GROUND UP SERIES

Syed Awase Khirni

RESEARCHER | ENTREPRENEUR | TECHNOLOGY COACH

@sak008 | sak@sycliq.com/sak@territorialprescience.com | +91. 9035433124

Syed Awase earned his PhD from University of Zurich in GIS, supported by EU V Framework Scholarship from SPIRIT Project (www.geo-spirit.org). He currently provides consulting services through his startup www.territorialprescience.com and www.sycliq.com . He empowers the ecosystem by sharing his technical skills worldwide, since 2008.

Terms of Use

- You shall not circulate these slides without written permission from Territorial Prescience Research I Pvt ltd.
- If you use any material, graphics or code or notes from these slides, you shall seek written permission from TPRI and acknowledge the author Dr. Syed Awase Khirni
- If you have not received this material, post-training session, you shall destroy it immediately and not use it for unauthorized usage of the material. If any of the material, that has been shared is further used for any unauthorized training by the recipient, he shall be liable to be prosecuted for the damages. Any supporting material that has been provided by the author, shall not be used directly or indirectly without permission.
- If this material, has been shared to any organization prior to the training and the organization does not award the contract to TPRI, it should not use the training material internally. If by any chance, the organization is using this training material without written permission, the organization is liable to pay for the damages to TPRI and is subjected to legal action, jurisdiction being Bangalore.
- Any unauthorized usage, TPRI has right to claim damages ranging from USD 50000 to USD 10,0000 dollars as damages.
- Any organization, which does not intend to go ahead with training or does not agree with the terms and conditions, should destroy the material from its network immediately. The burden of proof lies on the client, with whom this material has been shared.
- Recovery of the damages and legal fees will be born by the client organization/candidate/party, which has violated the terms and conditions.
- Only candidates who have attended the training session in person from Dr. Syed Awase Khirni, TPRI are entitled to hold this training material. They cannot further circulate it, or use it or morph it, or change it to provide trainings.
- TPRI reserves all the rights to this material and code plays and right to modify them as and when it deems fit.
- If you agree with the terms and conditions, please go ahead with using the training material. Else please close and destroy the slide and inform TPRI immediately

Slide Version Updates

Last Updated	Version	Release Date	Updated by	Code Plays Done @

Program Agenda

DAY 1	DAY 2	DAY 3	DAY 4

DAY 5	DAY 6	DAY 7	DAY 8

SYED AWASE KHIRNI

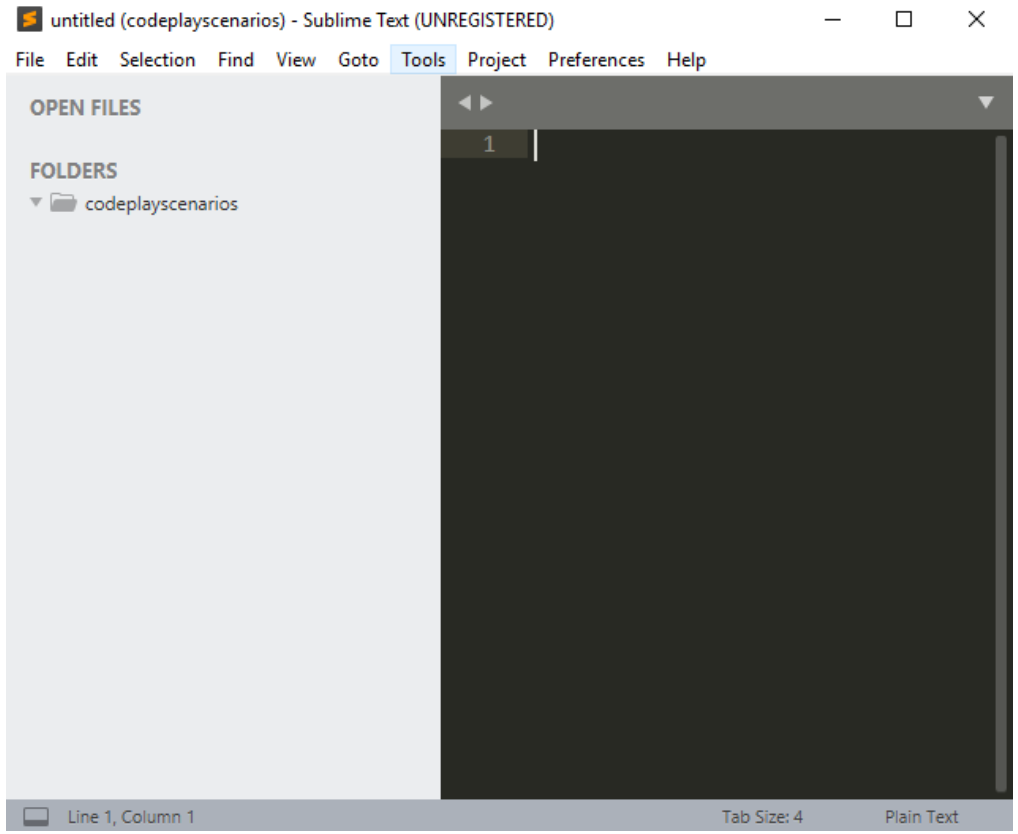
ELM

Original Series

- A Typed functional programming language with focus on immutability and reactive programming for creating web-browser apps.
- Designed by Evan Czaplicki in 2012.
- https://elm-lang.org
- Application Development Framework

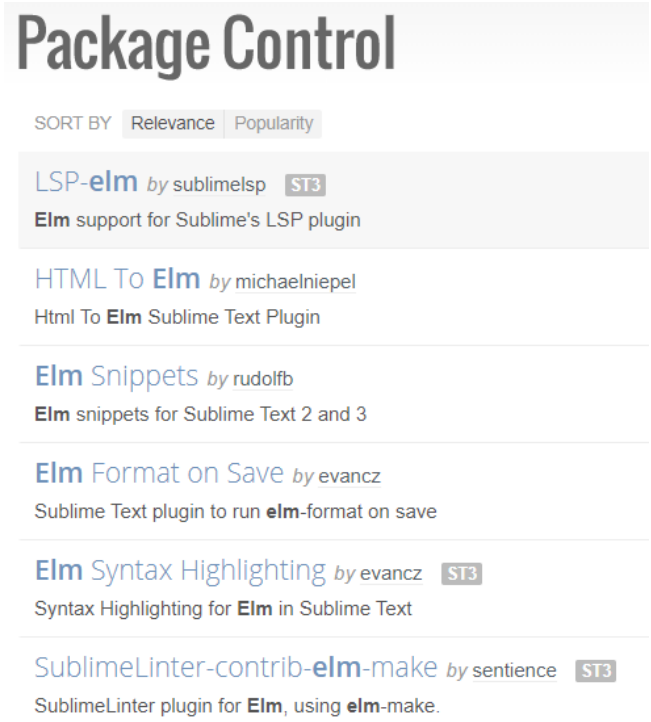
ELM INSTALLATION

Sublime Editor



Editor Plugins

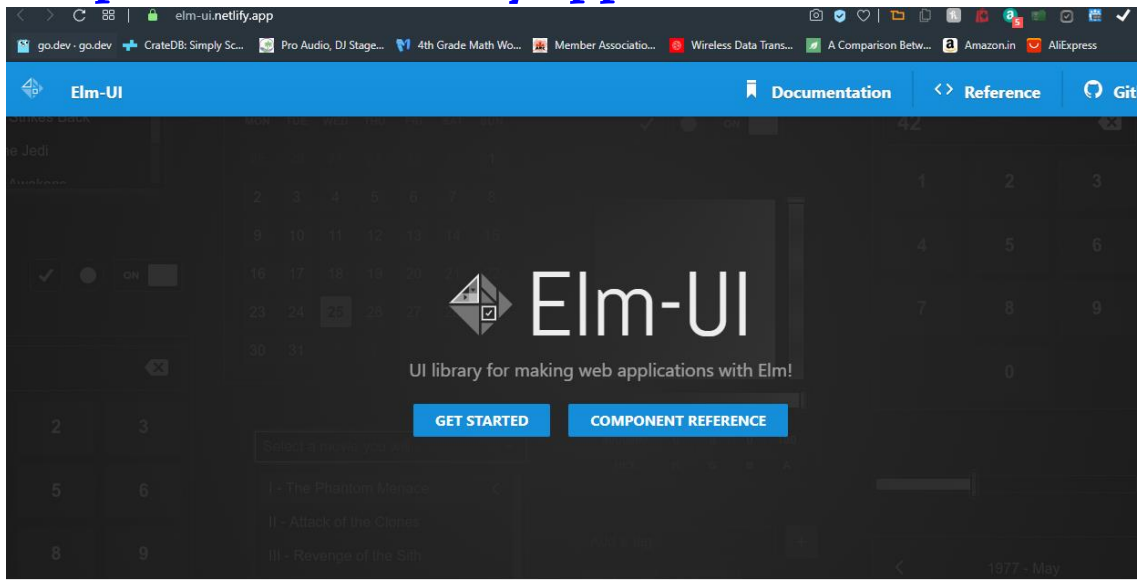
https://packagecontrol.io/search/elm



ELM

- Typed **pure functional** programming language
- Static type checking
- Automatic Semantic Versioning
- Immutable
- Interoperability with javascript
- Virtual DOM used to update the state
- Rich Error Messages
- Zero Runtime Exceptions
- No javascript fatigue.
- Having ELM compile to WebAssembly as a target could improve the application performance.

https://elm-ui.netlify.app



ELM ARCHITECTURE

Every ELM program will breakup into 3 cleanly separated parts:

- Model : canonical state of the application
- Update: a way to update the state of the application.
- View: a way to render view state as HTML (projecting the state)

WEB ASSEMBLY

https://webassembly.org

- WebAssembly is a binary instruction format for a stack-based virtual machine.
- It is designed as a portable target for compilation of high-level languages like C/C++/Rust, enabling deployment on the web for client and server application.s

ELM Users

http://builtwithelm.co

noredink

gizra

CircuitHub

TruQu

Prezi

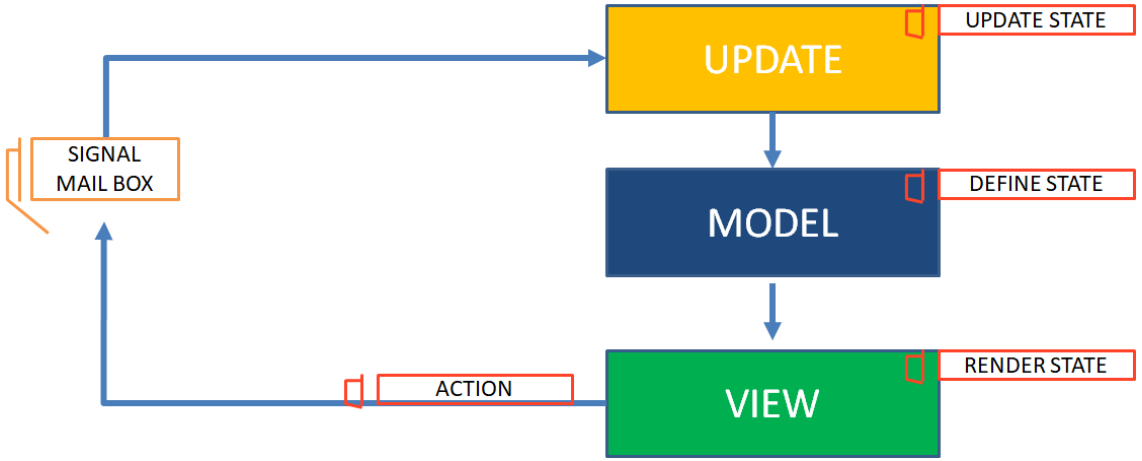
futurice

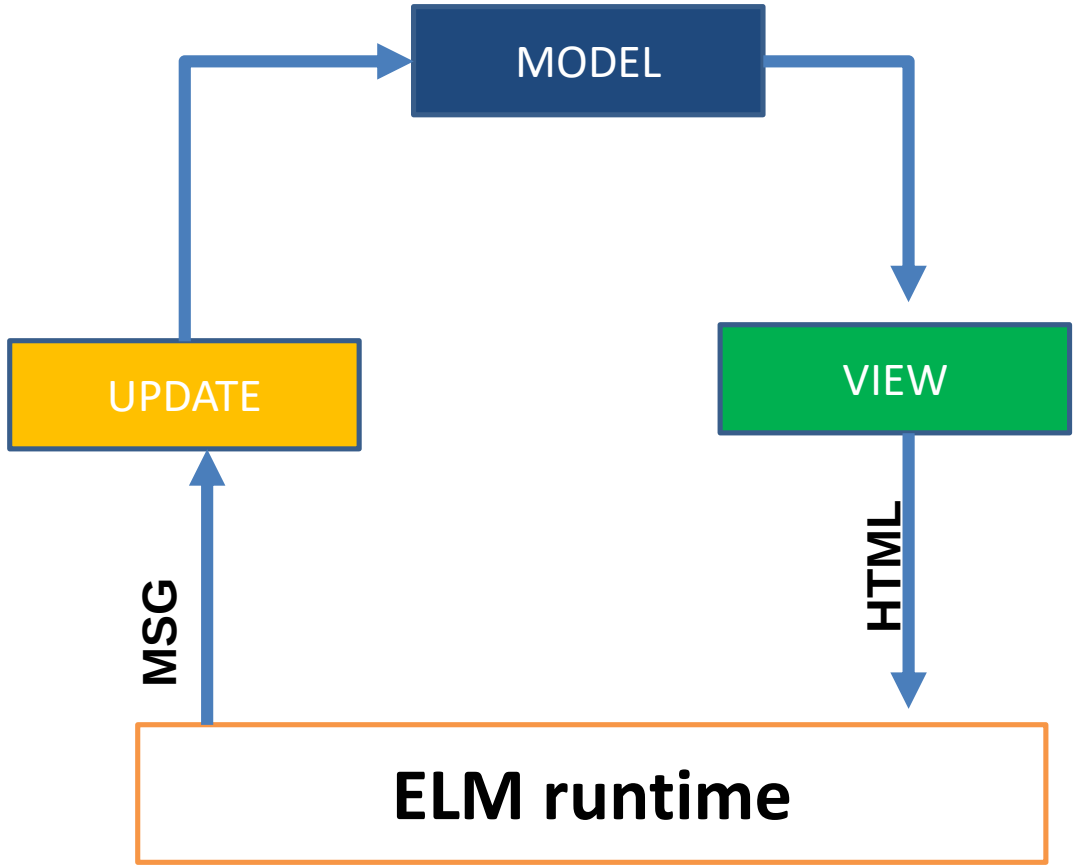
BEAUTIFUL DESTINATIONS

ELM RESOURCE

RESOURCE	DESC	Example
Elm-repl	REPL or ELM console	$elm-repl
Elm-reactor	Development Compiler and server	$elm-reactor –address=0.0.0.0
Elm-make	Elm compiler	$ elm Main.elm –output =main.html
Elm-package	Elm package manager	$elm-package install elm-lang/http
Online editor	Elm online editor	https://elm-lang.org/try

ELM ARCHITECTURE





ELM Commands

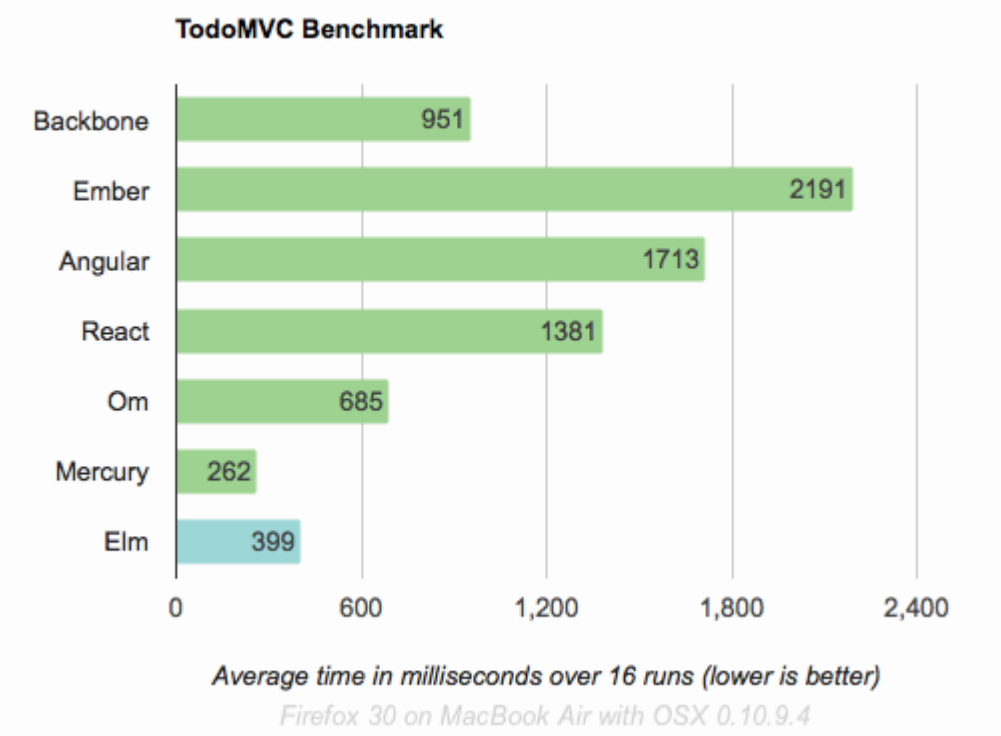
Elm.exe init	
Elm.exe repl	
Elm.exe reactor	https://github.com/elm-lang/elm-reactor
Elm.exe make	
Elm.exe bump	
Elm.exe diff	
Elm.exe publish	

ELM PRIMITIVES: DATA TYPES

TYPES	DESCRIPTION	EXAMPLE
String	string data type	“Syed Awase Khirni”
Integer	Integer data type	12368
Float	Float data type	6.626070
List	List data type	[123,36,37,28]
Tuple	Tuple data type	(383,2.48,”Syed Awase Khirni”)
Record	Record data type	{wahid=1, ithana=2.0, tisa=3.0, araba=4.0}
Function	Function	isEven n = (n %2) == 1

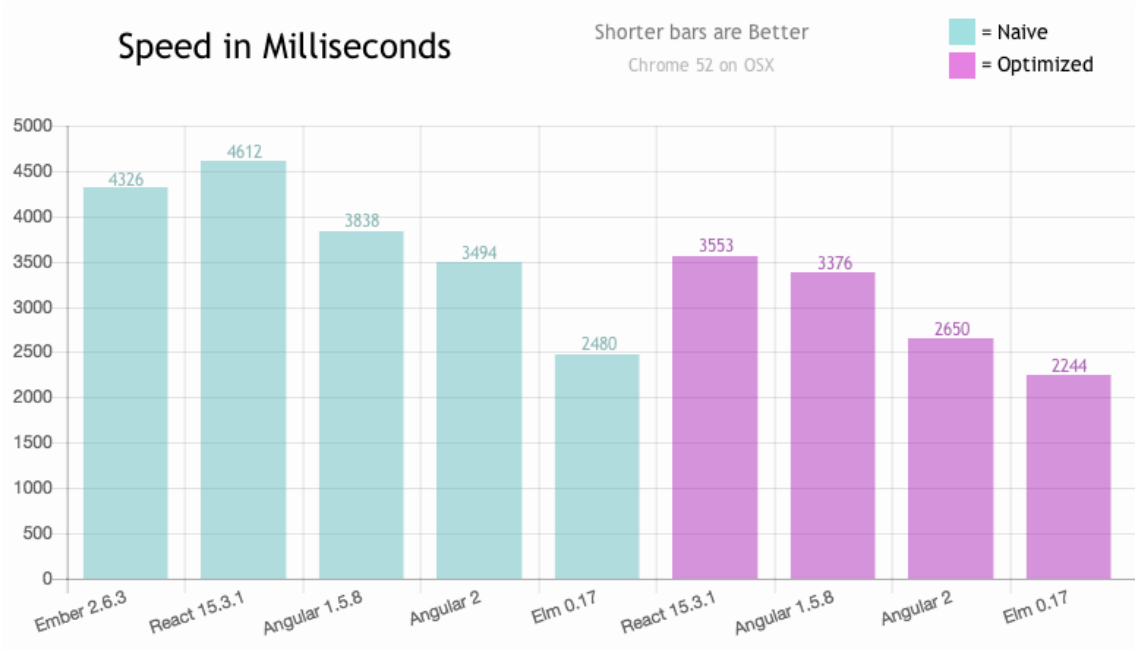
PERFORMANCE BENCHMARKS

[Evan Czaplicki](#) / 23 July 2014



https://elm-lang.org/news/blazing-fast-html

by [Evan Czaplicki](#) / 30 Aug 2016



https://elm-lang.org/news/blazing-fast-html-round-two

ELM OPERATIONS

DATA TYPE	OPERATION	EXAMPLE
String	concatenation	“I” ++ “love” ++ “You”
Integer	subtraction	-
Integer	Division	/
Integer	Integer division	//
Integer	exponentiation	^
Integer	Less than	<
Integer	Greater than	>
Integer	Equal to	==
Integer	Not Equal to	/=

ELM function construct

functionName : Float -> Float

functionName *argument* =

functionBody

Type definitions for function input and return type

Last statement in function body is return

```
1  import Html exposing (text)
2
3  main =
4   text (computeTax 75000 18)
5
6
7  printMessage =
8   "Hello! Syed Awase "
9
10
11 computeTax amount taxrate =
12  Debug.toString (amount * taxrate / 100)
13
```

Older versions of elm use toString while the newer version Debug.toString ()

ELM function construct


```
import Html exposing (text)

main =
 text (computeInterest 1000 3)

printMessage =
 "Hello! Syed Awase "


computeTax amount taxrate =
 Debug.toString (amount * taxrate / 100)

computeInterest amount interestrate =
 amount * interestrate / 100
 |> Debug.toString
```

```
import Html exposing (text)

main =
 computeInterest 1000 3
 |> text

printMessage =
 "Hello! Syed Awase "


computeTax amount taxrate =
 Debug.toString (amount * taxrate / 100)

computeInterest amount interestrate =
 amount * interestrate / 100
 |> Debug.toString
```

ELM function construct with type signatures

```
import Html exposing (text)

main =
 computeInterest 1000 3
 |> text

printMessage =
 "Hello! Syed Awase "

computeTax: Float -> Float -> String
computeTax amount taxrate =
 Debug.toString (amount * taxrate / 100)

computeInterest: Float -> Float -> String
computeInterest amount interestrate =
 amount * interestrate / 100
 |> Debug.toString


computeRatio: Float -> Float -> String
computeRatio divisor dividend =
 dividend / divisor
  |> Debug.toString


isdivisibleBy3: Float -> String
isdivisibleBy3 =
 computeRatio 3
```

Function currying

SECTION:

ELM CODE PLAY

SECTION:

EX01: SETTING UP VSCODE FOR ELM

Steps to create elm project

Setting up and Installing: Elm 0.19

EXERCISE DEMO: 1.1

LEARNING OUTCOMES

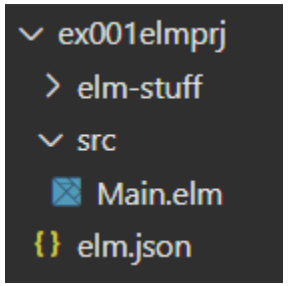
Creating ELM Project

- Create ELM project following the step outlined here.

1- create elm project

```
PS E:\awase-harddisk\CT\elm\codeplayscenarios\ex001elmprj> elm.exe init
```

2- elm project structure



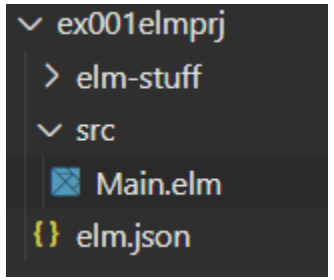
3-Running your elm project

```
PS E:\awase-harddisk\CT\elm\codeplayscenarios\ex001elmprj> elm.exe reactor
Go to http://localhost:8000 to see your project dashboard.
```

Step 2:

Understanding the ELM project structure

- Lets look at the ELM project structure
  - Upon executing the command
    - Elm.exe init
  - It creates a folder structure
    - Src
    - Elm.json

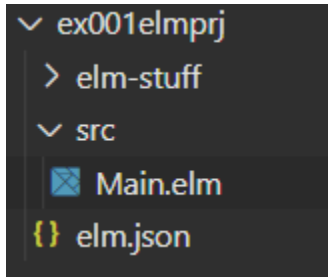


Elm.json

```
ex001elmprj > {} elm.json > {} dependencies > {} indirect > abc elm/virtual-dom
{
    "type": "application",
    "source-directories": [
        "src"
    ],
    "elm-version": "0.19.1",
    "dependencies": {
        "direct": {
            "elm/browser": "1.0.2",
            "elm/core": "1.0.5",
            "elm/html": "1.0.0"
        },
        "indirect": {
            "elm/json": "1.1.3",
            "elm/time": "1.0.0",
            "elm/url": "1.0.0",
            "elm/virtual-dom": "1.0.2"
        }
    },
    "test-dependencies": {
        "direct": {},
        "indirect": {}
    }
}
```

Understanding the ELM project structure

- Lets look at the ELM project structure
  - Upon executing the command
    - Elm.exe init
  - It creates a folder structure
    - Src
    - Elm.json

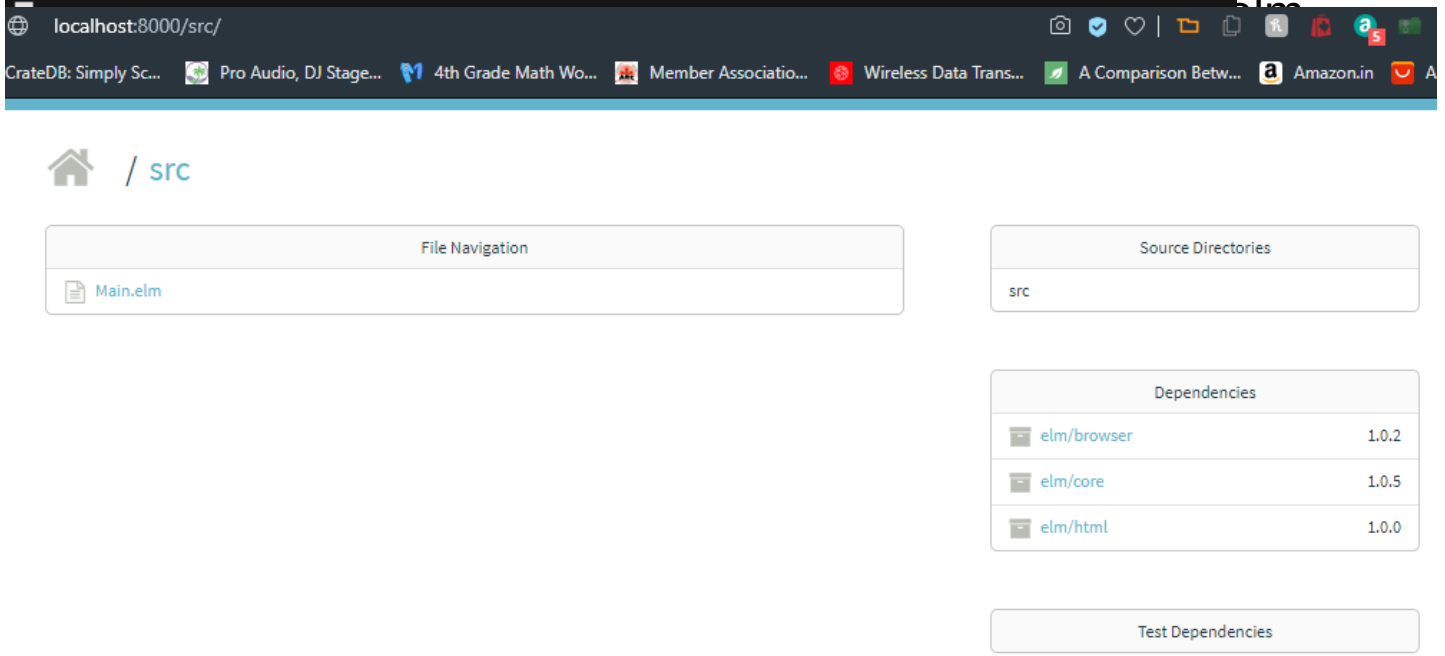


Main.elm

```
module Main exposing (..)
-- Syed Awase Khirni SYCLIQ/TPRI www.sycliq.com www.territorialprescience.com
import Html exposing (text)
-- this is a comment in elm
main =
 computeInterest 1000 3
 |> text
--function type signatures
computeTax: Float -> Float -> String
computeTax amount taxrate =
 Debug.toString (amount * taxrate / 100)
--function type signatures
computeInterest: Float -> Float -> String
computeInterest amount interestrate =
 amount * interestrate / 100
 |> Debug.toString
--function type signatures
computeRatio: Float -> Float -> String
computeRatio divisor dividend =
 dividend / divisor
  |> Debug.toString
-- function currying
isdivisibleBy3: Float -> String
isdivisibleBy3 =
 computeRatio 3
```

Step 3:

```
PS E:\awase-harddisk\CT\elm\codeplayscenarios> elm.exe reactor
Go to http://localhost:8000 to see your project dashboard.
```



Running the elm project

- Lets execute our work and check it out
- => Elm.exe reactor

30