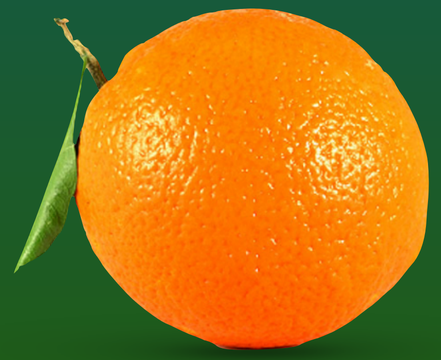
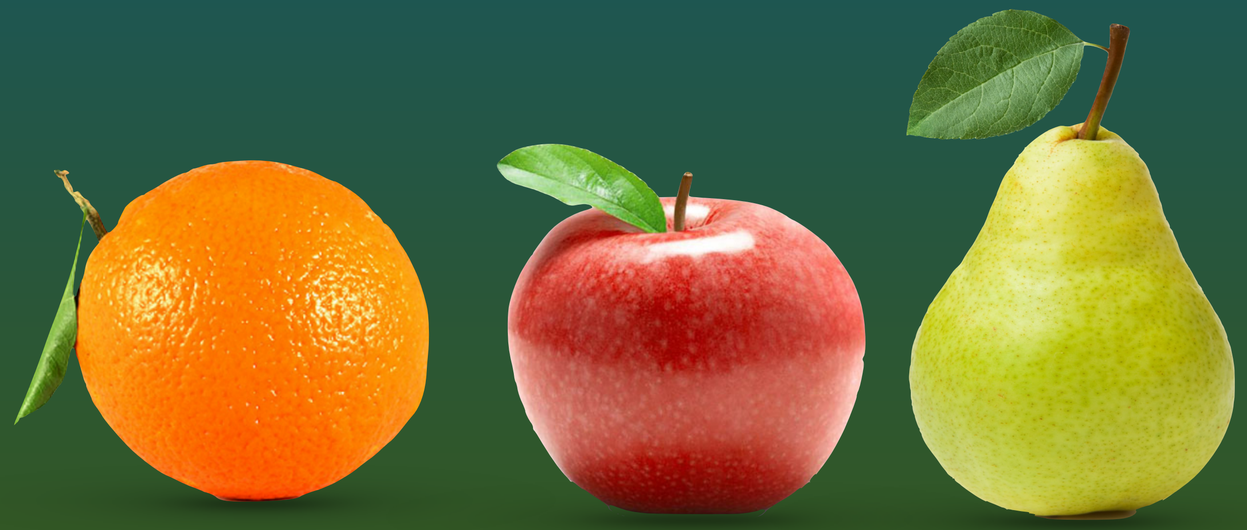
TypeScript, PureScript and Elm for functional JavaScript

Prof. D. König

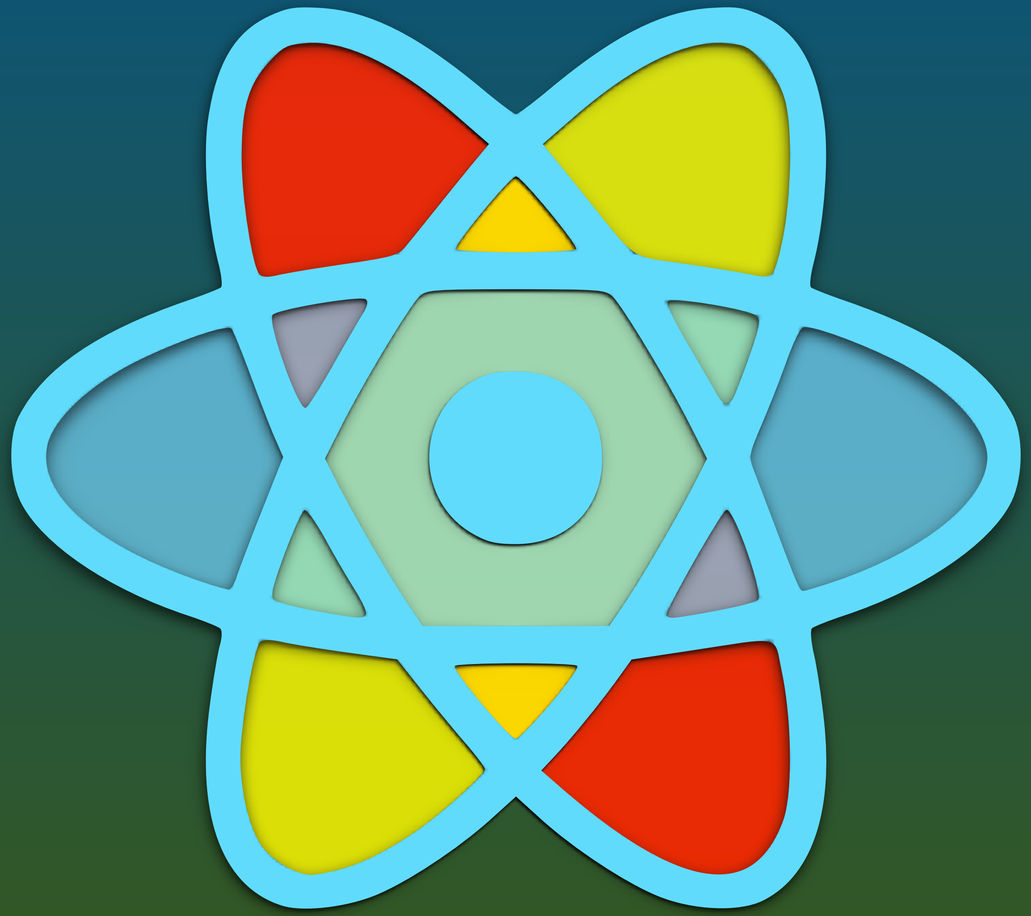


What they have in common

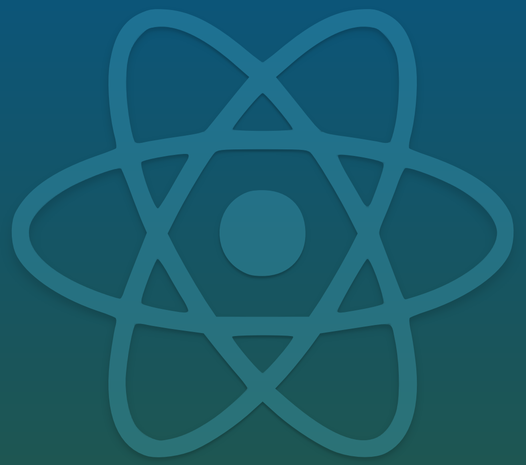
Transpiling to JavaScript



What they have in common



What they have in common

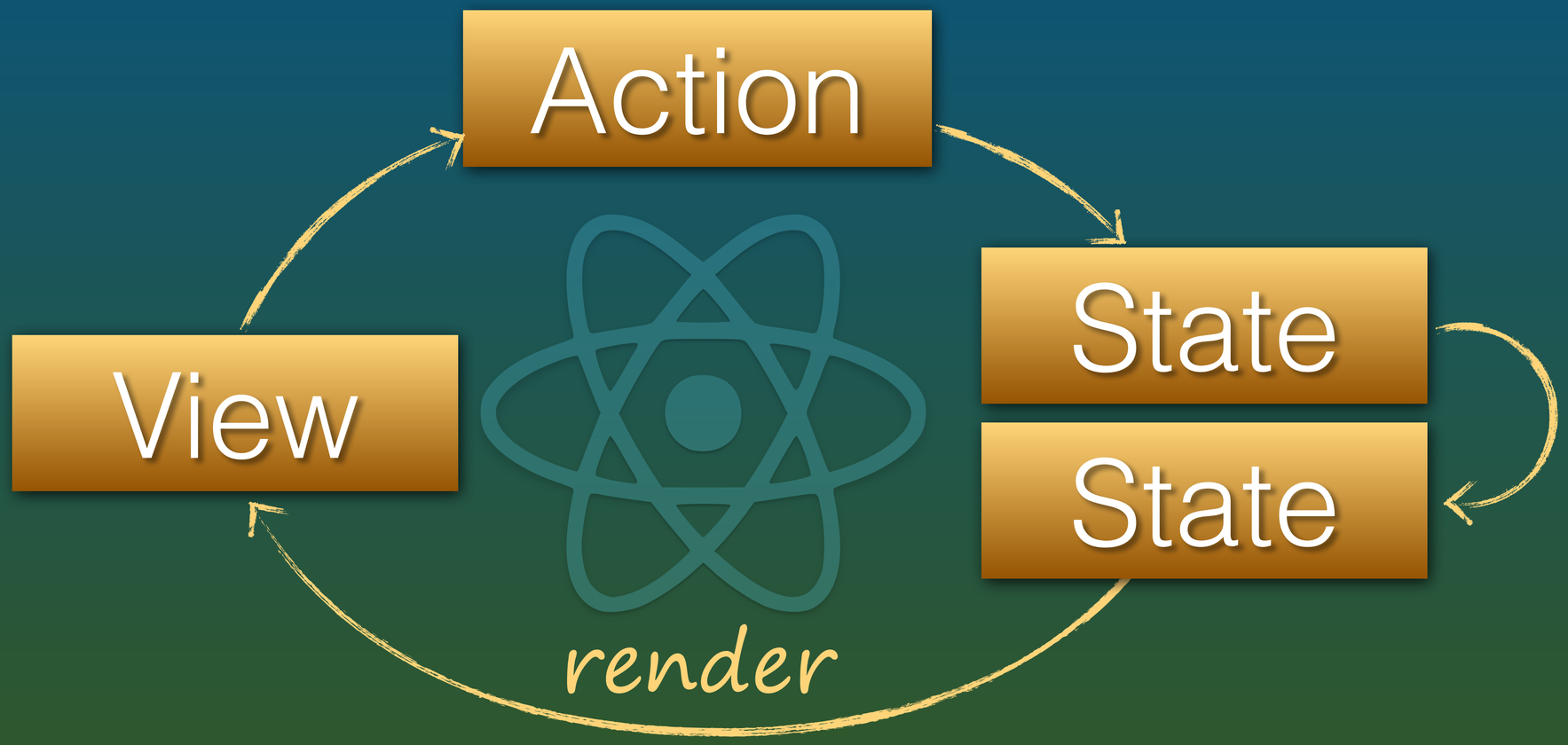


Allow React/Elm Architecture

for functional JavaScript/SPA

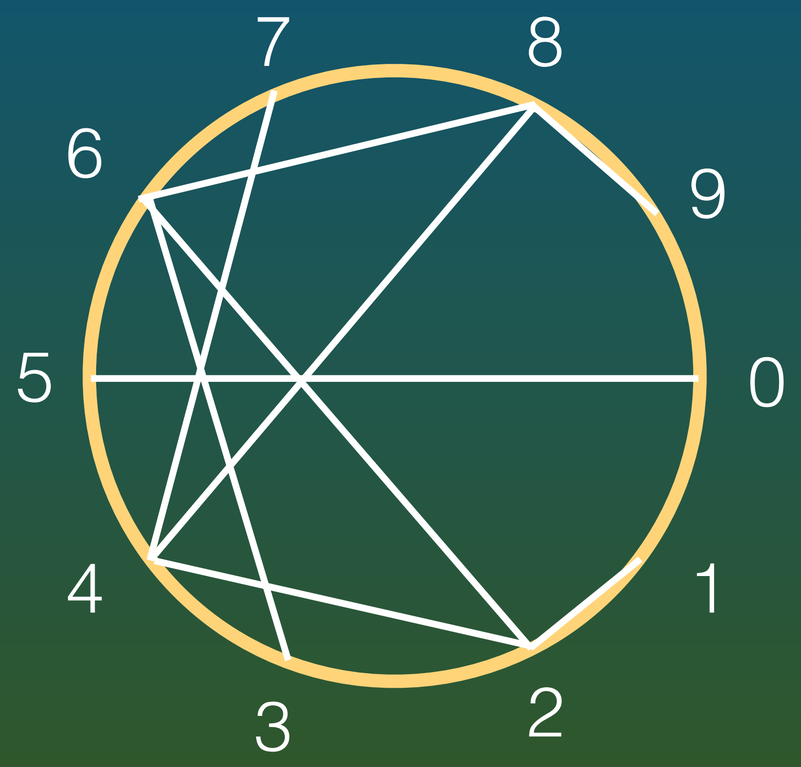
- TypeScript + React
- PureScript + Pux / Halogen
- Elm

The Cycle



Example: Multiplication Table

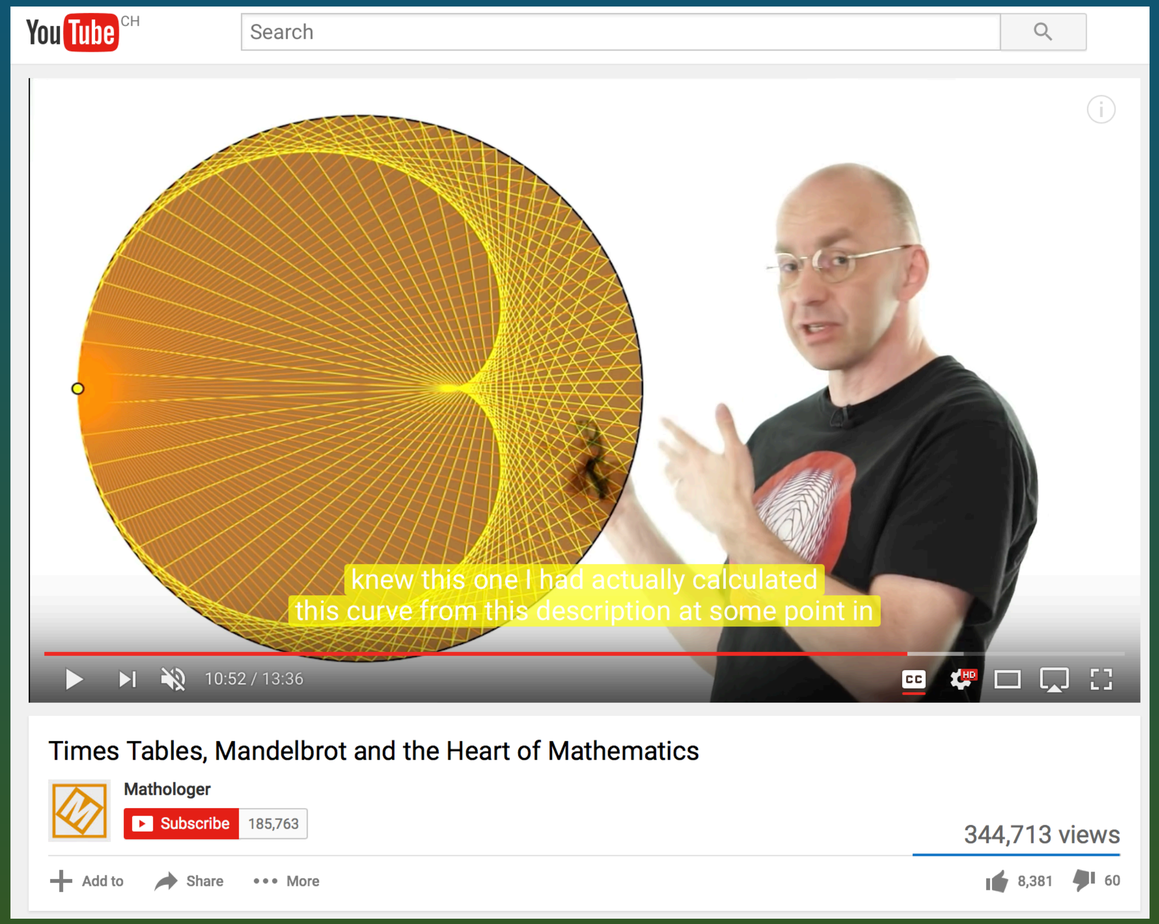
n	* 2
1	2
2	4
3	6
4	8
5	10
6	12
7	14
8	16
9	18



Demo

https://dierk.github.io/tryPux/beautifulMathInTS/index.html

Times Tables, Mandelbrot and the Heart of Mathematics



https://www.youtube.com/watch?v=qhbuKbxJsk8

TypeScript

Typed state

Actions as functions/lambdas

State immutability with discipline

Object/component abstraction

Elm

Typed state

Action type with values

State is immutable

Update function is a fold

Function composition

PureScript / Pux

Like Elm

but even more Haskell-ish

Calling JavaScript (FFI)

TypeScript	*Type declaration*
PureScript	*Type declaration*
Elm	*Port/Flag*

Applicability

TypeScript	*JS Environment*
PureScript	*JS Environment*
Elm	*Browser*

Paradigm

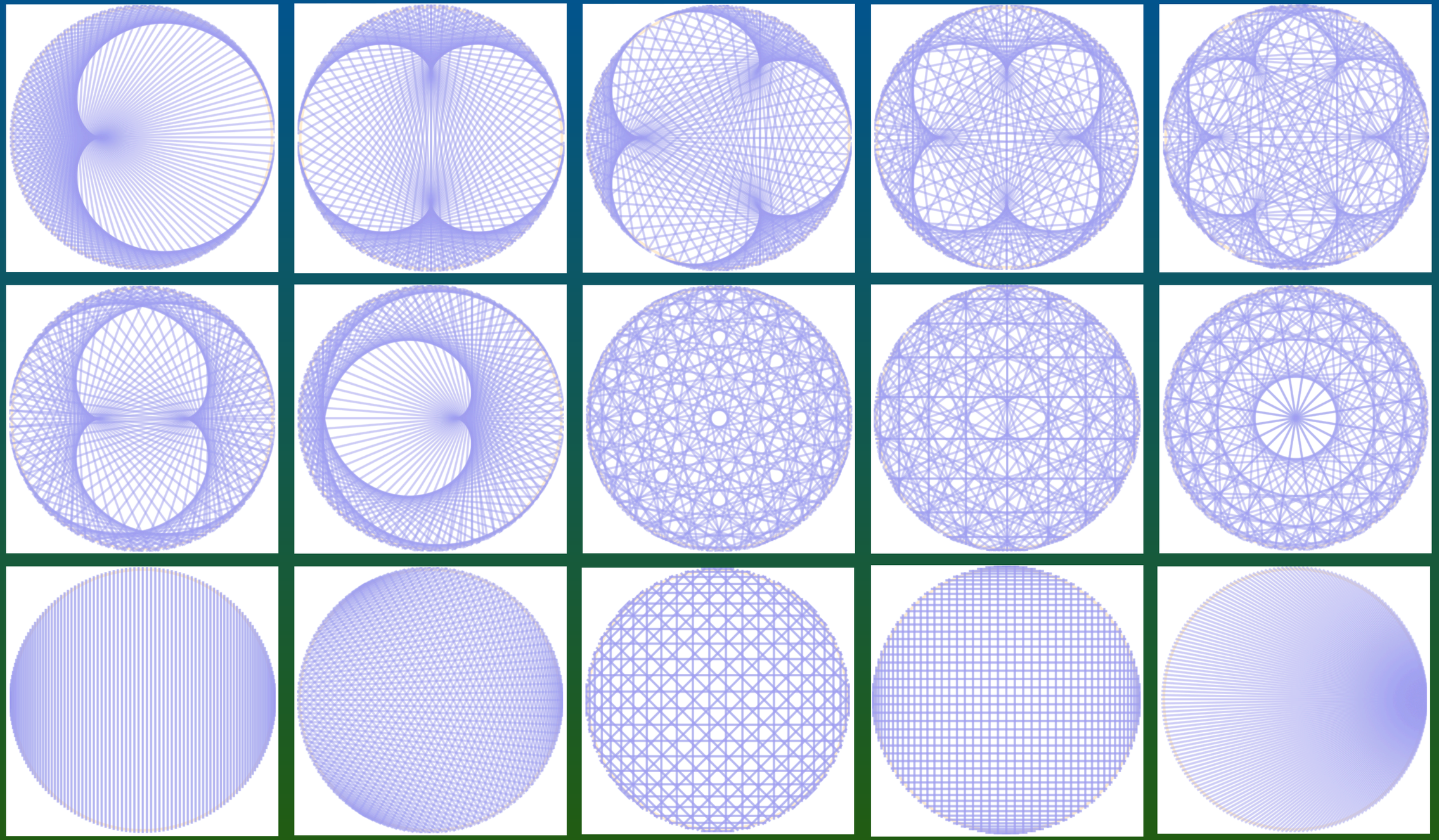
TypeScript	*OO with Generics*
PureScript	*Functional*
Elm	*Functional*

Approach

TypeScript	*Language*
PureScript	*Language + Tools*
Elm	*Programming System*

Cool

TypeScript	*Sum (union) type,*
	String Literal type
PureScript	*Eff Monad, [GADT]*
Elm	*Time travel debug,*
	SemVer guarantee

More Transpilers

ClojureScript	*Clojure(Lisp)*
GHCJS	*Haskell*
Babel	*JS*
CoffeeScript	*JS++*
GrooScript	*Groovy*