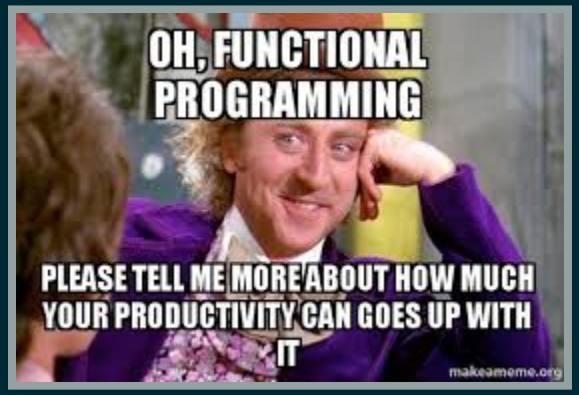
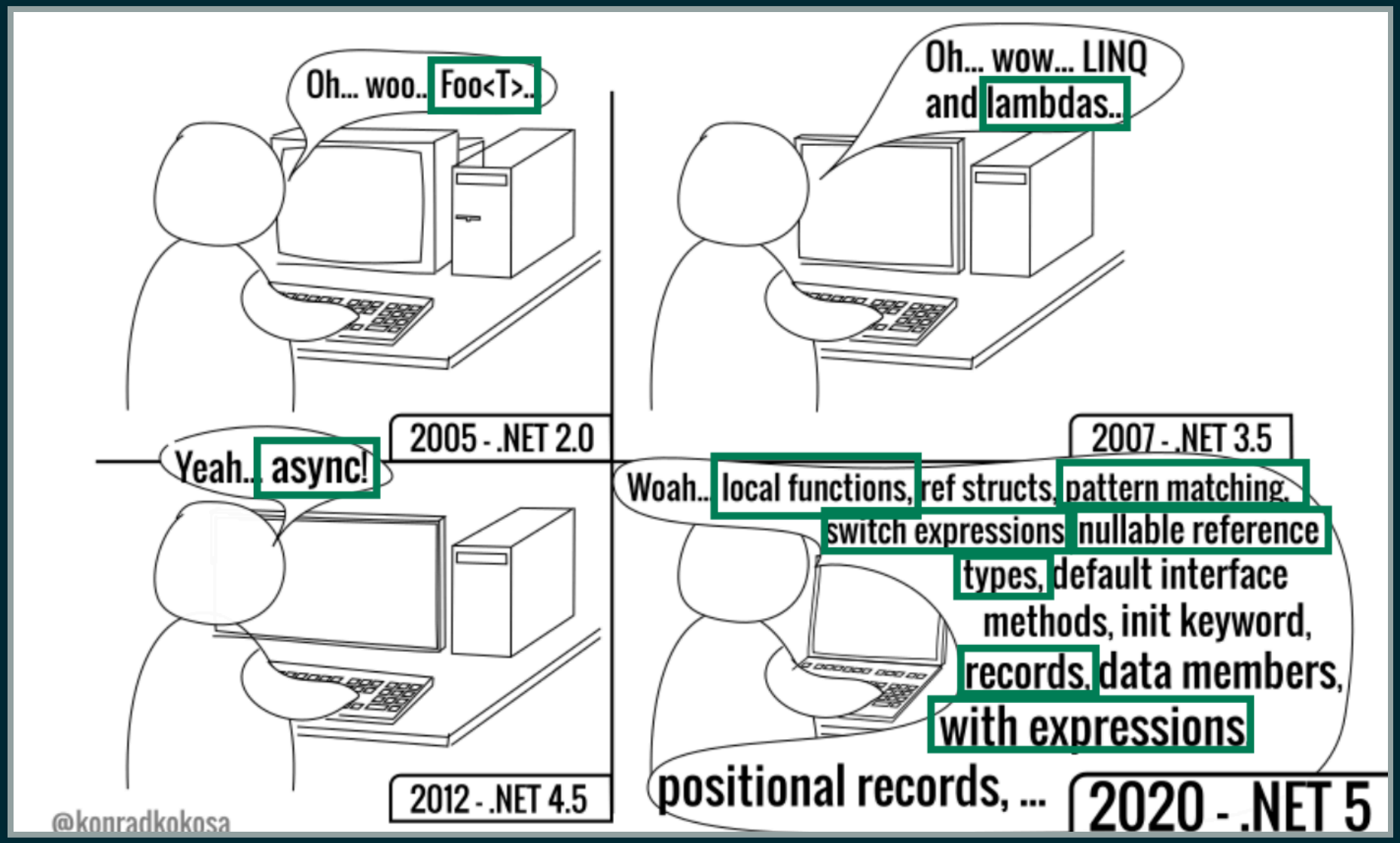
EARLY HISTORY OF FUNCTIONAL PROGRAMMING FOR .NET

- 1970-2024
- Tomas Grosup @ .NET @ DevDiv @ Microsoft



WHO CARES ABOUT FP ANYWAY?



THIS IS NOT A FULL HISTORY LESSON

- Backwards journey seen trough **selected** milestones:
  - Asynchronous programming
  - Generics
  - Early .NET days
  - Birth of F#

Peer reviewed history:
https://fsharp.org/history/hopl-final/hopl-fsharp.pdf

- (and do approach me after the talk for more!)

ASYNCHRONY - WHAT IS IT GOOD FOR?

- Continue with computation after "stuff" happens
- Let the thread do other useful work

```
function hell(win) {
 // for listener purpose
 return function() {
   loadLink(win, REMOTE_SRC+'/assets/css/style.css', function() {
     loadLink(win, REMOTE_SRC+'/lib/async.js', function() {
       loadLink(win, REMOTE_SRC+'/lib/easyXDM.js', function() {
         loadLink(win, REMOTE_SRC+'/lib/json2.js', function() {
           loadLink(win, REMOTE_SRC+'/lib/underscode.min.js', function() {
             loadLink(win, REMOTE_SRC+'/lib/backbone.min.js', function() {
               loadLink(win, REMOTE_SRC+'/dev/base_dev.js', function() {
                 loadLink(win, REMOTE_SRC+'/assets/js/deps.js', function() {
                   loadLink(win, REMOTE_SRC+'/src/' + win.loader_path + '/loader.js', function() {
                     async.eachSeries(SCRIPTS, function(src, callback) {
                       loadScript(win, BASE_URL+src, callback);
                     });
                   });
                 });
               });
             });
           });
         });
       });
     });
   });
 };
}
```

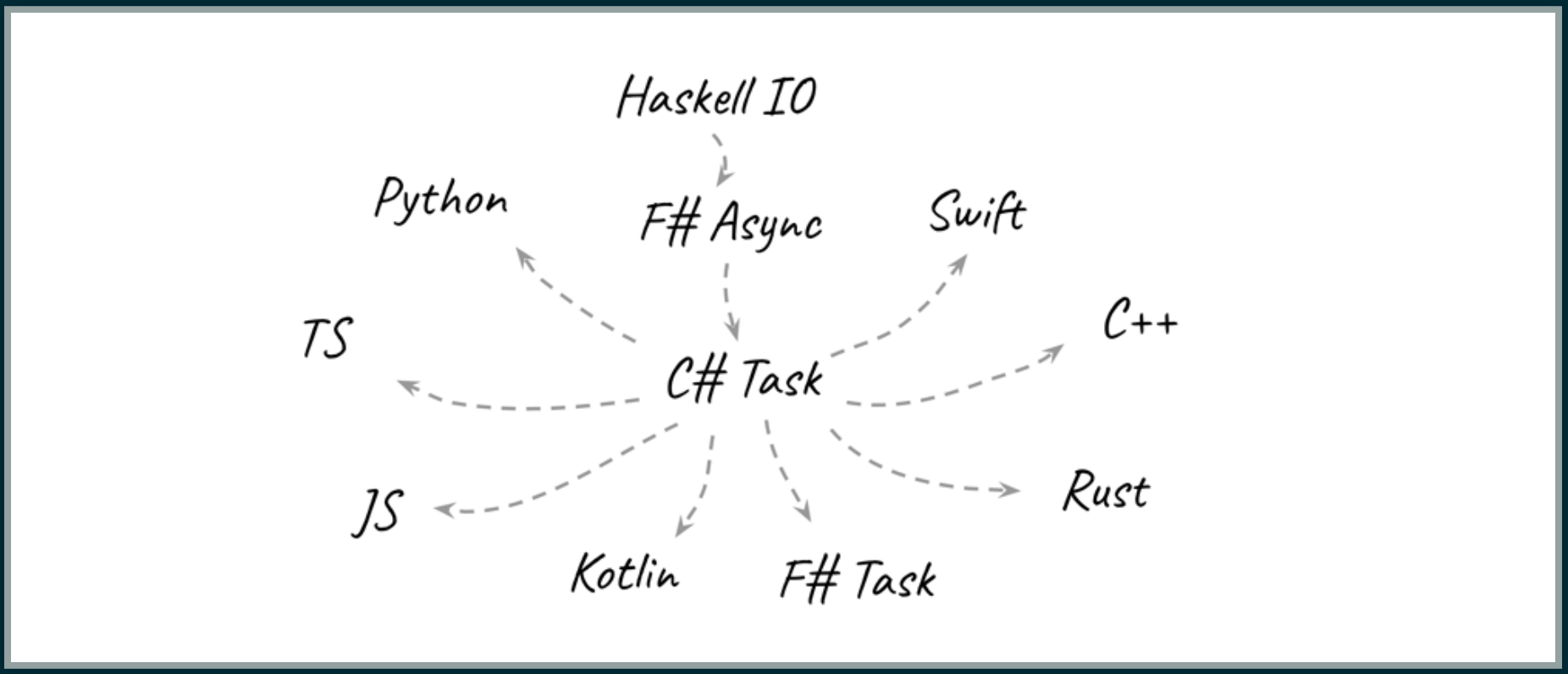
ASYNC/AWAIT (C# 5, 2012)

```
1: static async Task<string> FetchUrlAsync(HttpClient client, string url)
2: {
3:     var response = await client.GetAsync(url);
4:     response.EnsureSuccessStatusCode();
5:     return await response.Content.ReadAsStringAsync();
6: }
```

- Ease of transition sync<->async!
- 1st big feature of Mads Torgersen
- Today an industry standard

HOW IT CAME TO BE?

- Inspired by F# computation expression async{}



COMPUTATION EXPRESSIONS (F# 1.0, 2007)

```
1: let fetchUrlAsync (url:string) (client:HttpClient) =
2:     task {
3:         let! response = client.GetAsync(url)
4:         response.EnsureSuccessStatusCode()
5:         return! response.Content.ReadAsStringAsync()
6:     }
```

- Author provides a *Builder type (task,async,seq,optional,..)
- Compiler desugars bangs (let!,match!,while!,...) into calls
- builder.Bind(expression, fun result -> rest of the code)
- Nesting & Composition

GENERICS (.NET 2.0, 2005)

- Programming with data types "to be specified later"
- Key enabler for Collections, LINQ, TPL, Span

BEFORE

```
1: ArrayList list = new ArrayList();
2: list.Add(42);
3: int value = (int)list[0];
```

AFTER

```
1: List<int> list = new List<int> { 42 };
2: int value = list[0];
```

WHAT IS SO FP ABOUT GENERICS?

- Pioneered in ML in 70s
  - ML = Ancestor of F#,OCaml,Elm,Haskell,SML
- Strongly typed FP with type inference needs it
- Called 'Parametric polymorphism' in FP
  - Complements 'this' polymorphism

```
// public virtual List<T> GenericMethod(T)
var result = this.GenericMethod(x);
```

GENERICS AND TYPE INFERENCE

- Every symbol starts as a generic type parameter
- Example: Generic caching decorator

```
1: let cachedVersionOf originalFunction =
2:     let mutable cache = Map.empty
3:     fun key ->
4:         match cache |> Map.tryFind key  with
5:         | Some value -> value
6:         | None ->
7:             let value = originalFunction(key)
8:             cache <- cache |> Map.add key value
9:             value
10:
11: let expandEnvVars = cachedVersionOf System.Environment.ExpandEnvironmentV
```

HOW LANDED IN .NET?

- Don Syme from Microsoft Research
- Specifically designed (1998) to support FP languages on .NET
  - Not an accident!
  - Needed by Project 7 (wait for it.. :-))
- Unlike GJ in JVM, using JIT!

EARLY .NET

- MSFT: C,ASM,BASIC
- Loss of Java license (J++), OO wave
- COM+ 2.0 -> Lightning -> .NET
  - Regular Bill Gates reviews
  - Key decision: **Multi-language** runtime
  - +new COOL

PROJECT 7

- Approach industry and academia for 7+7 ports
  - MSR as link to academia
- COBOL, Perl, Python, Ada
- Eiffel, Mercury, **SML**, **Haskell**, **OCaml**, Scheme, **Alice**
- (OOP , Logic , ---------- ML ---------- , LISP, Concurrency)

INTEROP IS GOOD, BUT HARD

- Academical languages needed libraries, frameworks, tools
- BUT: 2 runtimes, 2 GCs ?
- Type systems compatibility
- Haskell.NET: 'Dirtyness' of .NET a problem

BIRTH OF F#

- Started as OCaml.NET by Don Syme
  - 2002
- Do a fresh .NET language with its own identity
- Main ancestor line:
  - OCaml (1996, Object features)
    - Caml (1985)
      - ML (early 70s, Generics, Strongly typed)
        - LISP (1958, GC, Recursion)

F#

```
let rec factorial = function
    | 0 -> 1
    | n -> n * factorial (n - 1)

```

ML

```
fun factorial 0 = 1
  | factorial n = n * factorial (n - 1);

```

LISP

```
(defun factorial (n)
    (if (= n 0) 1 (* n (factorial (- n 1)))))

```

Are the parens right?!

NEXT BIG THING?

- Type unions in C#?

```
type Size = Pixels of int | Percentage of float | Auto

type Option<'T> =
    | Some of 'T
    | None

type Result<'T,'TErr> =
    | Ok of 'T
    | Error of 'TErr


```

THE END

- FP found its way into mainstream
- Not the case 25y ago!
- Want to enjoy it even more?
  - https://fsharp.org/
  - Reach me at tomasgrosup @ Twitter/X/bluesky
  - T-Gro @ Github