Re-frame Reference Sheet https://purelyfunctional.tv/guide/re-frame-building-blocks/

Events name a user-generated or server-generated action and defines the intent.

```
(re-frame.core/dispatch [<:event-name> <arg1> <arg2> ...])   ← add an Event to the Queue

(rf/reg-event-fx
  <:event-name>                                              ← Event name used to look up handler
  [<interceptors> (rf/inject-cofx <:co-effect-name> <arg>)]   ← add Co-effects here; optional interceptor vector
  (fn [cofx [_ <arg1> <arg2> ...]]
    <effects-map>))                                          ← return a map of Effect names to Effect values

(rf/reg-event-db                                             ← special case for Events that only modify the database.
  <:event-name>
  [<interceptors> (rf/inject-cofx <:co-effect-name> <arg>)]   ← optional interceptor vector
  (fn [db [_ <arg1> <arg2> ...]]
    <modified database>))                                    ← return modified database
```

Effects change the world - the database, ajax requests, etc.

```
(rf/reg-fx
  <:effect-name>
  (fn [effect-value]                                         ← Effect value is the value in the Effects map.
    <do the effect>))                                        ← whatever action you need to take.
```

Co-effects are the data Event handlers need

```
(rf/reg-cofx
  <:co-effect-name>
  (fn [cofx arg]
    <modified cofx map>))                                    ← return a modified Co-effects map.
```

Subscriptions are current values calculated from the database.

```
(rf/subscribe [<:subscription-name> <arg1> <arg2> ...])      ← subscribe to the current value by name and args.

(rf/reg-sub
  <:subscription-name>
  (fn [db _]
    <value from database>))                                  ← extract value from the database.

(rf/reg-sub                                                  ← extract value from the database.
  <:subscription-name>
  (fn [[_ <arg1> <arg2> ...]]                                ← a function that returns a subscription
    (rf/subscribe [<:other-subscription>]))
  (fn [<x> [_ <arg1> <arg2> ...]]                            ← <x> is value from subscription
    <value calculated from x>))                              ← calculate a new value from it
```