JavaScript to PureScript

A Migration Story Starring Angular, Redux-Sagas, and Halogen

Jonathan Curran

Agenda

- Challenges in Scaling a JavaScript Web Application
- A Plan for Evaluating New Technology
- PureScript, Halogen
- Testing
- Introducing PureScript at work

Challenges

JavaScript

✅ Lots of existing libraries and frameworks

JavaScript

✅ Lots of existing libraries and frameworks

☒ Lack of common abstractions across stdlib, libraries, and frameworks

☒ Lack of types -> takes a lot of time to understand flow of data

Angular 1.x — *Superheroic JavaScript MVW Framework*

✅ Includes many useful things specific to web applications

Angular 1.x — *Superheroic JavaScript MVW Framework*

✅ Includes many useful things specific to web applications

☒ *Superheroic* = massive and complected

- components
- services
- resources
- scopes / digest cycle
- router
- validation
- directive
- templating language
- mocking

☒ Have to write code *The Angular Way*

Redux

✅ One location for application state

✅ Conceptually simple to understand

```
redux :: State -> (Action, Data) -> State
```

✅ Encourages separation of Data and Behavior

Redux

✅ One location for application state

✅ Conceptually simple to understand

```
redux :: State -> (Action, Data) -> State
```

✅ Encourages separation of Data and Behavior

☒ Boilerplate

☒ Synchronous

Redux Thunk

✅ It works

Redux Thunk

✅ It works

☒ Callback oriented

☒ Testing is very painful

Redux Sagas

✅ Easy to understand (`async/await`)

Redux Sagas

✅ Easy to understand (`async/await`)

☒ Terrible error handling

☒ Hard to follow code between 2 sets of reducers (redux, sagas)

☒ Testing

```
function* _sagaSearchForAllUsers(sagaServices) {
  const { users } = sagaServices;
  const searchQuery = select(allUsersSearchQuery);
  const filter = yield select(buildFilters);
  ...
}
```

```
uncaught at sagas
 at sagas
 at takeLatest
 at _sagaSearchForAllUsers
 $update_total_pages@http://localhost:3939/connect/out/dashboard.entry.js:265826:9
$items_changed@http://localhost:3939/connect/out/dashboard.entry.js:265863:9
set_items@http://localhost:3939/connect/out/dashboard.entry.js:265751:9
_sagaSearchForAllUsers$@http://localhost:3939/connect/out/dashboard.entry.js:259549:11
tryCatch@http://localhost:3939/connect/out/dashboard.entry.js:166604:37
invoke@http://localhost:3939/connect/out/dashboard.entry.js:166838:22
defineIteratorMethods/</prototype[method]@http://localhost:3939/connect/out/dashboard.entry.js:166656:16
next@http://localhost:3939/connect/out/dashboard.entry.js:80117:18
currCb@http://localhost:3939/connect/out/dashboard.entry.js:80193:7
runPutEffect/<@http://localhost:3939/connect/out/dashboard.entry.js:80304:16
exec@http://localhost:3939/connect/out/dashboard.entry.js:78972:5
flush@http://localhost:3939/connect/out/dashboard.entry.js:79013:5
asap@http://localhost:3939/connect/out/dashboard.entry.js:78986:5
runPutEffect@http://localhost:3939/connect/out/dashboard.entry.js:80291:5
runEffect@http://localhost:3939/connect/out/dashboard.entry.js:80240:307
next@http://localhost:3939/connect/out/dashboard.entry.js:80121:9
currCb@http://localhost:3939/connect/out/dashboard.entry.js:80193:7
resolvePromise/<@http://localhost:3939/connect/out/dashboard.entry.js:80256:14
processQueue@http://localhost:3939/connect/out/dashboard.entry.js:141457:28
scheduleProcessQueue/<@http://localhost:3939/connect/out/dashboard.entry.js:141473:27
$eval@http://localhost:3939/connect/out/dashboard.entry.js:142755:16
$digest@http://localhost:3939/connect/out/dashboard.entry.js:142569:15
$apply@http://localhost:3939/connect/out/dashboard.entry.js:142863:13
done@http://localhost:3939/connect/out/dashboard.entry.js:136843:36
completeRequest@http://localhost:3939/connect/out/dashboard.entry.js:137052:7
requestLoaded@http://localhost:3939/connect/out/dashboard.entry.js:136980:9
```

On Dynamic Languages

> "... how can you write a real program where you're just assigning random shit to other shit and expecting it to work?"
>
> — John Carmack [1]

[1] https://www.youtube.com/watch?v=00Q9-ftiPVQ&t=15m39s


I'll
do my
best!

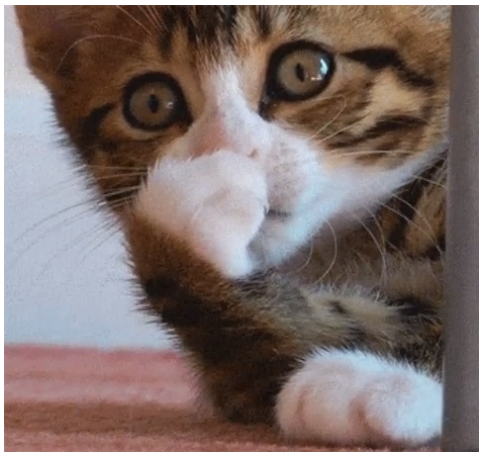
Ensuring it works

24,062 lines of application code

Ensuring it works

24,062 lines of application code

33,105 lines of test code



tl;dr

JavaScript is very difficult to scale

Evaluating New Technology

Criteria

Non-trivial

Criteria

Non-trivial

✅ Expressive Data Types

Criteria

Non-trivial

✅ Expressive Data Types

✅ Data validation (client-side)

Criteria

Non-trivial

✅ Expressive Data Types

✅ Data validation (client-side)

✅ XHR/AJAX

Criteria

Non-trivial

✅ Expressive Data Types

✅ Data validation (client-side)

✅ XHR/AJAX

✅ Child component(s)

Criteria

Non-trivial

✅ Expressive Data Types

✅ Data validation (client-side)

✅ XHR/AJAX

✅ Child component(s)

✅ Global/Mutable data

Criteria

Non-trivial

✅ Expressive Data Types

✅ Data validation (client-side)

✅ XHR/AJAX

✅ Child component(s)

✅ Global/Mutable data

✅ Testing

Users | Groups

Matching users

+ Add User

User	Role	Last Active
M myuser (unconfirmed)	viewer	*never logged in*
PP puser2 puser2 (puser2)	publisher	*never logged in*
PP puser3 puser3 (puser3)	publisher	*never logged in*
VV vuser1 vuser1 (vuser1)	viewer	*never logged in*
VV vuser2 vuser2 (vuser2)	viewer	*never logged in*
VV vuser3 vuser3 (vuser3)	viewer	*never logged in*

Options

Search

Search for a name...

Filter

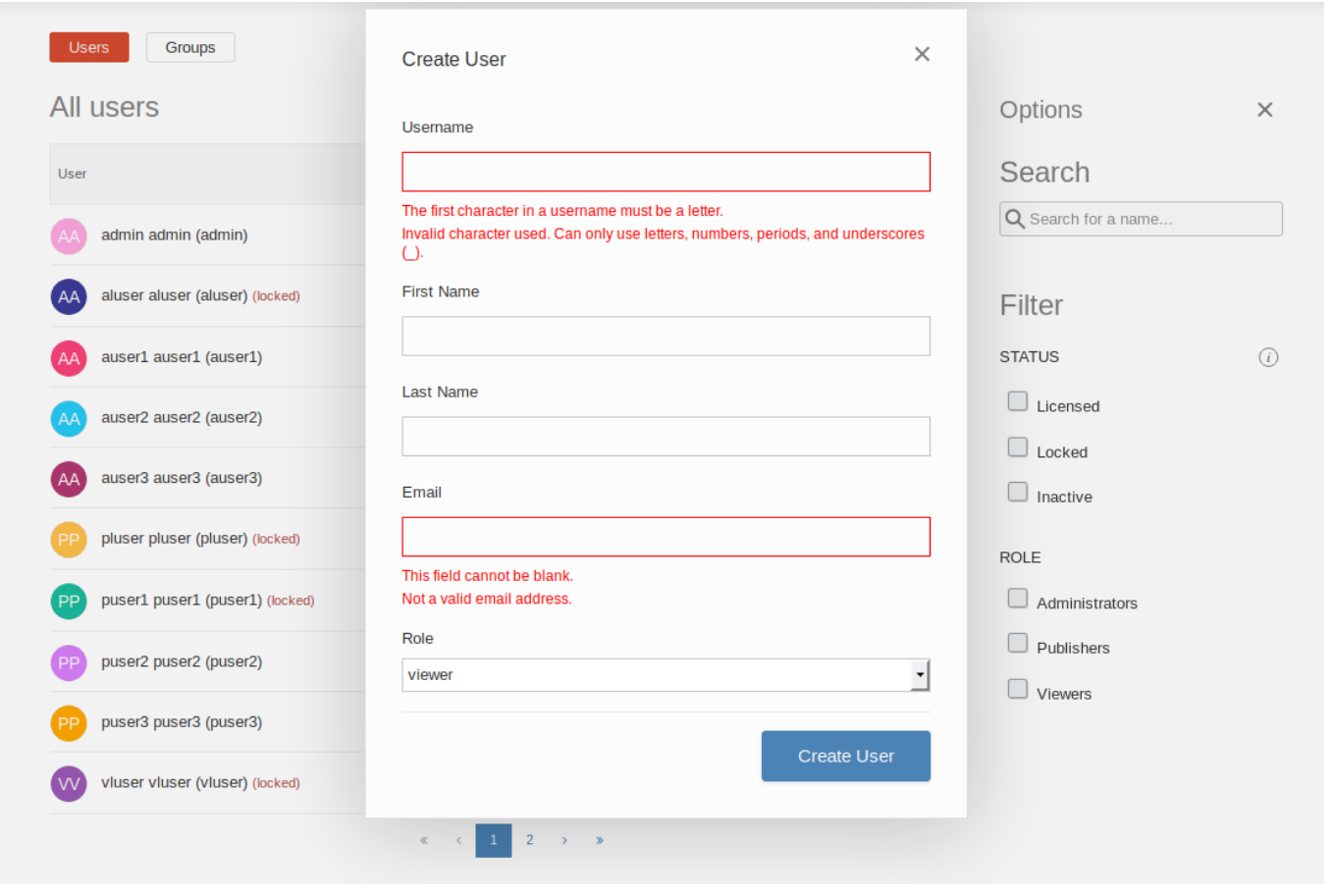
STATUS

Unlocked users who have been recently active count against your license. Locked users and inactive users do not count against user license limits.

- [ ] Licensed
- [ ] Locked
- [x] Inactive

ROLE

- [ ] Administrators
- [x] Publishers
- [x] Viewers


Users
Groups
All users
User
admin admin (admin)
aluser aluser (aluser) (locked)
auser1 auser1 (auser1)
auser2 auser2 (auser2)
auser3 auser3 (auser3)
pluser pluser (pluser) (locked)
puser1 puser1 (puser1) (locked)
puser2 puser2 (puser2)
puser3 puser3 (puser3)
vluser vluser (vluser) (locked)
Create User
Username
The first character in a username must be a letter.
Invalid character used. Can only use letters, numbers, periods, and underscores (_).
First Name
Last Name
Email
This field cannot be blank.
Not a valid email address.
Role
viewer
Create User
Options
Search
Search for a name...
Filter
STATUS
Licensed
Locked
Inactive
ROLE
Administrators
Publishers
Viewers
1
2

For security purposes, please re-enter your password

Log in

Log in with your credentials

Username

admin

Password

Forgot your password?

Log In

Don't have an account? Sign Up


PURESCRIPT

Expressive Data Types

```
type User
  = { id       :: Int
    , email    :: String
    , username :: String
    , fullName :: Maybe String
    , userRole :: String
    ...
    }
```

```
type User
  = { id       :: Int
    , email    :: String
    , username :: String
    , fullName :: Maybe String
    , userRole :: Role
    ...
    }

data Role
  = Admin
  | Publisher
  | Viewer
```

```
type User
  = { id       :: Int
    , email    :: Email
    , username :: Username
    , fullName :: Maybe String
    , userRole :: Role
    ...
    }

data Role
  = Admin
  | Publisher
  | Viewer

type Email = String
type Username = String
```

```
bob :: Username
bob = "bob"

exclaim :: String -> String
exclaim s = s <> "!!"

main =
  log (exclaim bob)
```

```
newtype User
  = User
  { id       :: Int
  , email    :: Email
  , username :: Username
  , fullName :: Maybe String
  , userRole :: Role
  ...
  }

data Role
  = Admin
  | Publisher
  | Viewer

newtype Email = Email String
newtype Username = Username String
```

```haskell
newtype User
  = User
  { id       :: Int
  , email    :: Email
  , username :: Username
  , fullName :: Maybe String
  , userRole :: Role
  ...
  }

data Role
  = Admin
  | Publisher
  | Viewer

newtype Email = Email String
newtype Username = Username String
```

```haskell
-- before
bob :: Username
bob = "bob"

-- after
bob :: Username
bob = Username "bob"
```

Deriving Functionality

```
-- e.g. Admin == Viewer

-- by-hand
instance eqRole :: Eq Role where
  eq Admin Admin         = true
  eq Publisher Publisher = true
  eq Viewer Viewer       = true
  eq _ _                 = false

-- compiler derived
derive instance eqRole :: Eq Role
```

Deriving Functionality

```
-- e.g. Admin == Viewer

-- by-hand
instance eqRole :: Eq Role where
  eq Admin Admin         = true
  eq Publisher Publisher = true
  eq Viewer Viewer       = true
  eq _ _                 = false

-- compiler derived
derive instance eqRole :: Eq Role
```

```
-- e.g. Admin > Viewer
derive instance ordRole :: Ord Role
```

Deriving Functionality

```
-- e.g. Admin == Viewer

-- by-hand
instance eqRole :: Eq Role where
  eq Admin Admin         = true
  eq Publisher Publisher = true
  eq Viewer Viewer       = true
  eq _ _                 = false

-- compiler derived
derive instance eqRole :: Eq Role
```

```
-- e.g. Admin > Viewer
derive instance ordRole :: Ord Role
```

```
derive instance genericRole :: Generic Role _

instance showRole :: Show Role where
  show = genericShow
```

Data Validation

```purescript
data ValidationError
  = IsEmpty
  | TooShort Int
  | TooLong Int
  | FirstCharNotLetter
  | InvalidUsername
  | BlacklistedUsername
  | InvalidEmail
  | InvalidRole

derive instance eqValidationError :: Eq ValidationError

errorMessage :: ValidationError -> String
errorMessage = case _ of
  IsEmpty            -> "This field cannot be blank."
  TooShort n         -> "Must be at least " <> show n <> " characters."
  TooLong n          -> "Must be at most " <> show n <> " characters."
  ...
```

purescript-validation

```
newtype V err result

-- interesting instances of V
  Functor (V err)
  Bifunctor V
  (Semigroup err) => Applicative (V err)

-- creates an instance of V given an err
invalid :: forall err result. err -> V err result

-- transforms V into another type
unV :: forall err result r. (err -> r) -> (result -> r) -> V err result -> r
```

purescript-validation

```
newtype V err result

-- interesting instances of V
  Functor (V err)
  Bifunctor V
  (Semigroup err) => Applicative (V err)

-- creates an instance of V given an err
invalid :: forall err result. err -> V err result

-- transforms V into another type
unV :: forall err result r. (err -> r) -> (result -> r) -> V err result -> r
```

```
type Validation result = V (NonEmptyList ValidationError) result
```

purescript-validation

```
newtype V err result

-- interesting instances of V
  Functor (V err)
  Bifunctor V
  (Semigroup err) => Applicative (V err)

-- creates an instance of V given an err
invalid :: forall err result. err -> V err result

-- transforms V into another type
unV :: forall err result r. (err -> r) -> (result -> r) -> V err result -> r
```

```
type Validation result = V (NonEmptyList ValidationError) result
```

```
isEmpty :: String -> Validation String
isEmpty input
  | String.null (String.trim input) = invalid (pure IsEmpty)
  | otherwise = pure input
```

purescript-validation

```
newtype V err result

-- interesting instances of V
  Functor (V err)
  Bifunctor V
  (Semigroup err) => Applicative (V err)

-- creates an instance of V given an err
invalid :: forall err result. err -> V err result

-- transforms V into another type
unV :: forall err result r. (err -> r) -> (result -> r) -> V err result -> r
```

```
type Validation result = V (NonEmptyList ValidationError) result
```

```
isEmpty :: String -> Validation String
isEmpty input
  | String.null (String.trim input) = invalid (pure IsEmpty)
  | otherwise = pure input
```

```
validEmail :: String -> Validation Email
validEmail input =
  isEmpty input *> invalidEmail
  where
    invalidEmail
      | not (Regex.test emailRegex input) = invalid (pure InvalidEmail)
      | otherwise = pure (Email input)
```

```haskell
validatedUser :: Either (NonEmpty ValidationError) CreateUserRequest
validatedUser =
  { username: _, firstName: _, lastName: _, email: _, role: _ }
  <$> validUsername usernameValidator usernameBlacklist username
  <*> validFirstName firstName
  <*> validLastName lastName
  <*> validEmail email
  <*> validRole role
  # unV Left (CreateUserRequest >>> Right)
```

XHR/AJAX

Handling JSON

simple-json

JSON String to Type

```
newtype AuthenticationSettings = AuthenticationSettings { ... }

decodeAuthSettings :: Either (NonEmptyList ForeignError) AuthenticationSettings
decodeAuthSettings =
  map AuthenticationSettings
  ( readJSON
    """{"handlesLogin":true, "externalUserData":false
       ,"externalGroupData":false, "groupsEnabled":true
       ,"supportsChallenge":false
       }
    """
  )
```

simple-json

JSON String to Type

```
newtype AuthenticationSettings = AuthenticationSettings { ... }

decodeAuthSettings :: Either (NonEmptyList ForeignError) AuthenticationSettings
decodeAuthSettings =
  map AuthenticationSettings
  ( readJSON
    """{"handlesLogin":true, "externalUserData":false
       ,"externalGroupData":false, "groupsEnabled":true
       ,"supportsChallenge":false
       }
    """
  )
```

Type to JSON String

```
encodeAuthSettings :: String
encodeAuthSettings =
  writeJSON {"handlesLogin":true, "externalUserData":false
            ,"externalGroupData":false, "groupsEnabled":true
            ,"supportsChallenge":false
            }
```

> purescript-foreign-generic

JSON String to Type

```
newtype AuthenticationSettings = AuthenticationSettings { ... }
derive instance gAuthenticationSettings :: Generic AuthenticationSettings _

decodeAuthSettings :: Either (NonEmptyList ForeignError) AuthenticationSettings
decodeAuthSettings =
  runExcept $
    genericDecodeJSON defaultOptions
    """{"tag": "AuthenticationSettings"
       ,"contents": {"handlesLogin":true,"externalUserData":false
                    ,"externalGroupData":false,"groupsEnabled":true
                    ,"supportsChallenge":false
                    }
       }
    """
```

> purescript-foreign-generic

JSON String to Type

```
newtype AuthenticationSettings = AuthenticationSettings { ... }
derive instance gAuthenticationSettings :: Generic AuthenticationSettings _

decodeAuthSettings :: Either (NonEmptyList ForeignError) AuthenticationSettings
decodeAuthSettings =
  runExcept $
    genericDecodeJSON defaultOptions
    """{"tag": "AuthenticationSettings"
       ,"contents": {"handlesLogin":true,"externalUserData":false
                    ,"externalGroupData":false,"groupsEnabled":true
                    ,"supportsChallenge":false
                    }
       }
    """
```

Type to JSON String

```
encodeAuthSettings :: String
encodeAuthSettings =
  genericEncodeJSON defaultOptions
    {"handlesLogin":true, "externalUserData":false
    ,"externalGroupData":false, "groupsEnabled":true
    ,"supportsChallenge":false
    }
```

purescript-argonaut

```
decodeAuthenticationSettings :: Json -> Either String AuthenticationSettings
decodeAuthenticationSettings json = do
  j                   <- decodeJson json
  handlesLogin        <- j .? "handles_login"
  externalUserData    <- j .? "external_user_data"
  externalGroupData   <- j .? "external_group_data"
  groupsEnabled       <- j .? "groups_enabled"
  supportsChallenge   <- j .? "challenge_response_enabled"
  name                <- j .? "name"
  pure $
    AuthenticationSettings
    { handlesLogin
    , externalUserData
    , externalGroupData
    , groupsEnabled
    , supportsChallenge
    , name
    }
```

purescript-argonaut

```
encodeCreateUserRequest :: CreateUserRequest -> Json
encodeCreateUserRequest (CreateUserRequest r) =
     "username"   := r.username
  ~> "email"      := r.email
  ~> "first_name" := r.firstName
  ~> "last_name"  := r.lastName
  ~> "role"       := r.role
```

purescript-affjax

```
data XhrError
  = ParseError String
  | DataError {code :: Int, error :: String}
  | Unauthorized
  | ReAuthenticate
  | ServerError
```

purescript-affjax

```
data XhrError
  = ParseError String
  | DataError {code :: Int, error :: String}
  | Unauthorized
  | ReAuthenticate
  | ServerError
```

```
decodeWithError
  :: forall a. Ajax.AffjaxResponse String
  -> (Json -> Either String a)
  -> Either XhrError a
decodeWithError response jsonDecoder = ...
```

purescript-affjax

```
data XhrError
  = ParseError String
  | DataError {code :: Int, error :: String}
  | Unauthorized
  | ReAuthenticate
  | ServerError
```

```
decodeWithError
  :: forall a. Ajax.AffjaxResponse String
  -> (Json -> Either String a)
  -> Either XhrError a
decodeWithError response jsonDecoder = ...
```

```
getSettings
  :: forall e. AVar XsrfToken
  -> Xhr e (Either XhrError Settings)
getSettings tokenAVar = do
  xsrfToken <- AVar.readVar tokenAVar
  res <- Ajax.affjax $
         Ajax.defaultRequest
         { url = "/api/settings"
         , headers = [RequestHeader "XSRF-Token" xsrfToken]
         }
  pure $ decodeWithError decodeSettings res
```

purescript-affjax

```
data XhrError
  = ParseError String
  | DataError {code :: Int, error :: String}
  | Unauthorized
  | ReAuthenticate
  | ServerError
```

```
decodeWithError
  :: forall a. Ajax.AffjaxResponse String
  -> (Json -> Either String a)
  -> Either XhrError a
decodeWithError response jsonDecoder = ...
```

```
getSettings
  :: forall e. AVar XsrfToken
  -> Xhr e (Either XhrError Settings)
getSettings tokenAVar = do
  xsrfToken <- AVar.readVar tokenAVar
  res <- Ajax.affjax $
         Ajax.defaultRequest
         { url = "/api/settings"
         , headers = [RequestHeader "XSRF-Token" xsrfToken]
         }
  pure $ decodeWithError decodeSettings res
```

```
type Xhr e = Aff (ajax :: AJAX, avar :: AVAR | e)
```

Halogen

Halogen Primer

> A declarative, type-safe UI library for PureScript

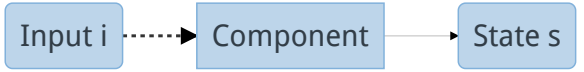
Components

- have local state
- nested
- communicate bi-directionally
    - child can send data to parent
    - parent can request data from child

Halogen Architecture

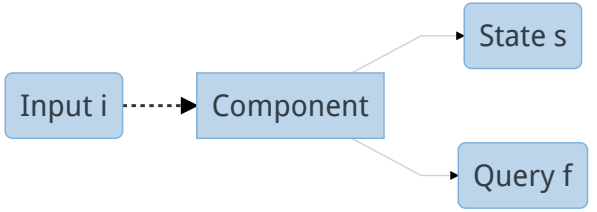

```
type Input =
  { xsrfToken         :: AVar XsrfToken
  , username          :: Username
  , usernameValidator :: UsernameValidator
  , usernameBlacklist :: Array Username
  }
```

Halogen Architecture



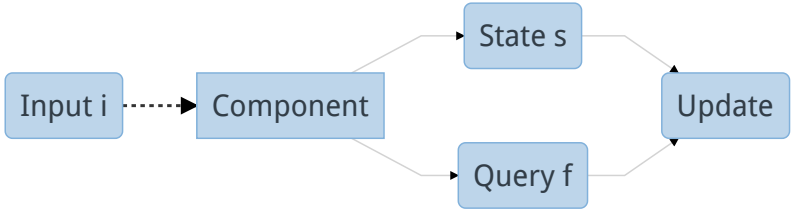

```
type State =
  { xsrfToken         :: AVar XsrfToken
  , usernameValidator :: UsernameValidator
  , usernameBlacklist :: Array Username
  , formData          :: FormData
  , formErrors        :: FormErrors
  , generalError      :: Maybe String
  }
```

Halogen Architecture



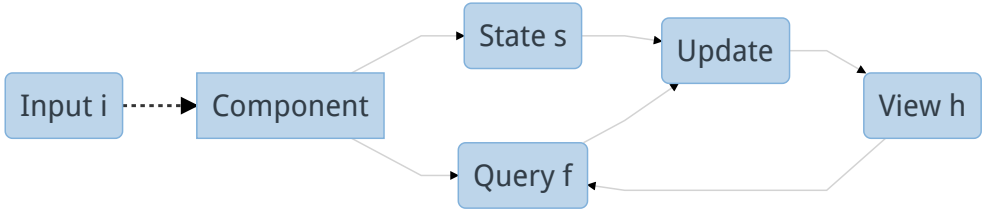

```
data Query a
  = PreventDefault Event (Query a)
  | UsernameChanged String a
  | PasswordChanged String a
  | SubmitForm a
```

Halogen Architecture



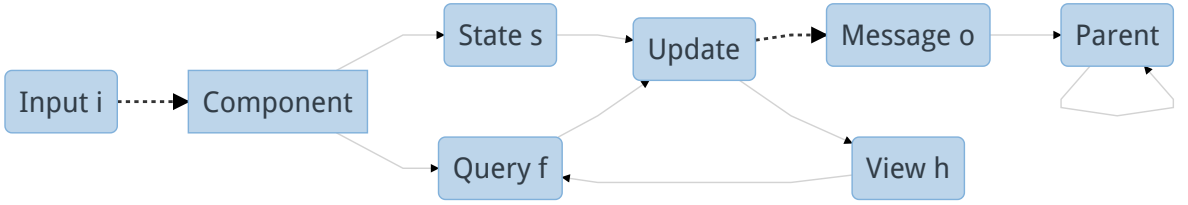

```
update = case _ of
  PreventDefault event query -> do
    -- call DOM preventDefault()
    update query

  UsernameChanged newUsername next -> pure next <* do
    H.modify (_ { formData { username = newUsername } } )

  PasswordChanged newPassword next -> pure next <* do
    H.modify (_ { formData { password = newPassword } } )

  SubmitForm next -> pure next <* do
    -- perform field validation and update state
    -- if valid perform AJAX and update state
```

Halogen Architecture



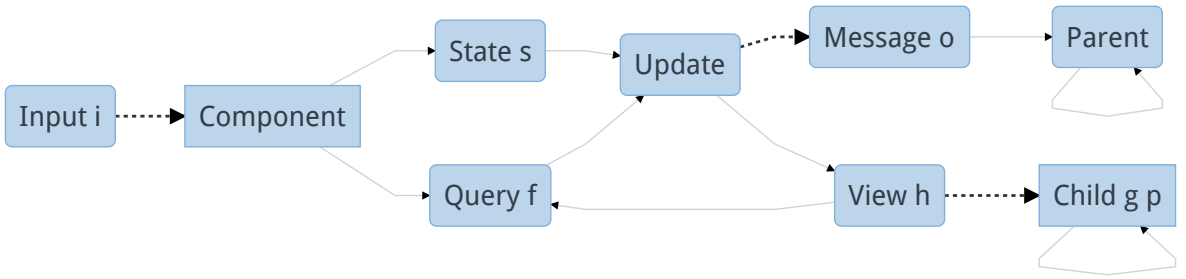

```
view state =
  HH.form [ class_ "loginDialog"
          , HE.onSubmit (toEvent >>> prevDefault SubmitForm)
          ]
  [ HH.input [ HP.type_ HP.InputText
             , HP.value state.formData.username
             , HE.onValueChange (HE.input UsernameChanged)
             ]

  ,
  -- snip --
  , HH.button_ [ HH.text "Log In" ]
  ]
```

Halogen Architecture




```
data Message
  = LoggedIn

update = case _ of
  SubmitForm next -> pure next <* do
    -- if successful:
    H.raise LoggedIn

  ...
```

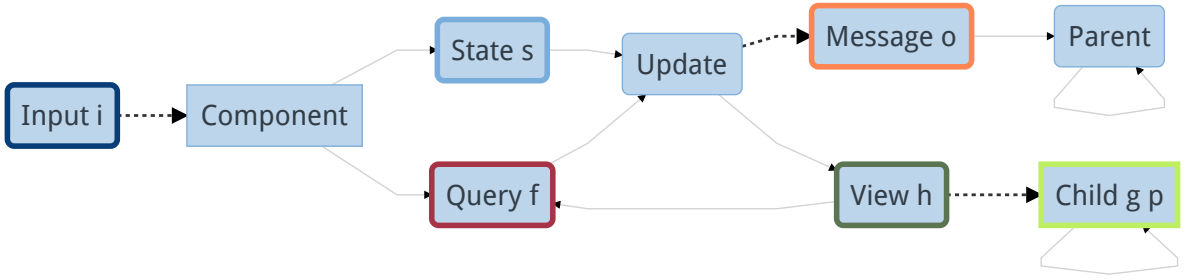
Halogen Architecture

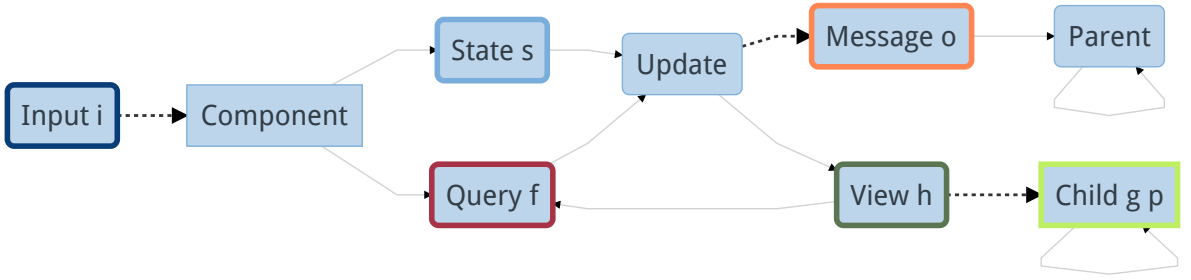


```
-- in parent component

data Query a
  = ...
  | HandleReAuth Login.Message a

update query = case query of
  ...

  HandleReAuth Login.LoggedIn next -> pure next <* do
    H.modify (_ { showLogin = false } )
    -- retry form-submission again (AJAX)
```

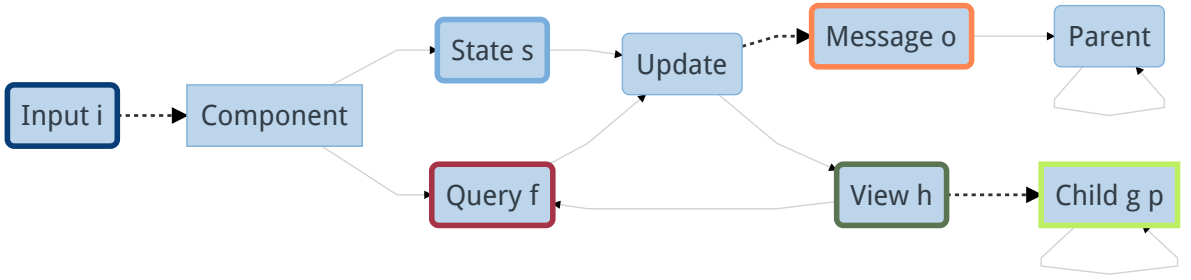
```
-- in parent component

data Query a
  = ...
  | HandleReAuth Login.Message a

update query = case query of
  ...

  HandleReAuth Login.LoggedIn next -> pure next <* do
    H.modify (_ { showLogin = false } )
    -- retry form-submission again (AJAX)
```

```
data Slot = ReAuthSlot
derive instance eqSlot :: Eq Slot
derive instance ordSlot :: Ord Slot

view state =
  if state.showLogin
  then
    -- display login form
    HH.div_ [HH.slot ReAuthSlot Login.component loginInput (HE.input HandleReAuth)]
  else
    -- display user creation form
    renderDialog state
```


Input i
Component
State s
Query f
Update
Message o
Parent
View h
Child g p

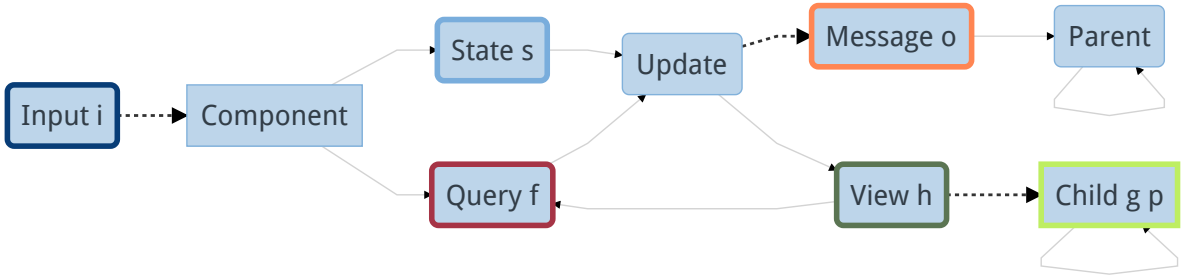



```
type Component h s f g p i o m = { ... }
```



type Component h s f g p i o m = { ... }

```
-- m
type UIEffect eff =
  Aff
  ( ajax :: AJAX
  , dom  :: DOM
  , avar :: AVAR
  | eff
  )
```




```
type Component h s f g p i o m = { ... }
```

```
component :: forall eff. Component HTML Query Input Message (UIEffect eff)
component =
  H.parentComponent
  { initialState: (setupInitialState :: Input -> State)
  , render: view
  , eval: update
  -- receiver :: Input -> Maybe (Query Unit)
  , receiver: const Nothing
  }
```

Embedding Halogen in Angular

```
init = do
  HA.runHalogenAff do
    mEl <- HA.selectElement (QuerySelector "#all-users-ps")
    case mEl of
      Nothing ->
        log "Failed to find #all-users-ps"
      Just element -> do
        token <- tokenFromCookie "cookie-name"
        xsrfToken <- Aff.makeVar (XsrfToken token)
        void $ runUI container xsrfToken element
```

Embedding Halogen in Angular

```
init = do
  HA.runHalogenAff do
    mEl <- HA.selectElement (QuerySelector "#all-users-ps")
    case mEl of
      Nothing ->
        log "Failed to find #all-users-ps"
      Just element -> do
        token <- tokenFromCookie "cookie-name"
        xsrfToken <- Aff.makeVar (XsrfToken token)
        void $ runUI container xsrfToken element
```

```
import * as PS from '../purescript/output/Main';

export class AllUsersPS {
  $onInit() {
    PS.init();
  }
}

AllUsersPS.$inject = [];

const allUsersPS = {
  controller: AllUsersPS,
  template: '<div id="all-users-ps"></div>',
};

export default angular.module('allUsersPS', [])
                      .component('allUsersPS', allUsersPS).name;
```

Halogen ↔ Other PureScript Code

```
init = do
  ...
  io <- runUI container xsrfToken element
  ...


io :: { query     :: Query ~> m
      , subscribe :: Consumer Message m Unit -> m Unit }
```

PureScript ↔JS

```
-- Util.purs

foreign import cookieValueImpl
  :: String                    --| cookie name
  -> (String -> Maybe String) --| Just constructor
  -> Maybe String              --| default value (Nothing)
  -> Maybe String              --| possible cookie value
```

PureScript ↔JS

```
-- Util.purs

foreign import cookieValueImpl
  :: String                    --| cookie name
  -> (String -> Maybe String) --| Just constructor
  -> Maybe String              --| default value (Nothing)
  -> Maybe String              --| possible cookie value
```

```
// Util.js
'use strict';

exports.cookieValueImpl = function(CookieName) {
  return function(Just) {
    return function(Nothing) {
      // ...
      return foundCookie ? Just(value) : Nothing
    }
  }
}
```

PureScript ↔JS

```
-- Util.purs

foreign import cookieValueImpl
  :: String                    --| cookie name
  -> (String -> Maybe String) --| Just constructor
  -> Maybe String              --| default value (Nothing)
  -> Maybe String              --| possible cookie value
```

```
// Util.js
'use strict';

exports.cookieValueImpl = function(CookieName) {
  return function(Just) {
    return function(Nothing) {
      // ...
      return foundCookie ? Just(value) : Nothing
    }
  }
}
```

```
-- lol it's not pure but w/e
cookieValue :: String -> Maybe String
cookieValue cookieName = cookieValueImpl cookieName Just Nothing
```

Testing

> purescript-test-unit

```
validationSuite =
  suite "decoding" do
    test "decodeBool" do
      for_ ["T", "t", "1", "TRuE", "true", "yes"] $
        \v -> decodeBool v `shouldEqual` (Right true)

    test "decodeRole" do
      for_ [ Tuple "administrator" Admin
           , Tuple "publisher" Publisher
           , Tuple "viewer" Viewer
           ] $ \(Tuple s expected) -> decodeRole s `shouldEqual` (Right expected)
```

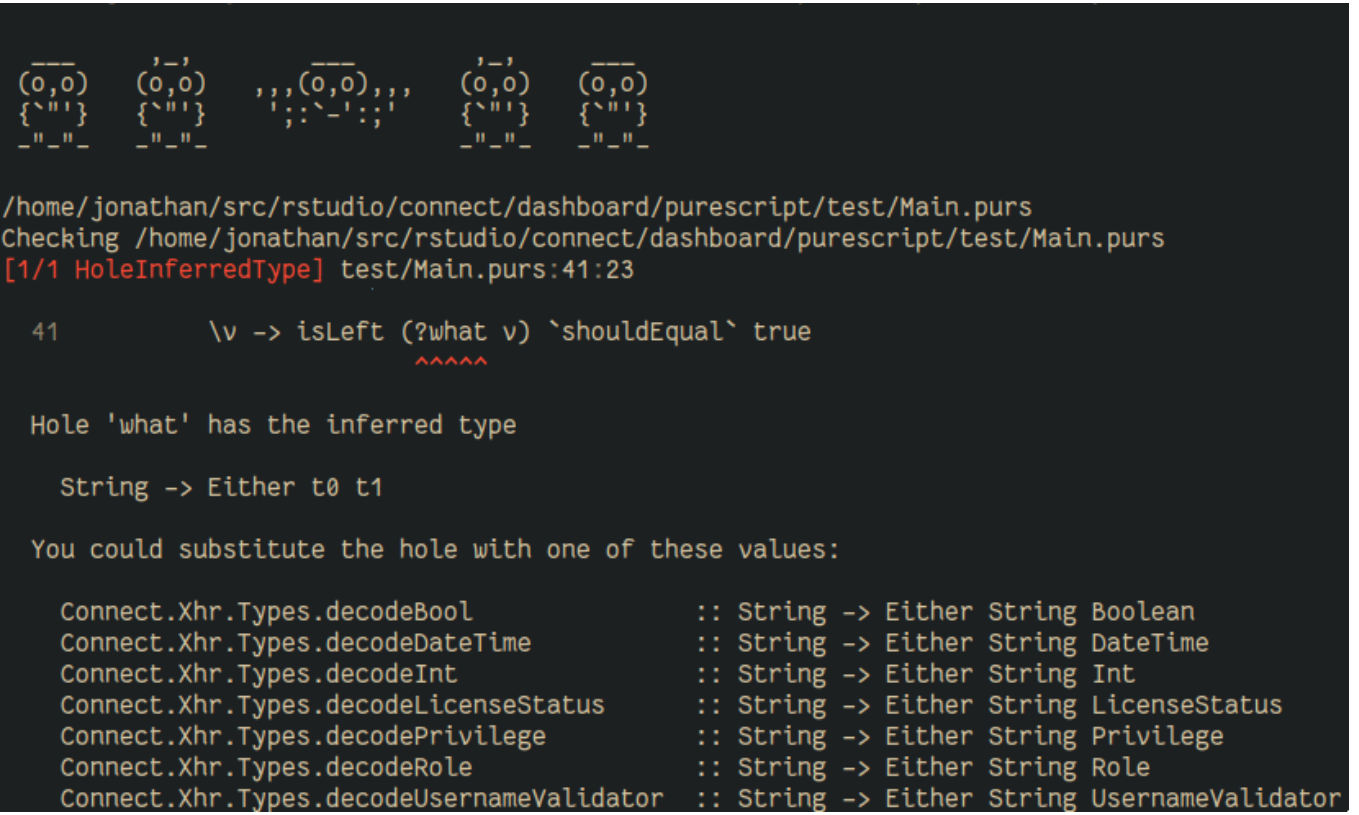
Testing

> purescript-quickcheck

```
test "decodeInt" do
  quickCheck $
    suchThat arbitrary (_ >= 0)
    >>= pure <<< \n -> decodeInt (show n) === Right n

test "cannot contain invalid characters" do
  quickCheck $
    suchThat arbitrary (not <<< Regex.test usernameValidCharsRe)
    >>= pure <<< (\s ->
                  let result = vusername Default s in
                  Array.elem InvalidUsername result === true)
```

Tooling

- pursuit
- psc-package
- pscid
- editor integration
- type holes


/home/jonathan/src/rstudio/connect/dashboard/purescript/test/Main.purs
Checking /home/jonathan/src/rstudio/connect/dashboard/purescript/test/Main.purs
[1/1 HoleInferredType] test/Main.purs:41:23
41 \v -> isLeft (?what v) `shouldEqual` true
^^^^^
Hole 'what' has the inferred type
String -> Either t0 t1
You could substitute the hole with one of these values:
Connect.Xhr.Types.decodeBool :: String -> Either String Boolean
Connect.Xhr.Types.decodeDateTime :: String -> Either String DateTime
Connect.Xhr.Types.decodeInt :: String -> Either String Int
Connect.Xhr.Types.decodeLicenseStatus :: String -> Either String LicenseStatus
Connect.Xhr.Types.decodePrivilege :: String -> Either String Privilege
Connect.Xhr.Types.decodeRole :: String -> Either String Role
Connect.Xhr.Types.decodeUsernameValidator :: String -> Either String UsernameValidator

¯_(ツ)_/¯

- FFI involves writing basic ES5 JS
- Embedding > 2 child components

¯_(ツ)_/¯

- FFI involves writing basic ES5 JS
- Embedding > 2 child components

ಥ_ಥ

- ☹ Terms in PS+Halogen docs can be unfamiliar
- ☹ Docs in PS ecosystem can be lacking

Introducing PureScript at Work

☒ Go at it alone

Introducing PureScript at Work

☒ Go at it alone

✓ Plan to go slow

Introducing PureScript at Work

☒ Go at it alone

✓ Plan to go slow

✓ Join the PureScript community

>  + 
>
> http://fpchat-invite.herokuapp.com/
>
> `#purescript`
>
> `#purescript-beginners`

Thanks!

Reach out - joncfoo@curran.in

Books

- *Haskell Programming From First Principles*
- *PureScript by Example*

Example github projects

- `vladciobanu/purescript-affjax-errors`
- `slamdata/purescript-halogen` (docs, examples)
- `vladciobanu/purescript-halogen-example`
- `natefaubion/slamdata`

Thanks!

Reach out - joncfoo@curran.in

Books

- *Haskell Programming From First Principles*
- *PureScript by Example*

Example github projects

- `vladciobanu/purescript-affjax-errors`
- `slamdata/purescript-halogen` (docs, examples)
- `vladciobanu/purescript-halogen-example`
- `natefaubion/slamdata`

We're hiring - https://rstd.io/lambdaconf