Another cond-> example

```
(defn normalize [{:keys [age date name] :as m}]
  (cond-> m
          (string? age) (update :age edn/read-string)
          (string? date) (update :date clojure.instant/read-instant-date)
          (not (re-matches #"[A-Z][a-z]+" name)) (update :name
                                                         (comp
                                                           cs/capitalize
                                                           cs/lower-case))))
```

```
(normalize
  {:name "sam"
   :age "54"
   :date "2018-08-11"})
=> {:name "Sam", :age 54, :date #inst"2018-08-11T00:00:00.000-00:00"}
```

```
(normalize
  {:name "Bob"
   :age 23
   :date #inst"2012-01-02"})
=> {:name "Bob", :age 23, :date #inst"2012-01-02T00:00:00.000-00:00"}
```

as->, as->>

- Use when you want to bind a symbol through your pipeline
- Use when both -> and ->> are needed or are becoming awkward
- I don’t see this macro used a lot
- Often times an intermediate result in a let form or a separate function solves the same issue more cleanly

```
(as-> [:foo :bar] v
      (map name v)
      (first v)
      (.substring v 1))
```

Example from https://clojure.org/guides/threading_macros