REALKINETIC
YOUR POTENTIAL REALIZED

This will give you a good mapping of how the Elm concepts and terminology map into Purescript

```
-- | Equivalent to Purescript's `show`.
toString :: ∀ a. (Show a) => a -> String
toString = show

-- | Equivalent to Purescript's `<<<`.
compose :: ∀ a b c. (b -> c) -> (a -> b) -> (a -> c)
compose = (<<<)

-- | Equivalent to Purescript's `$`.
applyFn :: ∀ a b. (a -> b) -> a -> b
applyFn = ($)

-- | The Purescript equivalent is `id`.
identity :: ∀ a. a -> a
identity = id

-- | The Purescript equivalent is `const`.
always :: ∀ a b. a -> b -> a
always = const

-- | The Purescript equivalent is `Void`.
type Never = Void
```

This will give you a good mapping of how the Elm concepts and terminology map into Purescript