Full-Stack Web Applications with SAFE Stack

Lambda Days

About Me

- Software Engineer since 2011
- YouTube Content Creator
- .NET, Java, JavaScript

Goals

- Not another SAFE Stack Tutorial
- Very High Level
- Focuses on the Architectural Ideas
- There is some coding involved

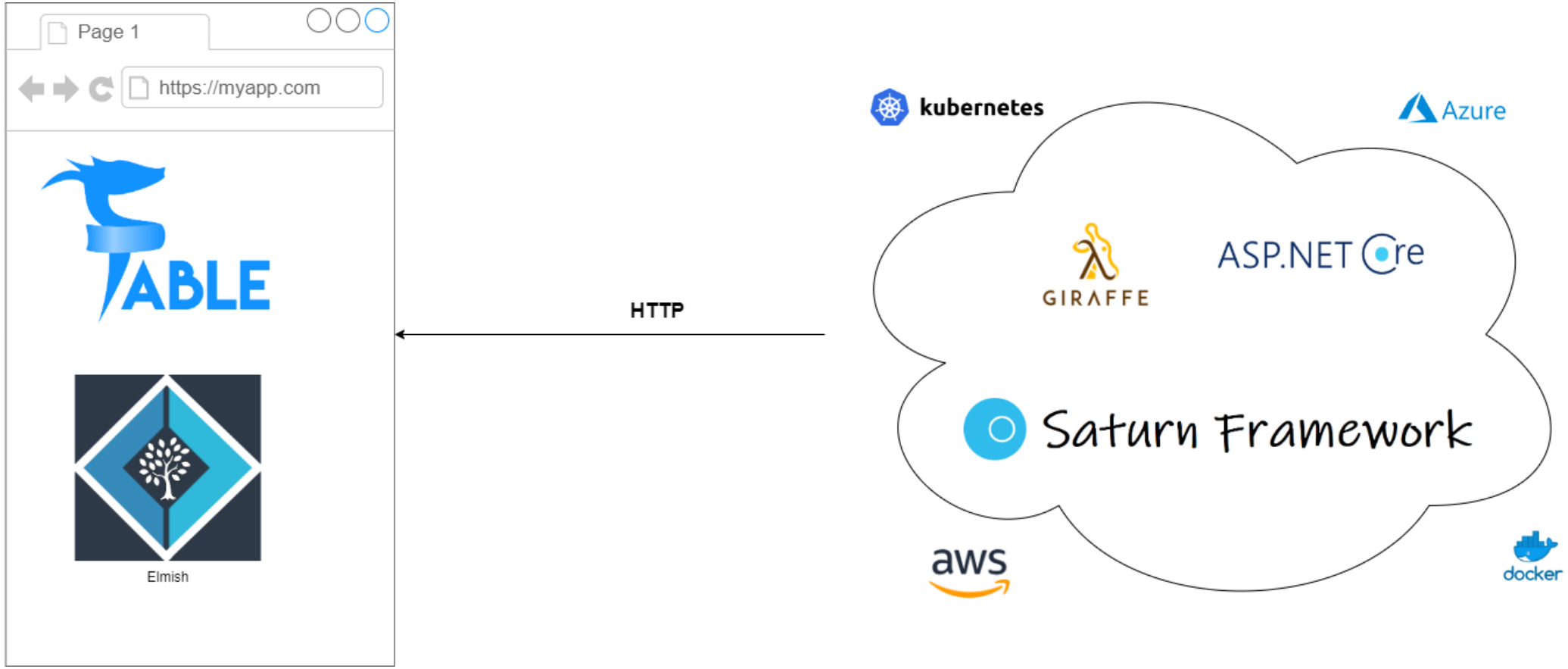
What is SAFE Stack?

- Full F# Web Stack (Client and Server)
- Functional First Architecture
- Strong Type Safety

What is the SAFE Stack?

- **S**uave/Saturn
- **A**zure/AWS
- **F**able
- **E**lmish

What is the SAFE Stack?



Server

F# as a part of .NET Ecosystem

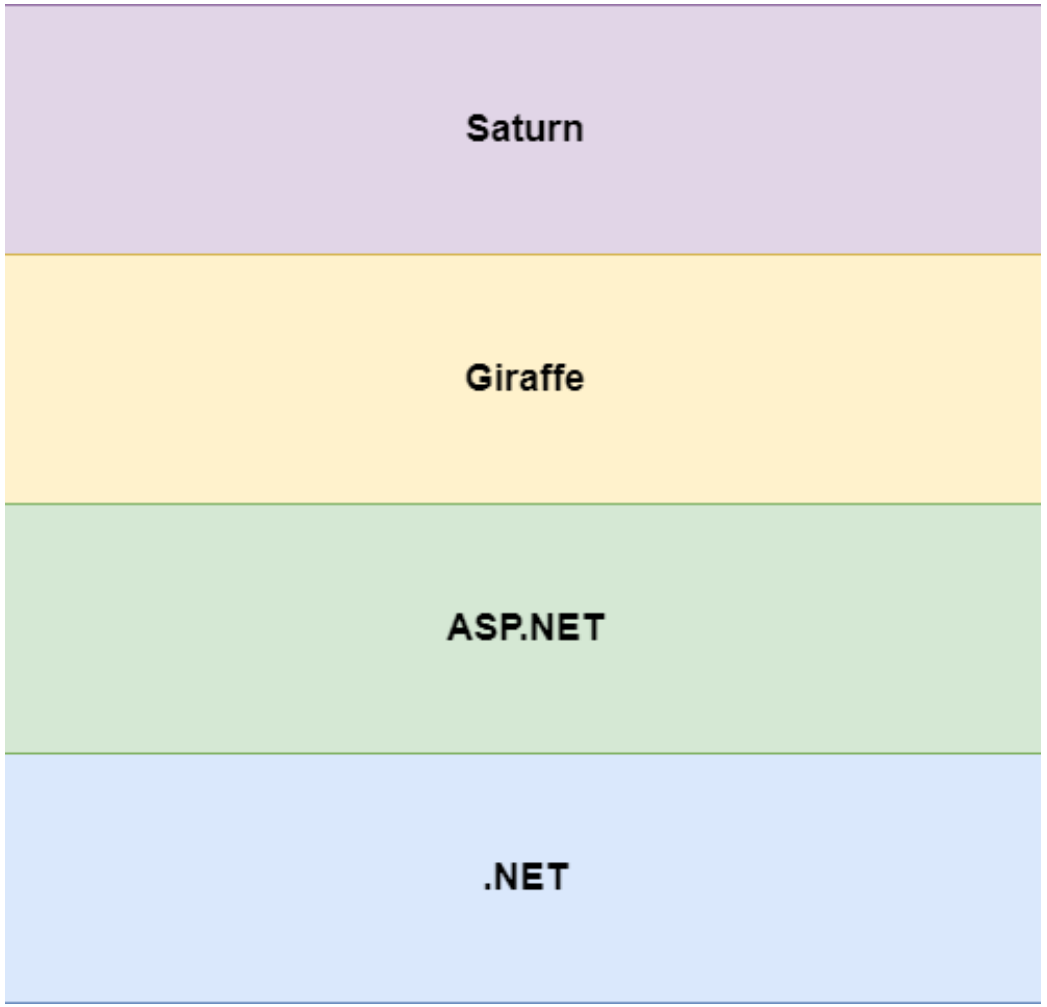
.NET Ecosystem

- Originally launched in 2002
- Was re-written in June 2016 as “.NET Core”
- Has been unified as just .NET
- Is fully cross platform and open source
- Is currently seeing a Renaissance

F# as Part of .NET

- F# is an Integral part of .NET
- Ships with the .NET SDK out of the Box
- Reuses the same High-Performance Libraries and Tools

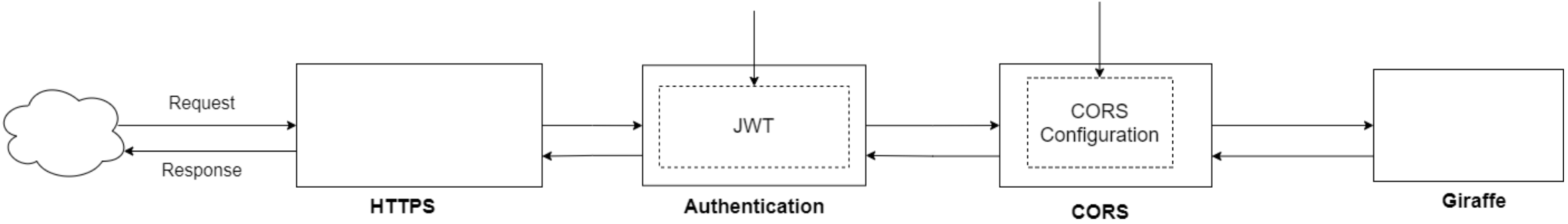
F# as Part of .NET



ASP.NET

- High Performance .NET Web Framework
- Consists of a Pipeline Middleware and Services
- Runs a Production Ready Webserver called Kestrel

ASP.NET Middleware Pipeline



Giraffe

- Simply an ASP.NET Core Middleware
- Leverages a lot of the power of ASP.NET
- Functional Architecture

Functional Architecture

HttpHandler

Web Application

```
type HttpFuncResult = Task<HttpContext option>
type HttpFunc = HttpContext -> HttpFuncResult
type HttpHandler = HttpFunc -> HttpContext -> HttpFuncResult


let sayHelloWorld : HttpHandler =
    fun (next : HttpFunc) (ctx : HttpContext) ->
        task {
            let greeting = sprintf "Hello World, from Nigeria"
            return! text greeting next ctx
        }
```

Giraffe HttpHandler

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

Handler Combination

```
let defaultView = router {
    get "/" (htmlView Index.layout)
    get "/index.html" (redirectTo false "/")
    get "/default.html" (redirectTo false "/")
}

let appRouter = router {
    not_found_handler (htmlView NotFound.layout) //Use the default 404 webpage
    forward "" defaultView //Use the default view
}

let app = application {
    error_handler (fun ex _ → pipeline { render_html (InternalError.layout ex)
})  use_router Router.appRouter
    url "http://0.0.0.0:8085/"
    memory_cache
    use_static "static"
    use_gzip
}

[<EntryPoint>]
let main _ =
    printfn "Working directory - %s" (System.IO.Directory.GetCurrentDirectory())
    run app
    0 // return an integer exit code
```

Saturn

Documentation

- ASP.NET Core – https://docs.microsoft.com/aspnet/core/
- Giraffe - https://github.com/giraffe-fsharp/Giraffe
- Saturn - https://saturnframework.org/

Demo Time

Simple Giraffe Web Application

Browser

F# as a part of JavaScript Ecosystem

JavaScript Ecosystem

- Originally launched with the early web in 2002
- Experienced a Renaissance in 2008 due to V8 and Browser Wars
- In 2009, NodeJS was created making Server-Side Application possible
- Language has gone through several revisions
  - ES1 – ES6
  - ES2016 – ES2020
- Libraries and Frameworks Evolve Rapidly
- Modern JavaScript uses Build Tools like Webpack, Rollup etc.

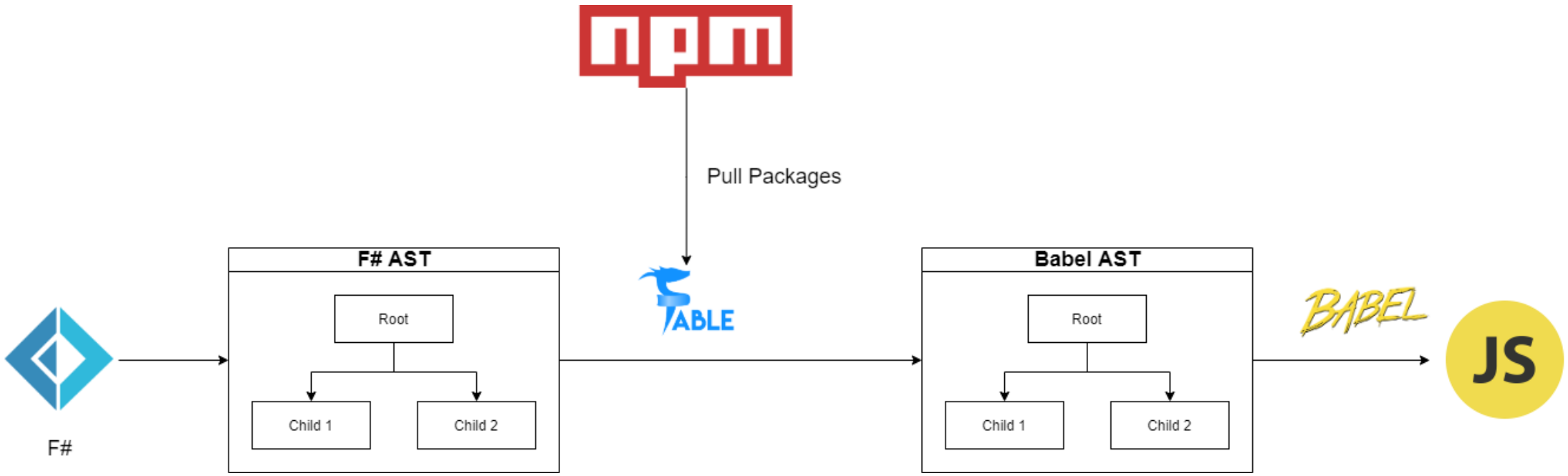
F# as Part of the JavaScript Ecosystem

- F# on the client is made possible by the Fable
- Fable is simply a Webpack plugin like LESS or SASS
- Reuses the same Client-Side Libraries and Tools like Webpack
  - Tree Shaking
  - Hot Module Replacement (Live Reload)
  - Minification
  - CSS Preprocessor etc.
- Web Assembly is also supported via Bolero Project
- Fable is also compatible with NodeJS

Fable

- Converts F# to JavaScript
- Uses the Babel JavaScript Compiler
- Shims out .NET APIs
  - Some API are replaced with JavaScript APIs e.g., Date
  - Some APIs are reimplemented in JavaScript e.g., Async
  - .NET Library Shims for Native JavaScript APIs
- Fully compatible with Existing NPM packages

FABLE Transpilation



FABLE Configuration

```
{
    "private": true,
    "scripts": {
        "start": "webpack-dev-server"
    },
    "dependencies": {
        "@babel/core": "^7.8.4",
        "fable-compiler": "^2.4.15",
        "fable-loader": "^2.1.8",
        "react": "^16.12.0",
        "react-dom": "^16.12.0",
        "webpack": "^4.41.6",
        "webpack-cli": "^3.3.11",
        "webpack-dev-server": "^3.10.3"
    }
}
```

webpack.config.js

```
var path = require("path");

module.exports = {
    mode: "development",
    entry: "./src/App.fsproj",
    output: {
        path: path.join(__dirname, "./public"),
        filename: "bundle.js",
    },
    devServer: {
        publicPath: "/",
        contentBase: "./public",
        port: 8080,
    },
    module: {
        rules: [{
            test: /\.fs(x|proj)?$/,
            use: "fable-loader"
        }]
    }
}
```

package.json

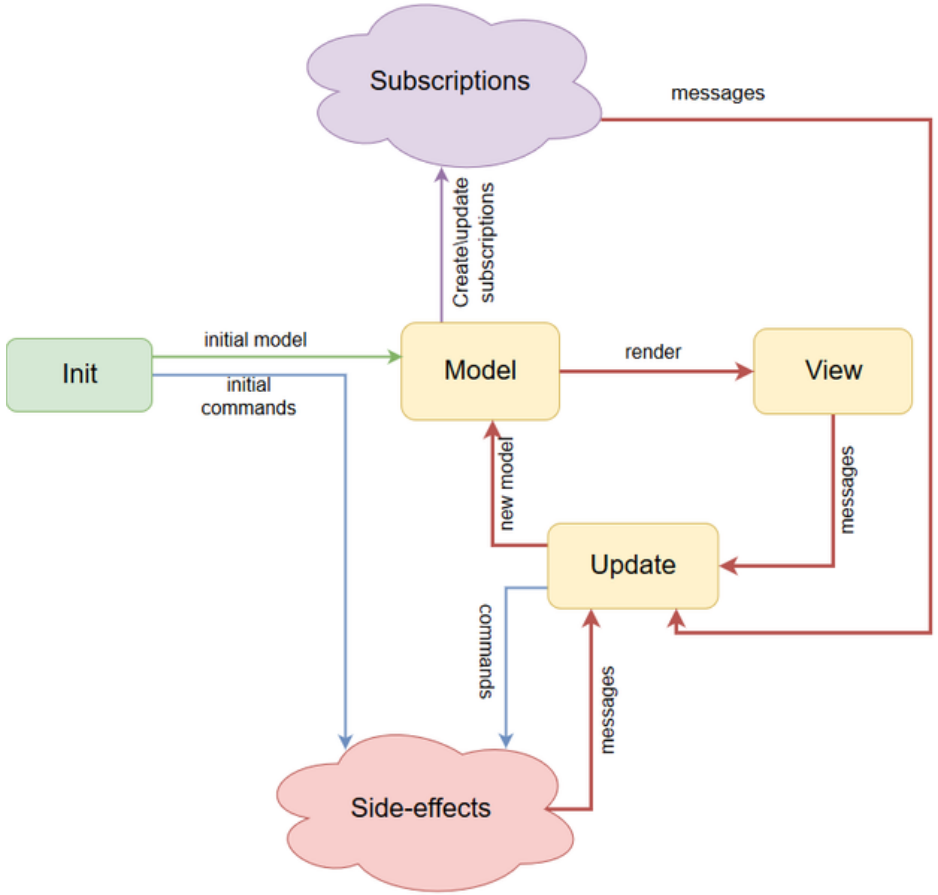
Demo Time

Simple Fable Application

Elmish

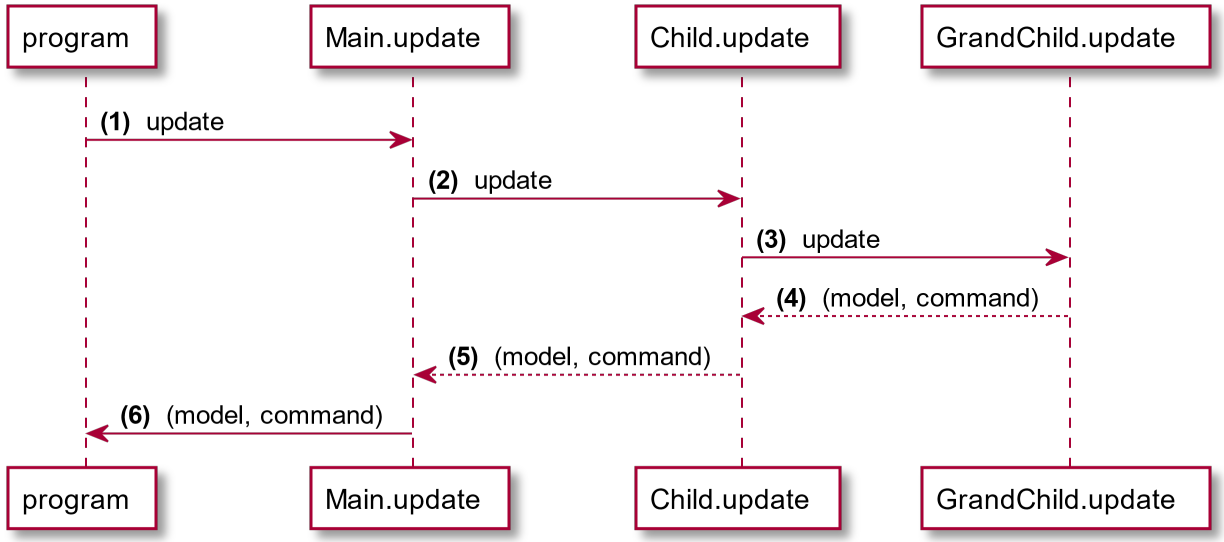
- Single Page Application Framework
- Leverages the ELM Architecture
- Uses React under the Hood
- Model-View-Update Pattern
  - Init – Creates the model
  - Update – Replaces the model in response to an Event (Message)
  - View – Renders the UI

Elm Architecture



Source: https://steemit.com/utopian-io/@tensor/using-the-elm-architecture-or-the-mvu-pattern-with-dartea-inside-of-dart-s-flutter-framework

Elmish Composition



Source: https://elmish.github.io/elmish/

Documentation

- Fable – https://fable.io/docs/
- Elmish Book - https://zaid-ajaj.github.io/the-elmish-book/

SAFE Stack

Putting it all together with a nice bow on top

SAFE Stack

- Install using *dotnet new -i SAFE.Template*
- Documentation available at https://safe-stack.github.io/
- Enterprise Support Available by Compositional.IT

Demo Time

SAFE Stack Example – Edelwiess Data

Thank You

🦤@odytrice

📺youtube.com/odytrice