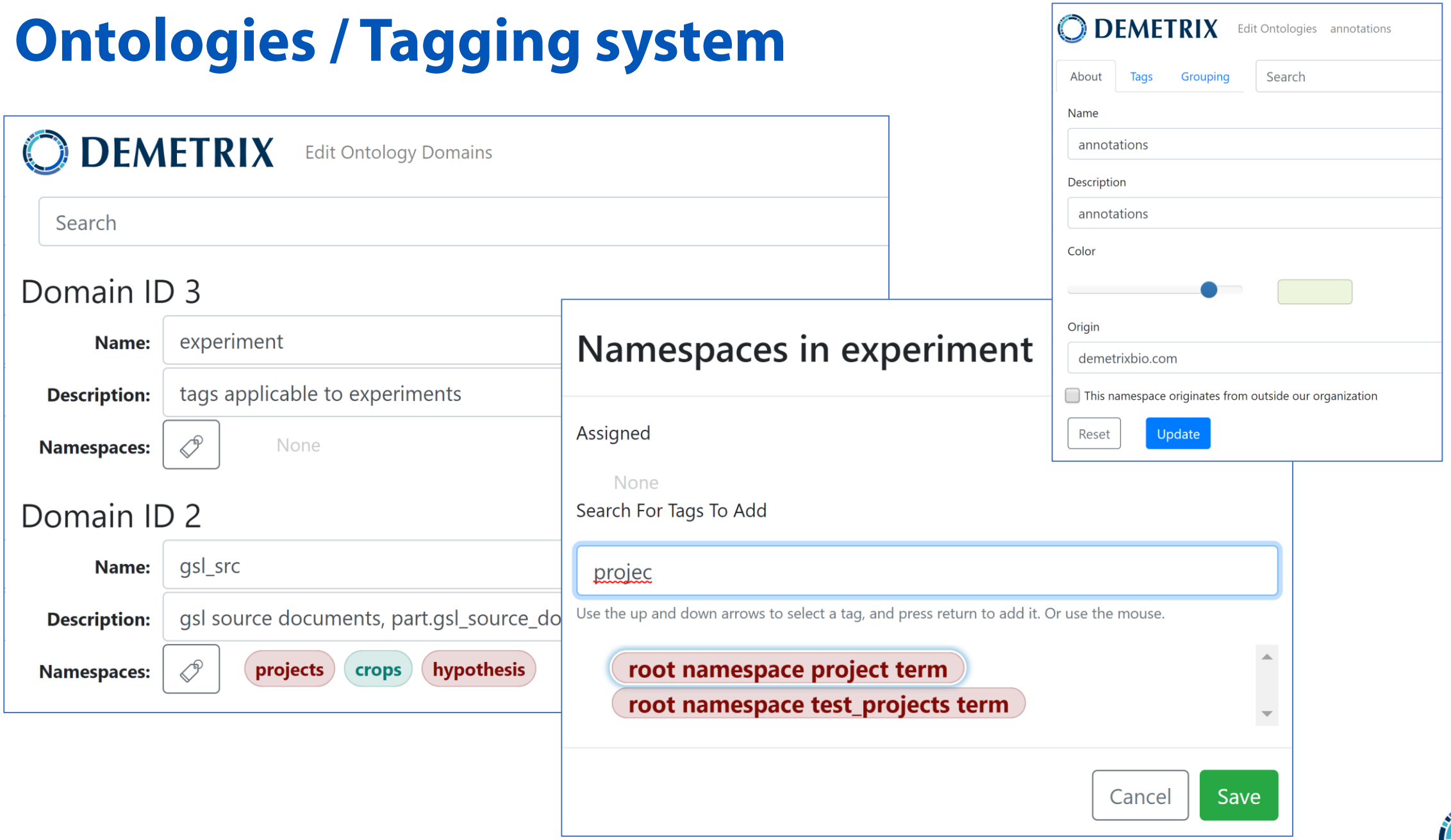


DNA Visualization





Ontologies / Tagging system



Things we like:
People / Community

Thanks

```
1070 +
1071 +        override __.ReturnType =
1072 +            if isTgt then
1073 +                match returnType.Namespace, returnType.Name with
1074 +                | "System", "Void"->
1075 +                    if ImportProvidedMethodBaseAsILMethodRef_OnStack_HACK() then
1076 +                        typeof<Void>
1077 +                    else
1078 +                        returnType
1079 +                | _ -> returnType
1080 +            else
1081 +                returnType
1082 +
```

cartermp closed this 3 days ago

dmitry-a-morozov referenced this issue in fsprojects/FSharp.TypeProviders.SDK 5 hours ago

.NET Standard 2.0 TPDTC with external dependencies example #244 Open

dmitry-a-morozov commented 5 hours ago

@cartermp I think you claimed victory too early
fsprojects/FSharp.TypeProviders.SDK#244
Hopefully it will get fixed soon.

dsyme commented 4 hours ago Contributor

I've find the cause of this:

> error FS1108: The type 'Void' is required here and is unavailable. You must add a reference to assembly 'System.Private.CoreLib, Version=4.0.0.0, Culture=neutral, PublicKeyToken=7cec85d7bea7798e'.

It's a painful problem and will require a painful workaround. More later tonight

1

Contributor

t package.

ovider

Less, Better, Software

dugnad

Noun
(*plural* dugnads)

- Unpaid voluntary, orchestrated community work.

CC2.0 https://www.flickr.com/photos/pagedooley/8435953365/sizes/l

Dugnad

Meetup

9 DEC

Past Meetup

Open Source Hack Day / F# Dugnad!

Hosted by Mathias B.

From The San Francisco F# User Group

Public group ?

Details

Are you itching to fix an issue in an F# open source library? Would you like to begin your journey in becoming an open source contributor? Want to sit down and hack on real F# code with friends?

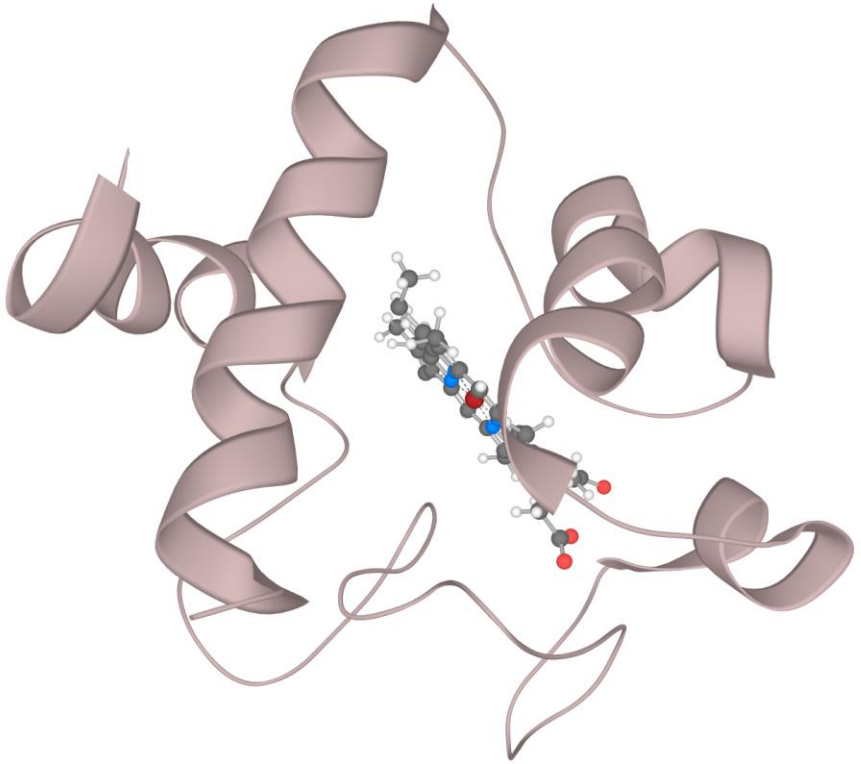
Then this event is for you! The idea is simple: let's meet one afternoon, pick a couple issues we'd like to improve in F# open source libraries, and work as a team, helping each other out to make the ecosystem a nicer place with some

https://twitter.com/brandewinder/status/939915781315354624

Things we like:
Fsharp.Data

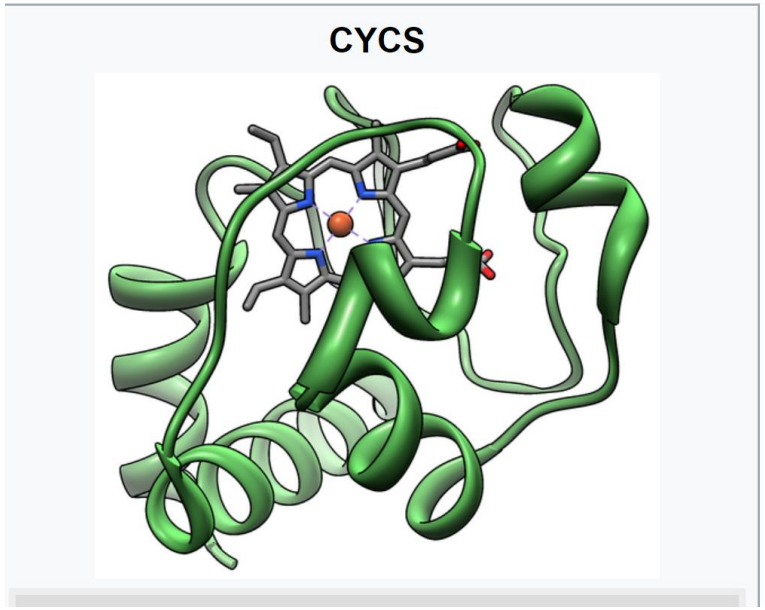
Blast example

```
>sp|P99999|CYC_HUMAN Cytochrome c OS=Homo sapiens OX=9606 GN=CYCS PE=1 SV=2
MGDVEKGKKIFIMKCSQCHTVEKGGKHKTGPNLHGLFGRKTGQAPGYSYTAANKNKGIIW
GEDTLMEYLENPKKYIPGTKMIFVGIKKKEERADLIAYLKKATNE
```



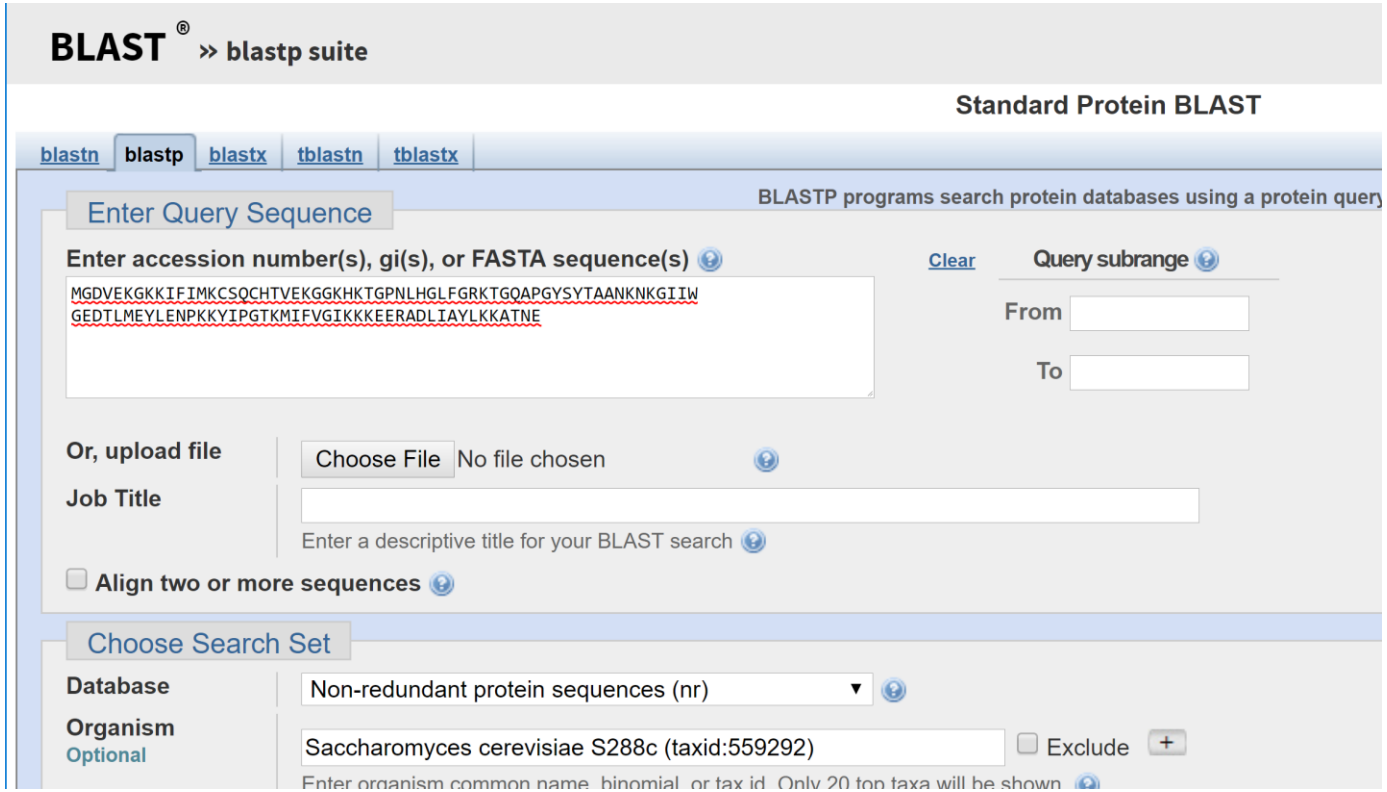
https://www.uniprot.org/uniprot/P99999#structure

https://en.wikipedia.org/wiki/Cytochrome_c



Query against yeast proteins

- Use tool called blastp (part of the blast family)
- https://blast.ncbi.nlm.nih.gov/Blast.cgi?PAGE=Proteins



```
Database: All non-redundant GenBank CDS
translations+PDB+SwissProt+PIR+PRF excluding environmental samples
from WGS projects
           169,976,473 sequences; 62,085,889,637 total letters
Query=
Length=105


                                                                  Score     E
Sequences producing significant alignments:                      (Bits)  Value

5LYC_A  Chain A, Cytochrome C In Complex With Phosphonato-cali...  143     1e-46
3TYI_A  Chain A, Crystal Structure of Cytochrome c - p-Sulfona...  143     1e-46
2N18_B  Chain B, Dominant form of the low-affinity complex of ...  143     2e-46
4N0K_A  Chain A, Atomic Resolution Crystal Structure Of A Cyto...  142     3e-46
4P4Q_B  Chain B, Complex Of Yeast Cytochrome C Peroxidase (w19...  141     9e-46
NP_012582.1  cytochrome c isoform 1 [Saccharomyces cerevisiae ...  141     9e-46
2N18_C  Chain C, Dominant form of the low-affinity complex of ...  141     1e-45
5KLU_A  Chain A, Crystal Structure Of A Domain-swapped Dimer O...  140     2e-45
5T7H_A  Chain A, Crystal Structure Of Dimeric Yeast Iso-1-cyto...  140     2e-45
4MU8_A  Chain A, Crystal Structure Of An Oxidized Form Of Yeas...  140     2e-45
4YE1_A  Chain A, A Cytochrome C Plus Calixarene Structure - Al...  140     2e-45
2MHM_A  Chain A, Solution structure of cytochrome c Y67H           140     3e-45
NP_010875.1  cytochrome c isoform 2 [Saccharomyces cerevisiae ...  138     1e-44
2LIR_A  Chain A, NMR Solution Structure of Yeast Iso-1-cytochr...  138     1e-44
```

Yeast and Human almost the same still..

```
>NP_012582.1 cytochrome c isoform 1 [Saccharomyces cerevisiae S288C]
 P00044.2 RecName: Full=Cytochrome c iso-1
 DAA08835.1 TPA: cytochrome c isoform 1 [Saccharomyces cerevisiae S288C]
Length=109

 Score = 141 bits (355),  Expect = 9e-46, Method: Compositional matrix adjust.
 Identities = 65/101 (64%), Positives = 78/101 (77%), Gaps = 0/101 (0%)

Query  2    GDVEKGKKIFIMKCSQCHTVEKGGKHKTGPNLHGLFGRKTGQAPGYSYTAANKNKGIIWG  61
            G  +KG  +F  +C QCHTVEKGG HK GPNLHG+FGR +GQA GYSYT AN  K ++W
Sbjct  7    GSAKKGATLFKTRCLQCHTVEKGGPHKVGPNLHGIFGRHSGQAEGYSYTDANIKKNVLWD  66

Query  62   EDTLMEYLENPKKYIPGTKMIFVGIKKKEERADLIAYLKKA  102
            E+ + EYL NPKKYIPGTKM F G+KK+++R DLI YLKKA
Sbjct  67   ENNMSEYLTNPKKYIPGTKMAFGGLKKEKDRNDLITYLKKA  107
```


XML equivalent

```
</Hit>
<Hit>
  <Hit_num>11</Hit_num>
  <Hit_id>gi|820957511|pdb|4YE1|A</Hit_id>
  <Hit_def>Chain A, A Cytochrome C Plus Calixarene Structure - Alternative Ligand Binding Mode &gt;gi|820957512|pdb|4YE1|B Chain B, A Cytochrome C Plus Calixarene Structure - Alternative Ligand Binding Mode</
  <Hit_accession>4YE1_A</Hit_accession>
  <Hit_len>108</Hit_len>
  <Hit_hsps>
    <Hsp>
      <Hsp_num>1</Hsp_num>
      <Hsp_bit-score>140.969</Hsp_bit-score>
      <Hsp_score>354</Hsp_score>
      <Hsp_evalue>1.75512e-45</Hsp_evalue>
      <Hsp_query-from>2</Hsp_query-from>
      <Hsp_query-to>102</Hsp_query-to>
      <Hsp_hit-from>6</Hsp_hit-from>
      <Hsp_hit-to>106</Hsp_hit-to>
      <Hsp_query-frame>0</Hsp_query-frame>
      <Hsp_hit-frame>0</Hsp_hit-frame>
      <Hsp_identity>65</Hsp_identity>
      <Hsp_positive>78</Hsp_positive>
      <Hsp_gaps>0</Hsp_gaps>
      <Hsp_align-len>101</Hsp_align-len>
      <Hsp_qseq>GDVEKGKKIFIMKCSQCHTVEKGGKHKTGPNLHGLFGRKTGQAPGYSYTAANKNKGIIWGEDTLMEYLENPKKYIPGTKMIFVGIKKKEERADLIAYLKKA</Hsp_qseq>
      <Hsp_hseq>GSAKKGATLFKTECLQCHTVEKGGPHKVGPNLHGIFGRHSGQAEGYSYTDANIKKNVLWDENNMSEYLTNPKKYIPGTKMAFGGLKKEKDRNDLITYLKKA</Hsp_hseq>
      <Hsp_midline>G  +KG  +F  +C QCHTVEKGG HK GPNLHG+FGR +GQA GYSYT AN  K ++W E+ + EYL NPKKYIPGTKM F G+KK+++R DLI YLKKA</Hsp_midline>
    </Hsp>
  </Hit_hsps>
</Hit>
<Hit>
  <Hit_num>12</Hit_num>
  <Hit_id>gi|700588045|pdb|2MHM|A</Hit_id>
  <Hit_def>Chain A, Solution structure of cytochrome c Y67H</Hit_def>
  <Hit_accession>2MHM_A</Hit_accession>
  <Hit_len>108</Hit_len>
  <Hit_hsps>
    <Hsp>
      <Hsp_num>1</Hsp_num>
      <Hsp_bit-score>140.198</Hsp_bit-score>
      <Hsp_score>352</Hsp_score>
```


Have a civil type safe conversation with XML

```
|> List.map (fun bc -> // add in meta data about the hsp in which they were found
                {bc with
                    hitLength = hsp.HspAlignLen
                    hitIdentity = (float hsp.HspIdentity)/(float hsp.HspAlignLen)
                    bitScore = hsp.HspBitScore |> float
                    qFrom = hsp.HspQueryFrom
                    qTo = hsp.HspQueryTo
                    subject = hit.HitDef
                    isFwd = hitFrame=1
                }
            )
```


Chunking XML input / ParallelSeq

How big are blast files?

```
string -> seq<BlastAnalysis>
let procBlast2 (file:string) =
    let lines = file
                |> Amyris.Bio.utils.eachLineIn
                |> Seq.skipWhile (fun line ->
                                      line.StartsWith("<Iteration>") |> not
                                  ) // Skip till we find iteration

    // accumulate one blast result
    let buffer = System.Text.StringBuilder()
    let iterations =
        seq { for line in lines do
                buffer.Append(line) |> ignore
                if line.StartsWith("</Iteration>") then
                    // End of one section, emit to parser      You, a few seconds ago
                    yield buffer.ToString()
                    buffer.Clear() |> ignore
        }
    iterations |> PSeq.mapi procOneIteration |> Seq.sort |> Seq.map snd
```

Parallelism

```
top - 09:20:50 up 12 days, 11:23,  4 users,  load average: 25.64, 21.16, 18.31
Tasks: 501 total,   2 running, 498 sleeping,   0 stopped,   1 zombie
%Cpu(s): 28.0 us,  0.9 sy,  0.1 ni, 71.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 32878972 total,  1264596 free,  4093984 used, 27520392 buff/cache
KiB Swap: 31249404 total, 31249404 free,        0 used. 27900812 avail Mem

   PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
117677           20   0 5648672 553636  32104 S 926.6  1.7 443:06.08 dotnet
114581           20   0 1878128 237468  34564 S   5.0  0.7 264:23.14 MinKNOW
```


Vendors love XML – it's not going away

```
    <DeviceId>HiP-ALS_1</DeviceId>
    <DisplayName>Multisampler</DisplayName>
    <IsRCDevice>True</IsRCDevice>
  </Devices>
  <Devices>
    <DeviceId>BinPump_1</DeviceId>
    <DisplayName>Binary Pump</DisplayName>
    <IsRCDevice>True</IsRCDevice>
  </Devices>
  <Devices>
    <DeviceId>QuatPump_1</DeviceId>
    <DisplayName>Quat. Pump</DisplayName>
    <IsRCDevice>True</IsRCDevice>
  </Devices>
  <Devices>
    <DeviceId>TCC_1</DeviceId>
    <DisplayName>Column Comp.</DisplayName>
    <IsRCDevice>True</IsRCDevice>
  </Devices>
  <Devices>
    <DeviceId>DAD_1</DeviceId>
    <DisplayName>DAD</DisplayName>
    <IsRCDevice>True</IsRCDevice>
```


CSV Type Provider is super useful

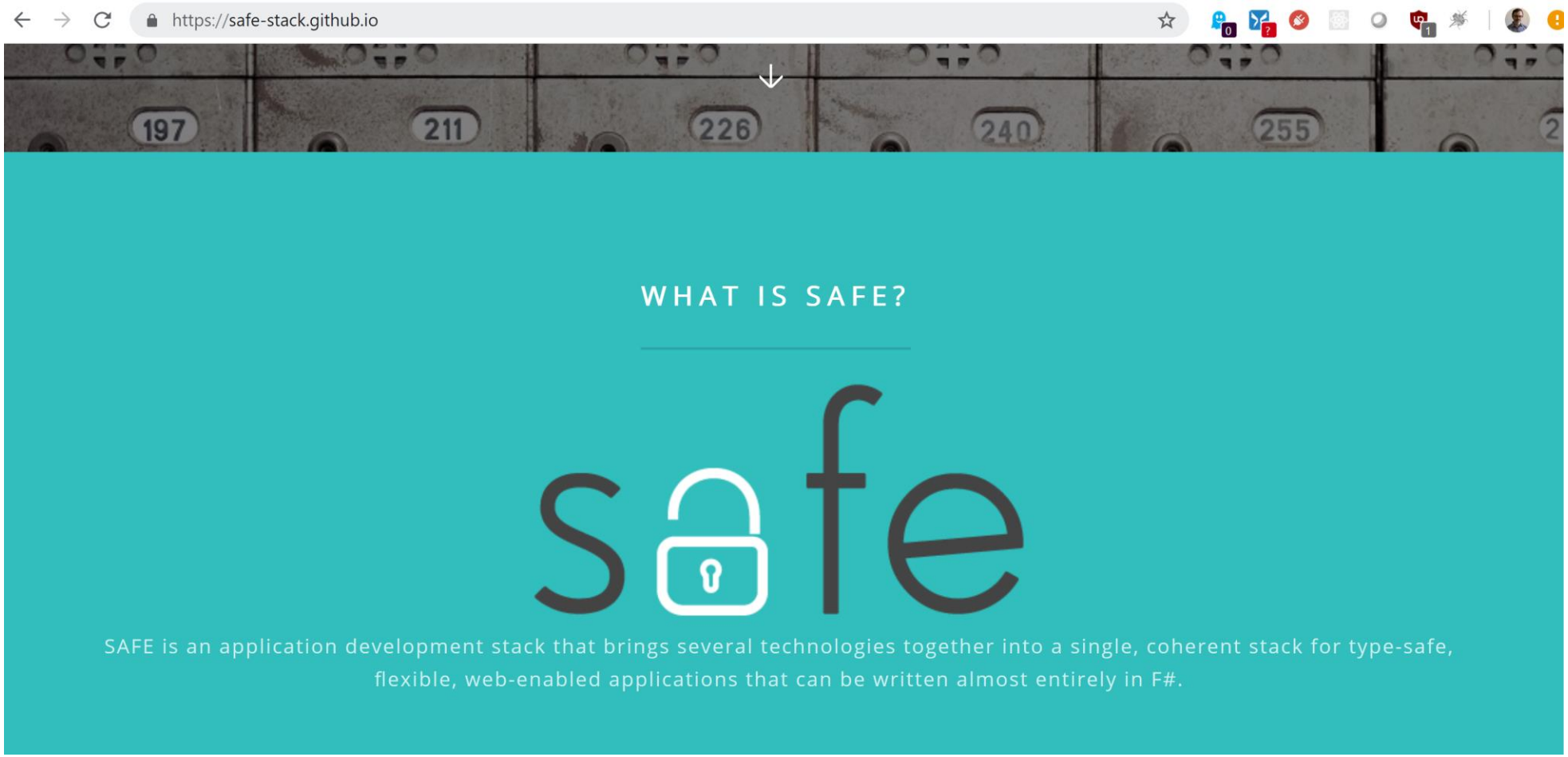
- Experiment CSVs
- Blast parsing, instrument files

```
You, 7 months ago | 1 author (You)
type ReadDispositionFile = CsvProvider<"name,ref,file,method
@87d3181f-a54a-4017-8497-9f2beae59bd1,X,Y,unknown
@3f2a0182-50e7-48ec-b6e1-93986db522fa,abc,def,unknown
@b7ac1606-1d50-453a-b9af-04b1ad63b627,-,-,unknown
@cdd33ab8-8308-4bcf-b48c-02cc0cdede5b,-,-,unknown">

let readDispositions = ReadDispositionFile.Parse(File.ReadAllText "read_disposition.txt").Rows

/// All the ref/file combinations we are mapping onto, in an arbitrary order
let allRefs = readDispositions
              |> Seq.map (fun row -> row.Ref,row.File)
              |> Seq.filter (fun (a,_) -> a <> "-")
              |> Seq.distinct
              |> Array.ofSeq
```


Things we like:
SAFE Stack

SAFE - Stack

G[Aws|GCP]FE’



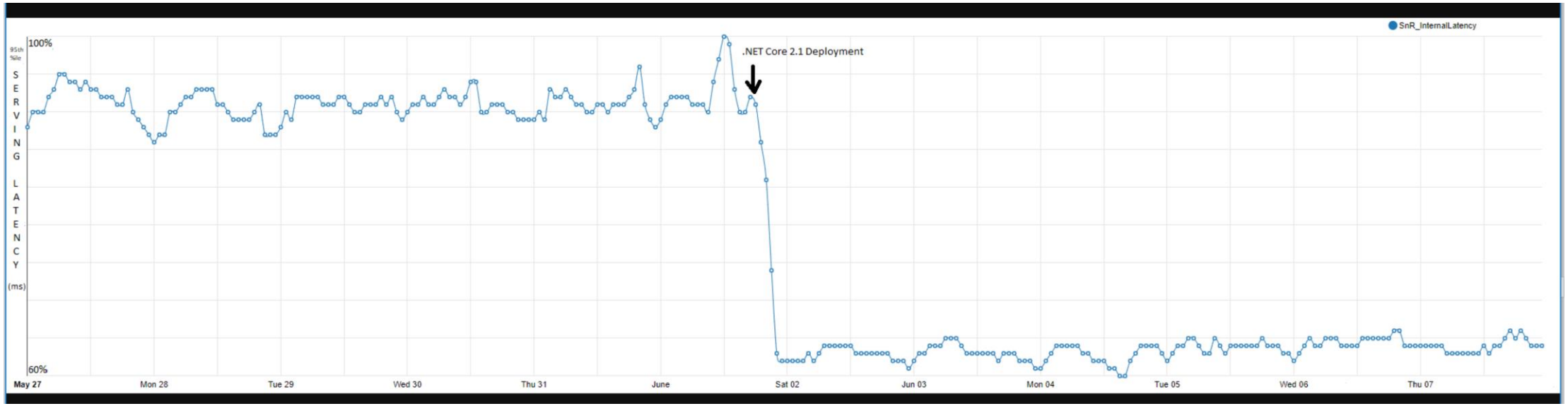
https://safe-stack.github.io/

Giraffe

- Smallish layer on top of industrial strength ASP.NetCore
- Easy to get started
- Small cost to add a new function
- Almost entirely used as an API server (no server side HTML rendering)
- https://github.com/giraffe-fsharp/Giraffe
- Respectable performance benchmarks

Aside: .NET Core performance rocks

Bing core improvement with .NET Core 2.1 deployment



https://msdnshared.blob.core.windows.net/media/2018/08/bingnetcoreimprovement2.png

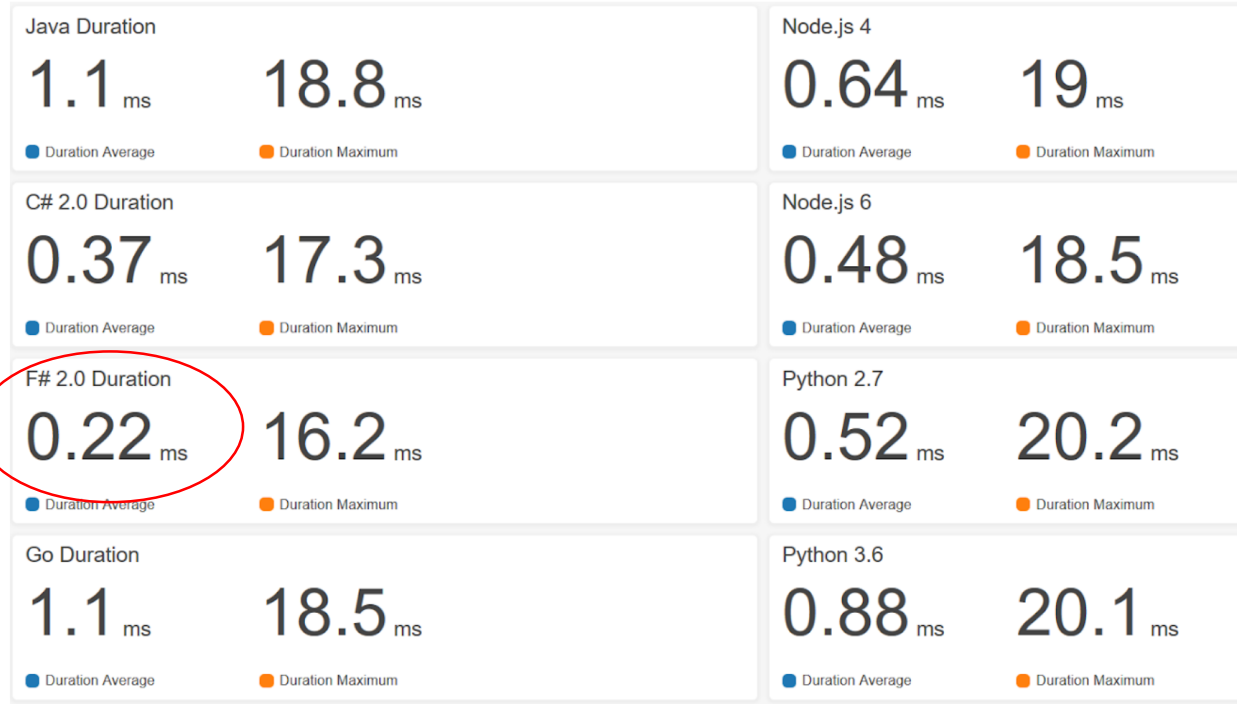
AWS lambda benchmarks

- https://read.acloud.guru/comparing-aws-lambda-performance-of-node-js-python-java-c-and-go-29c1163c2581

Observation 1—.Net Core 2.0 significantly outperforms

Both C# and F# on .Net Core 2.0 exceeds all expectations and outperforms all other runtimes in average duration. AWS Lambda developers on .Net Core should consider .Net Core 2.0 as the default—and upgrade all existing projects still using 1.0.

Observing the average durations on the graph, both C# and F# on .Net Core 2.0 are consistently lower than other runtimes throughout the 1-hour period:



Giraffe minimal project

```
open System
open Microsoft.AspNetCore.Builder
open Microsoft.AspNetCore.Hosting
open Microsoft.Extensions.DependencyInjection
open Giraffe

let webApp =
    choose [
        route "/ping"   >=> text "pong"
        route "/"       >=> htmlFile "/pages/index.html" ]

let configureApp (app : IApplicationBuilder) =
    // Add Giraffe to the ASP.NET Core pipeline
    app.UseGiraffe webApp

let configureServices (services : IServiceCollection) =
    // Add Giraffe dependencies
    services.AddGiraffe() |> ignore

[<EntryPoint>]
let main _ =
    WebHostBuilder()
        .UseKestrel()
        .Configure(Action<IApplicationBuilder> configureApp)
        .ConfigureServices(configureServices)
        .Build()
        .Run()
    0
```

Dispatch of URL endpoints

```
// API that requires authenticated user.  Returns boilerplate JSON error message if not authenticated.
routeStartsWith "/api" >=> Handlers.authenticateJSONIfOnline offline >=>
    choose [
        // GSLDocument API (get)

        // Load a source document, summary documents of its parents, and its compiler run
        routef "/api/gsleditor/%i" (makeJSONHandlerWithArg2Async gslAPI.getGSLEditorState)
        // Return a list of summary documents
        route "/api/gsldocumentlist" >=> makeJSONHandler2Async gslAPI.getGSLDocumentSummaries
        routef "/api/download/%i/%s"
                (fun (gslDocId,fileType) ->
                    let docs = gslAPI.getOutputFiles gslDocId fileType
                    let fileName = sprintf "gsl_output_%s.zip" (System.DateTime.Now.ToString("yyyyMMddHHmmss"))
                    let contentType = "application/zip"
                    makeDownload fileName contentType docs)
        route "/api/gsldocument/overview" >=> makeJSONHandler2 gslAPI.getGSLDocumentOverview

        // Ontology API (get)

        subRoute "/api/onto" (
            choose [
                route "/namespace/all" >=> makeJSONHandler2Async ontAPI.getNamespacesAndMasterTags
                routef "/namespace/%i" (makeJSONHandlerWithArg2Async ontAPI.getNamespaceAndTags)
                routef "/namespace/name/%s" (makeJSONHandlerWithArg2Async ontAPI.getNamespaceAndTagsByName)
                route "/domain/all" >=> makeJSONHandler2 ontAPI.getOntologyDomains
                routef "/domain/name/%s" (makeJSONHandlerWithArg2Async ontAPI.getOntologyDomainByName)
                routef "/tagassignment/domainnameandobj/%s/%i" (makeJSONHandlerWithArg2Async ontAPI.getAssigned
                routef "/tagassignment/domainandobj/%i/%i" (makeJSONHandlerWithArg2Async ontAPI.getAssignedTags
                routef "/tagassignment/domainname/%s" (makeJSONHandlerWithArg2Async ontAPI.getAssignedTagsForDo
```

Error handling: a personal Journey

```
      C:  return -1;
Python:  return (-1,1234)
Python:  return ((-1,"error msg"),1234)
    F#:  None
    F#:  None,"error msg"
    F#:  Some 1234,"ok!"
    F#:  Result<Success of 1234 | Failure of string>
    F#:  Success of {data=1234 ; warning=""}
    F#:  Failure "error msg"
    F#:  Failure of ErrorType
```


Fleshed out Success and Failure

```
type FailureMessage =
    Gien Verschatse, 10 days ago
    | Unknown of string
    | Database of string
    | Parse of string
    | Validation of string
    | NotFound of string
    | ExceptionFailure of string

type Success<'TSuccess> =
    Gien Verschatse, 10 days ago
    { Data : 'TSuccess
      Warnings : string list }
```


Routing

```
route "/api/example/success" >=> makeJSONHandler2Async exampleGETSuccess
route "/api/example/fail" >=> makeJSONHandler2Async exampleGETFail
route "/api/example/warning" >=> makeJSONHandler2Async exampleGETWarning
route "/api/example/exception" >=> makeJSONHandler2Async exampleException
```


Simple GET success

```
int -> Async<Result<Success<Example>,'a>>
let exampleGETSuccess userId =
    async {
        return succeed  { Greeting = "success" ; Method = "GET" ; UserId = userId}
    }
```


GET / Fail

```
'a -> Async<Result<'b,FailureMessage>>
let exampleGETFail userId =
    async {
        return
            FailureMessage.Unknown
                "failing example (unknown cause)"
            |> fail
    }
```

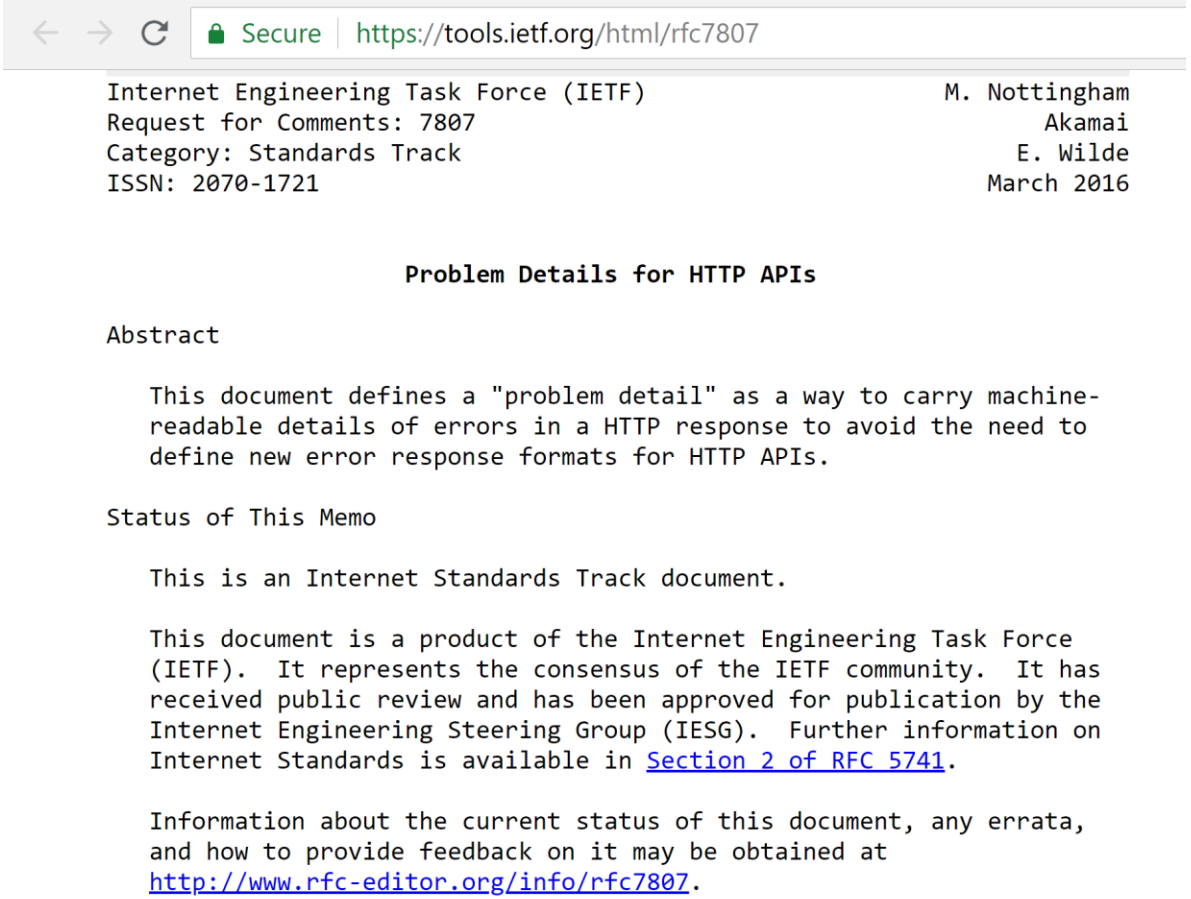

Success with warning

```
int -> Async<Result<Success<Result<Example,'a>>,'b>>
let exampleGETWarning userId =
    async {
        return
            Ok { Greeting = "success" ; Method = "GET" ; UserId = userId}
            |> warn "Ok with warning example"
    }
```


Play nice with Web / RFC7807

```
match response with
| Ok success -> (json success) next ctx
| Error failure ->
    let problem = failureToProblemReport failure
    // Can't use ResponseWriter.json because it sets the content type.
    // So we call the serializer ourselves.
    let serializer = ctx.GetJsonSerializer()
    let serializedProblem = serializer.Serialize problem
    let errorResponse = clearResponse
                        >=> setStatusCode (failureTypeAsHttpStatus failure)
                        >=> setHttpHeader "Content-Type" "application/problem+json" // https://tools.ietf.org/html/rfc7807
                        >=> setBodyFromString serializedProblem
    errorResponse next ctx
```


https://tools.ietf.org/html/rfc7807

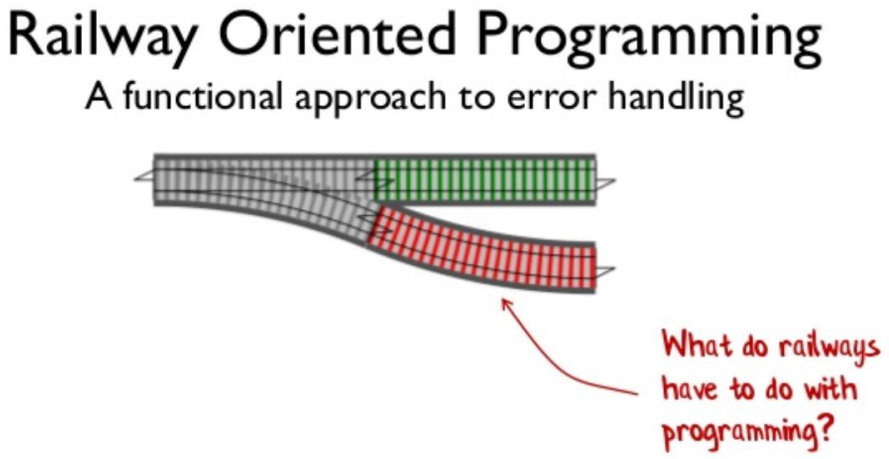


For example, an HTTP response carrying JSON problem details:

```
HTTP/1.1 403 Forbidden
Content-Type: application/problem+json
Content-Language: en

{
 "type": "https://example.com/probs/out-of-credit",
 "title": "You do not have enough credit.",
 "detail": "Your current balance is 30, but that costs 50.",
 "instance": "/account/12345/msgs/abc",
 "balance": 30,
 "accounts": ["/account/12345",
              "/account/67890"]
}
```

Final plug for principled error handling

https://fsharpforfunandprofit.com/rop/

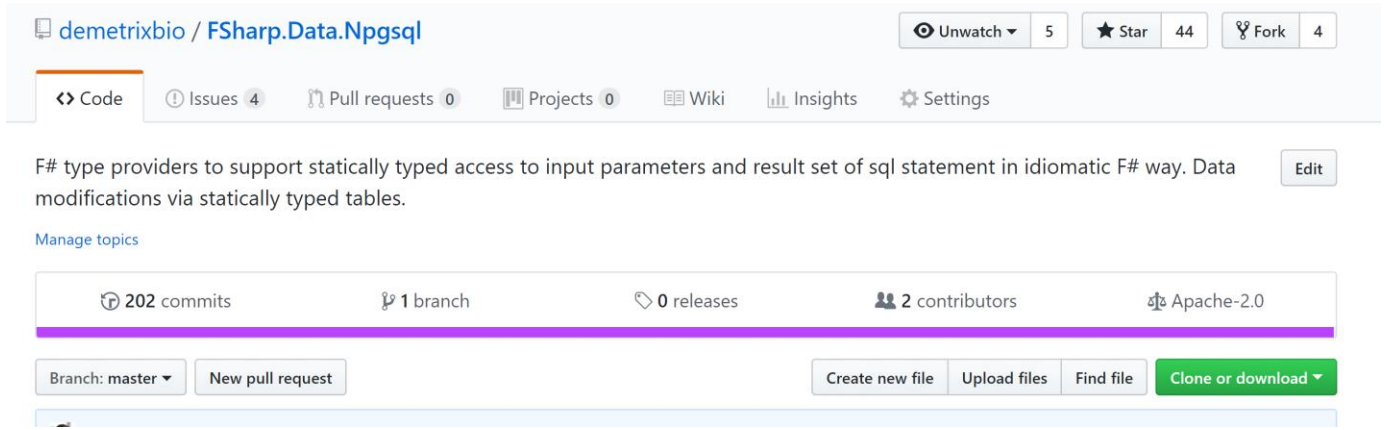


Dmitry Morozov
@mitekm

Postgres Type Provider

https://github.com/demetrixbio/FSharp.Data.Npgsql

DEMETRIX



Simple Example

```
do
    use cmd = DvdRental.CreateCommand<"SELECT title, release_year FROM public.film LIMIT 3">(dvdRental)

    for x in cmd.Execute() do
        printfn "Movie '%s' released in %i." x.title x.release_year.Value
```


Postgres Type provider example

```
use conn = openConnection()
use trans = conn.BeginTransaction()

use gslCompilerRunUpsert = new NpgsqlCommand<"
                                INSERT INTO part.gsl_compiler_run
                                    (id_source_doc, compiler_message, exit_code, errors, warnings,created)
                                VALUES
                                    (@doc_id, @compiler_message, @exit_code, @errors, @warnings,now())
                                ON CONFLICT (id_source_doc)
                                DO UPDATE SET
                                    (compiler_message, exit_code, errors, warnings) =
                                    (@compiler_message, @exit_code, @errors, @warnings)
                                WHERE
                                    part.gsl_compiler_run.id_source_doc = @doc_id",DevCs>(conn)
// Create or update the compiler run record
gslCompilerRunUpsert.Execute(
                doc_id =           doc.index,
                compiler_message = doc.compilerMessage,
                exit_code =        doc.exitCode,
                errors =           doc.errors,
                warnings =         doc.warnings
               ) |> ignore
```

Runtime connection

Design time connect string

Proud to support Fable

Acknowledgments

We had many great contributors during Fable 2 development, so I apologize in advance if I'm forgetting someone. If you like the improvements, please thank ncave and Maxime Mangel who have put a lot of work in this new release. But also Valery Vitko, Ilja Nosik, Zack Young, Zaid Ajaj and Aleksander Heintz.

And of course not only the compiler, the ecosystem arount it is very important to keep Fable alive, so big kudos from her to Eugene Tolmachev, Diego Esmerio, humhei, Kunjan Dalal, Stachu Korick, and the SAFE-stack team!

I also want to thank my employers, who trust Fable (and me!) and are really supportive to keep its development active: Ben Taylor, Robert Kuzelj and Demetrix.

Things we love: Elmish

Elmish

> `-ish`
>
> a suffix used to convey the sense of “having some characteristics of”

Elmishish: Elm |> ish |> ish

DEMETRIX

```
(**
Elmish-ish Program
Code taken from the 1.0.0 release of Elmish and modified.
https://github.com/fable-elmish/elmish
With much gratitude to the Elmish developers.
Modifications:
This derivation of Elmish is not strongly bonded to React.
All 'view'-specific functionality has been removed, and the 'model update' phase is now a generalized message servicing phase.
The servicing phase gets a reference to dispatch, instead of only returning messages, so it can create DOM objects with callbacks.
*)
```

Elmishish

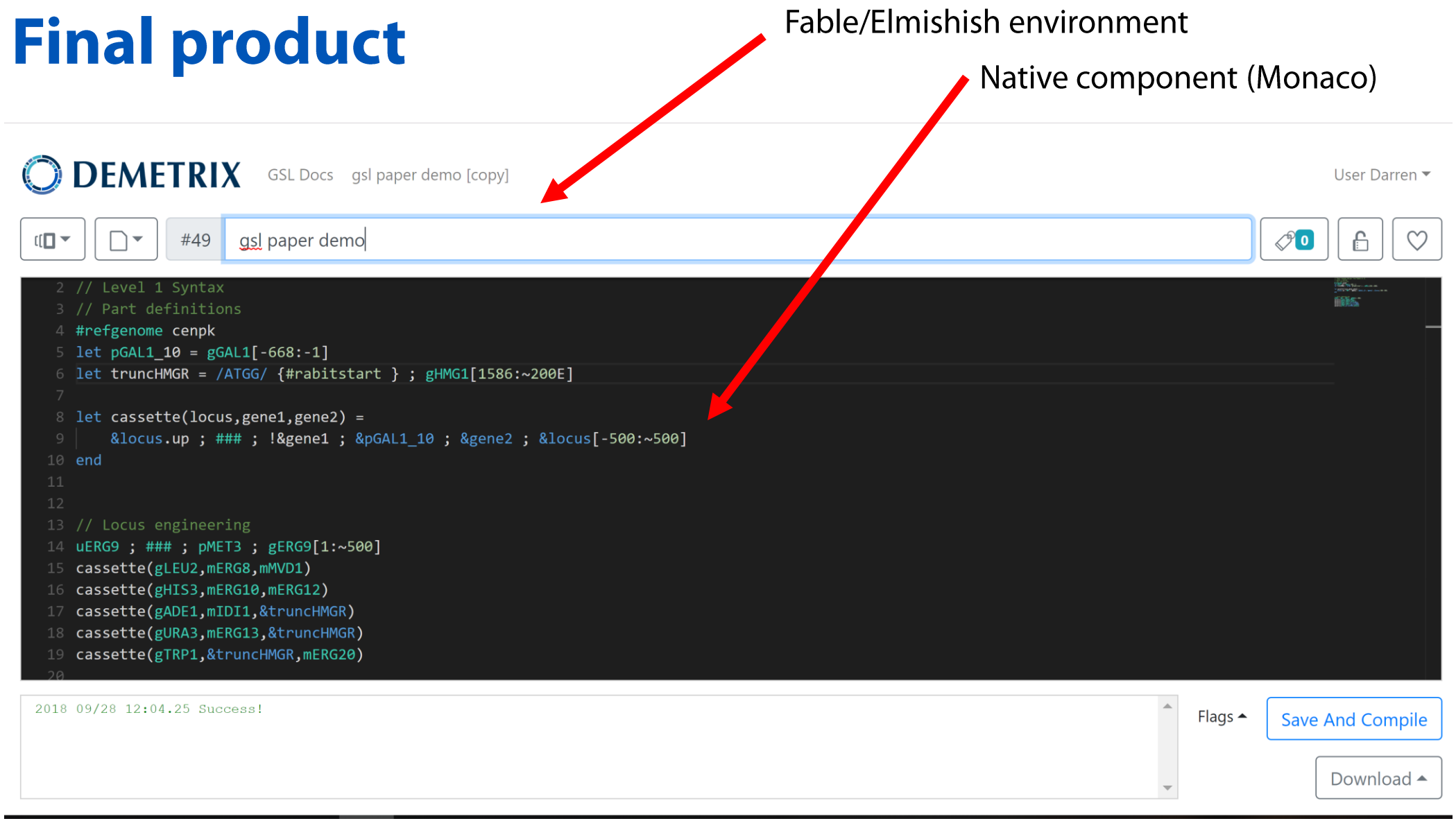
```
Msg -> Model -> (Msg -> unit) -> Model * Cmd<Msg>
let update (msg:Msg) model dispatch : Model*Cmd<Msg> =
    match msg with
    | LoginMsg lmsg ->
        // The Login UI expects to dispatch messages of type Login.Msg only.
        // So we wrap the dispatch we're given in an adapter before passing it down.
        let ldispatch lmsg = dispatch (LoginMsg lmsg)
        let lm,lcmd = Login.update lmsg model.loginModel ldispatch
        // The resulting messages (in commands) that are returned from Login's
        // update function need to be cast into our carrier subtype before they're passed up.
        let lcmd = Cmd.map LoginMsg lcmd
        // If we got LogInSuccess, add another command to proceed past the login page.
        let m = { model with loginModel = lm }
        match lmsg with
            | Login.LogInSuccess ->
                // Call our custom navbar update function
                UI.updateNavBarUI m dispatch |> ignore
                m, Cmd.batch [lcmd; Cmd.ofMsg OpenEditor]
            | Login.CheckLogInResponse false ->
                renderNavBarUI lm ldispatch [] |> ignore
                m, Cmd.batch [lcmd; Cmd.ofMsg CloseEditor]
            | _ -> m, lcmd

    | OpenEditor ->
        UI.renderEditorUI model dispatch |> ignore
        // If we haven't already obtained an initialized Monaco editor instance in the model,
        if model.monacoEditorInstance.IsNone then
            // Call the custom init routine on the Monaco editor element, now that the element exists
            UI.loadEditorPanel dispatch
        model, Cmd.none
```

```
(Msg -> unit) -> unit
let loadEditorPanel (dispatch) =
    let callback () = dispatch Msg.LoadedMonaco
    dmxMonacoJs.loadMonacoEditor callback
```

Wrapping native JS components

```
type IDMXMonacoJs =
    abstract loadMonacoEditor: (unit->unit) -> unit
    abstract initializeMonacoEditor: string -> string -> (unit->obj) -> (int->unit) -> unit
    abstract setEditorContents: int -> string -> unit
    abstract getEditorContents: int -> string
    abstract toggleEditorReadOnly: int ->bool -> unit
```


Final product



View layer example

```
38      let r =
39       table [ClassName tableClassName; Id "gslDocTable"] [
40          thead [] [
41              tr [] [
42                  (tableColHeader "#" "id" BrowserTableColumn.BTCID)
43                  (
44                      if model.showTags then
45                          tableColHeader "Title / Tags" "title" BrowserTableColumn.BTCTitle
46                      else
47                          tableColHeader "Title" "title" BrowserTableColumn.BTCTitle
48                  )
49                  (tableColHeader "Owner" "owner" BrowserTableColumn.BTCOwner)
50                  (tableColHeader "Created" "created" BrowserTableColumn.BTCCreated)
51                  (tableColHeader "Modified" "modified" BrowserTableColumn.BTCModified)
52                  th [Scope "col"
53                      ColSpan 2.0
54                      ClassName "dropdown dmx-columnChooserButton"
55                     ] [
56                      a [
57                          ClassName "dropdown-toggle"
58                          Role "button"
59                          DataToggle "dropdown"
60                          AriaHasPopup true; AriaExpanded false
61                      ] [
62                          U.UI.svgIconColumns
63                      ]
64
```


View example 2

```
                    compare b.dateSubmitted a.dateSubmitted
// Modify the sort to run backwards if the sort direction is reversed
let directionalSortFunc =
    match model.sortDirection with
    | Ascending -> fun (a:GSLDocumentSummary) (b:GSLDocumentSummary) -> sortFunc b a
    | Descending -> sortFunc
docList
|> List.sortWith directionalSortFunc
// Render the filtered rows
|> List.map (fun docSum ->
    tr [] [
        th [Scope "row"] [
            a [ Href (Pages.toURL (Pages.Page.GSLEditor (string docSum.index))) ]
                (U.UI.highlightSearchTermsInString (string docSum.index) model.searchTerms)
        ]
        td [] (
            let linkElement = [
                    div [] [
                        a [ Href (Pages.toURL (Pages.Page.GSLEditor (string docSum.index))) ]
                            (U.UI.highlightSearchTermsInString docSum.title model.searchTerms
                    ]
                ]
            let contentSearchElements =
                if model.serverSideSearchResults.ContainsKey docSum.index |> not then [] else
                    let docContRes = model.serverSideSearchResults.[docSum.index]
                    List.append (
```


Things we survived:
Cloud interactions

Cloud deployment easy

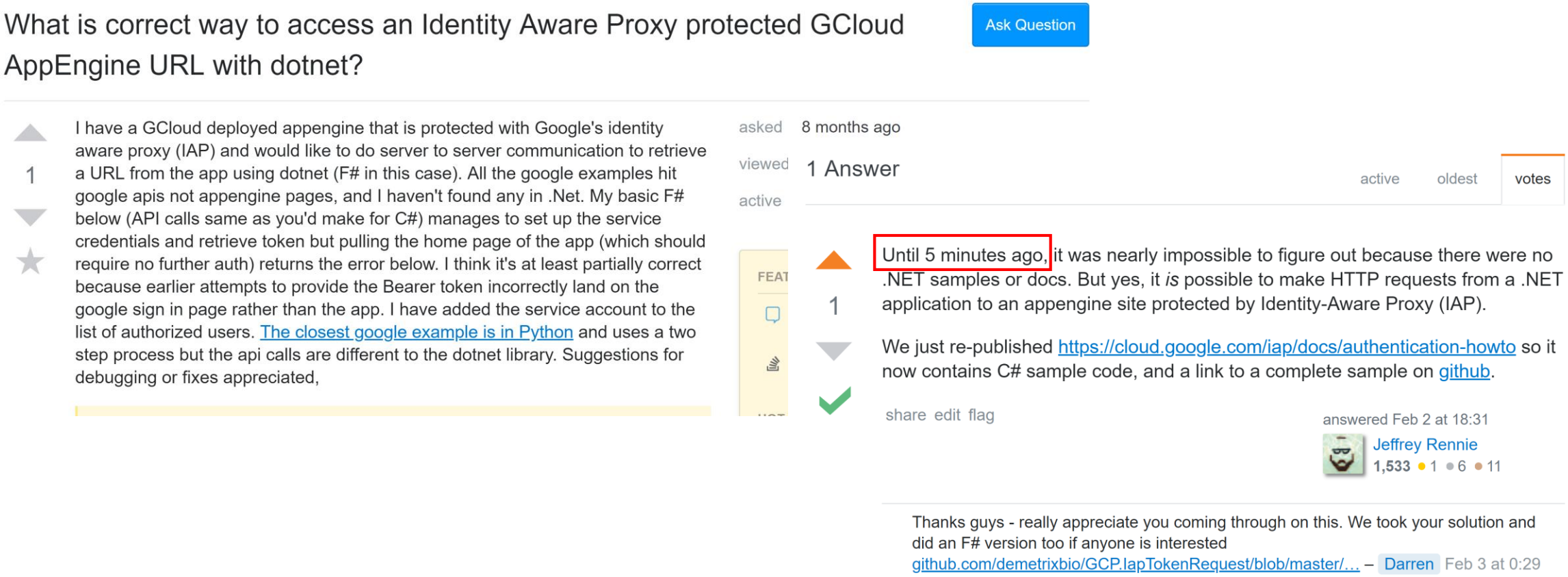
- E.g. Google Cloud Platform
  - **app.yaml** in deploy folder
  - Clitool deploy

```
runtime: aspnetcore
env: flex
service: myservice
manual_scaling:
  instances: 1
health_check:
    enable_health_check: True
    check_interval_sec: 5
    timeout_sec: 4
    unhealthy_threshold: 2
    healthy_threshold: 2
resources:
  cpu: 2
  memory_gb: 8.0
  disk_size_gb: 10
```

- Considerations:
  - Quality of docs for features you want to use
  - Access to support
    - especially for small teams
    - .NET examples
  - Degree of control
    - Platform as a service [low]
    - A la carte: be your own network engineer

Security surprisingly complicated

- Remember we want **some** people to use our services



https://github.com/giraffe-fsharp/Giraffe/blob/master/samples/GoogleAuthApp/GoogleAuthApp/Program.fs

Conclusions

- Rich type safe languages are good for web dev in biotech
- Full stack F# quite viable for delivering our type of applications
- Safe stack is inspirational
- Pay attention to error handling (more complex probably better)
- Recycle / reuse solid foundations like ASP.Net
- Dugnad Dugnad Dugnad
- Type providers for XML, CSV, SQL improve ergonomics
- Choose your cloud tooling carefully
- Cloud security is a surprising PITA
- .Net Core rocks
- We have a wonderful community, give back to it