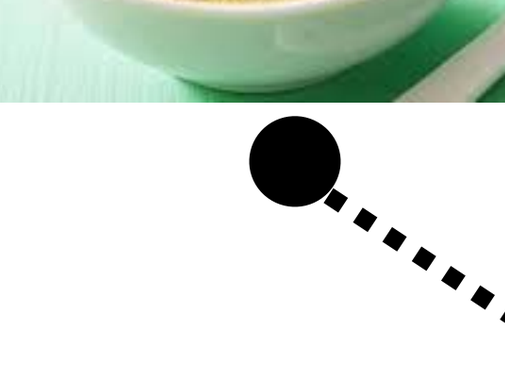
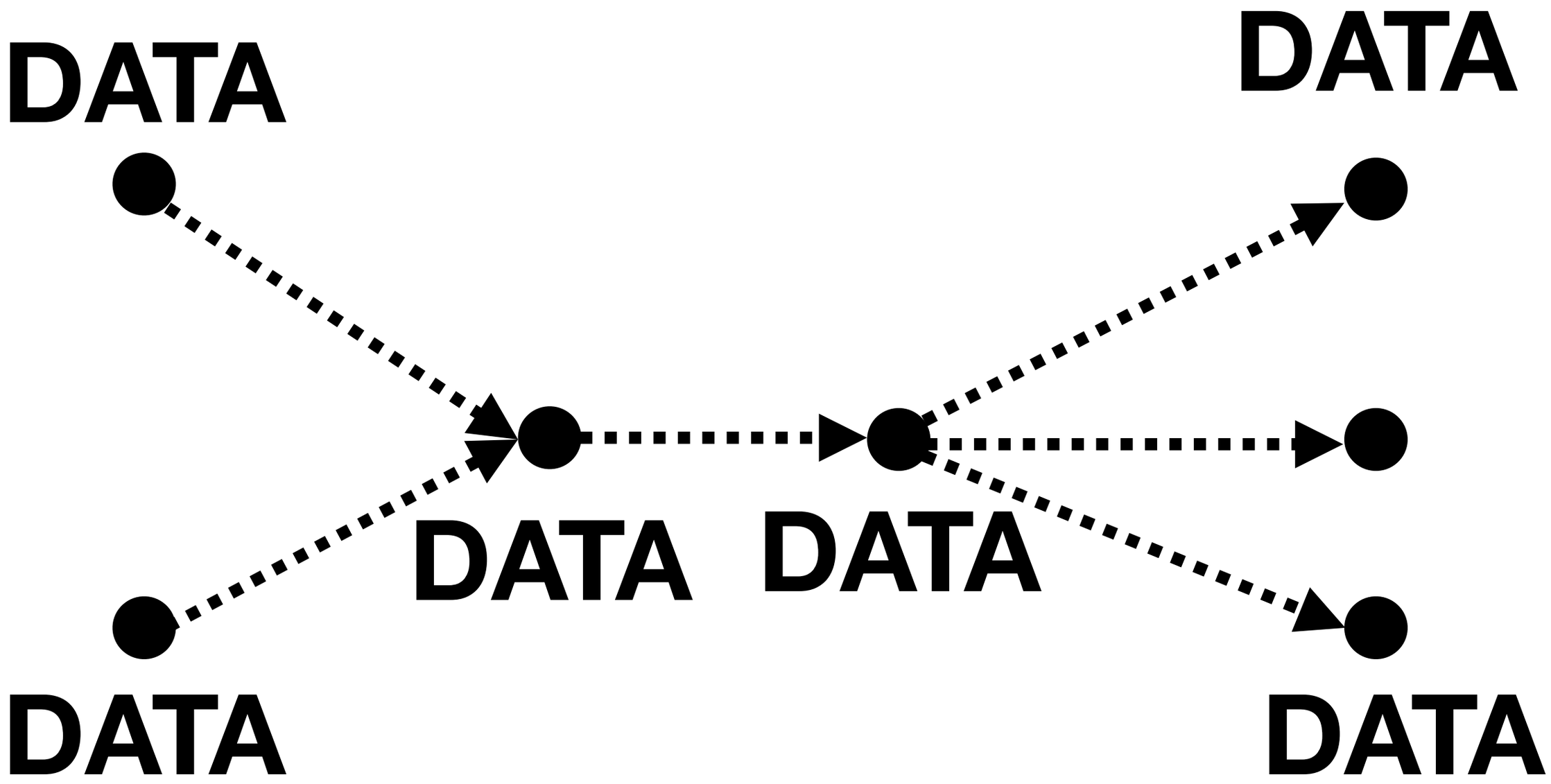
All things data

@DanLebrero
akvo.org

?




DATA
DATA
DATA
DATA
DATA
DATA
DATA


DATA
A
B

Plain old data

numbers → 10, 1.1

boolean → **true** / **false**

string → **“foo”**

enum/keyword → **:color**, **:red**

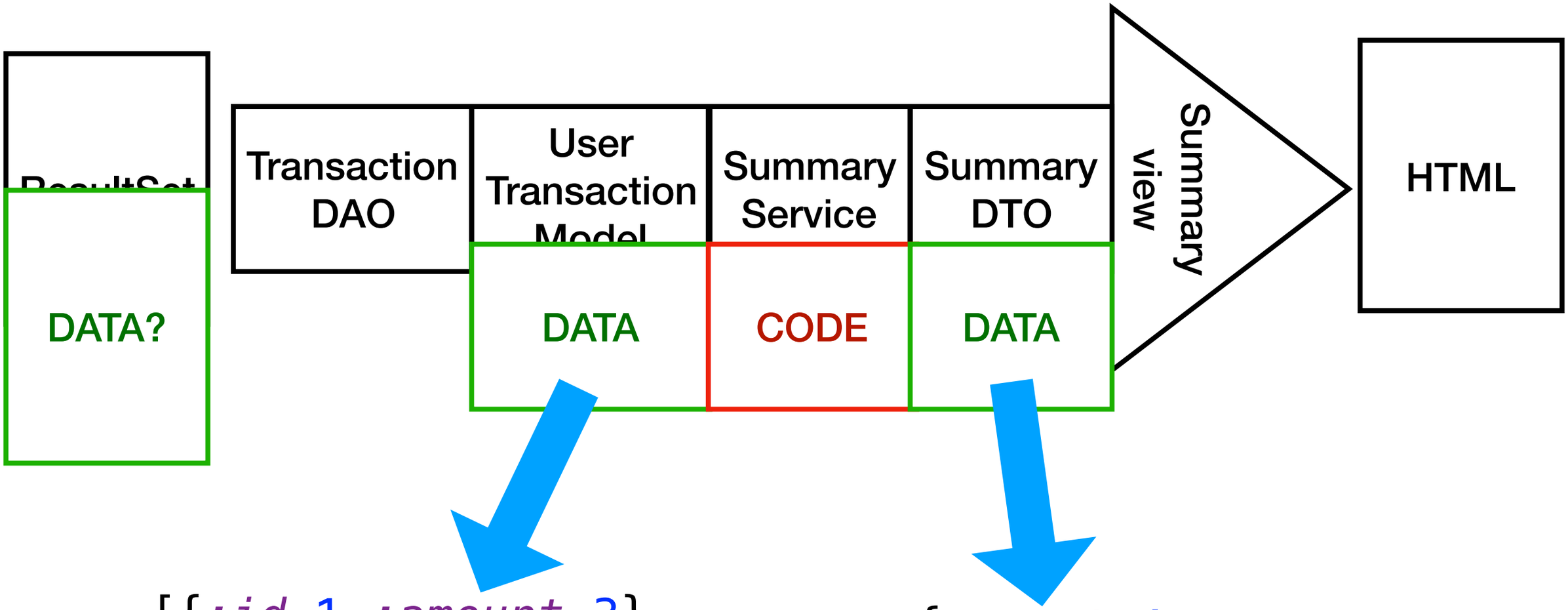
lists → (1 2 3) or [1 2 3]

map → {key1 val1, key2 val2}

set → #{e1, e2}

nothing → **null**

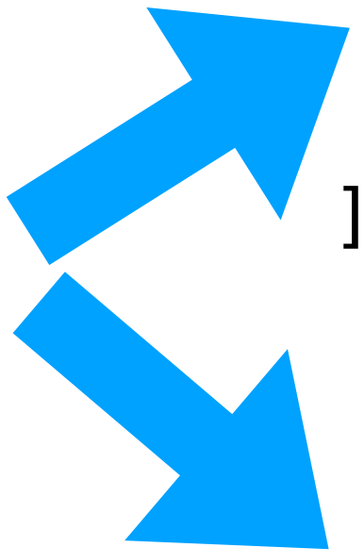
Example




```
[{:id 1 :amount 3}
 {:id 2 :amount 7}
 {:id 3 :amount 4}
 {:id 4 :amount 6}]
```

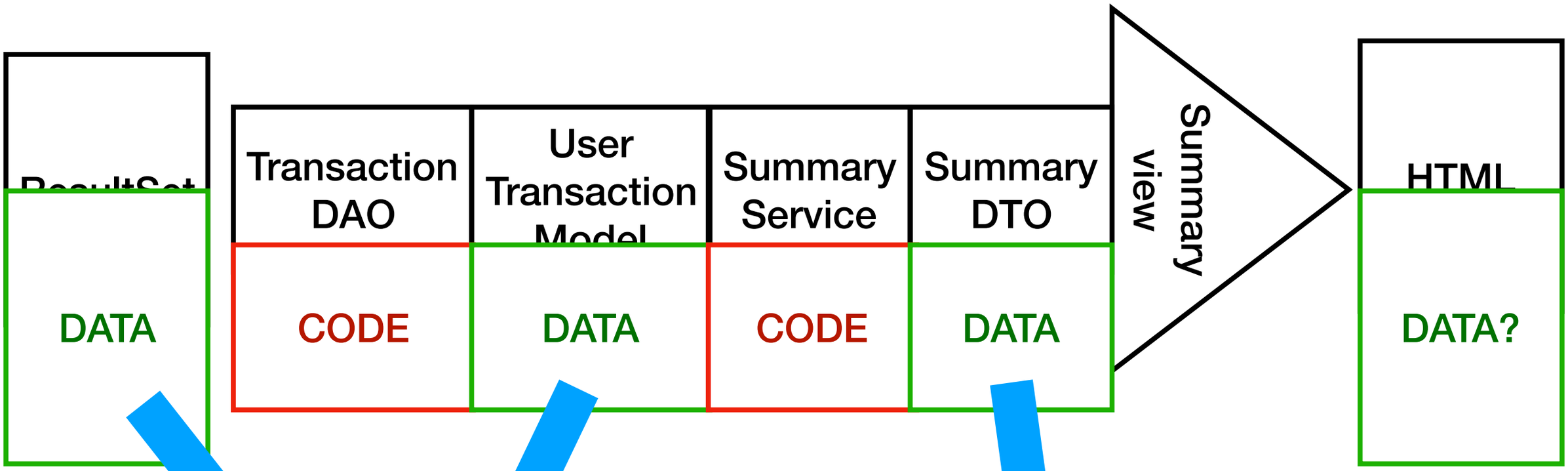
```
{:count 4
 :total 20}
```

id	client	amount
1	a	10
2	a	32
3	b	34
4	c	3
...	...	...
...	...	...
...	...	...
...	...	...
...	...	...
...	...	...
...	...	...
...	...	...



```
[
 [:id :client :amount]
 [1 "a" 10]
 [2 "a" 32]
 [3 "b" 34]
 [4 "c" 3]
]
```

```
[
 {:id 1 :client "a" :amount 10}
 {:id 2 :client "a" :amount 32}
 {:id 3 :client "b" :amount 34}
 {:id 4 :client "c" :amount 3}
]
```




```
[{:id 1 :amount 3}
 {:id 2 :amount 7}
 {:id 3 :amount 4}
 {:id 4 :amount 6}]
```

```
{:count 4
 :total 20}
```

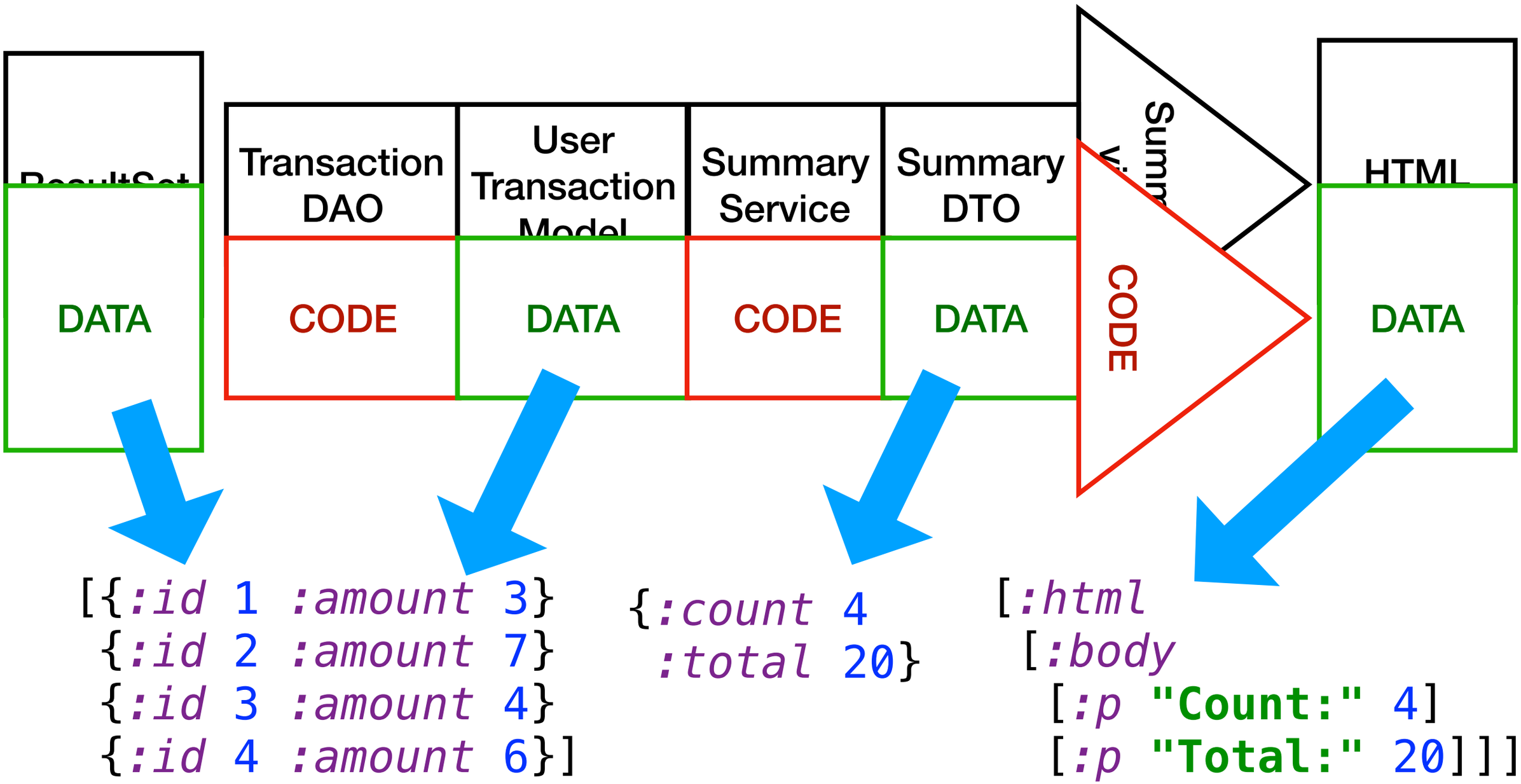
```
<html>
    <body>
        <p>Count: 4</p>
        <p>Total: 20</p>
    </body>
</html>
```

```
<html>
    <body>
        <p>Count: 4</p>
        <p>Total: 20</p>
    </body>
</html>
```

```
[html
    [body
        [p Count: 4]
        [p Total: 20]
    ]
]
```

```
[:html
    [:body
        [:p Count: 4]
        [:p Total: 20]
    ]
]
```

```
[:html
    [:body
        [:p "Count: 4"]
        [:p "Total: 20"]
    ]
]
```



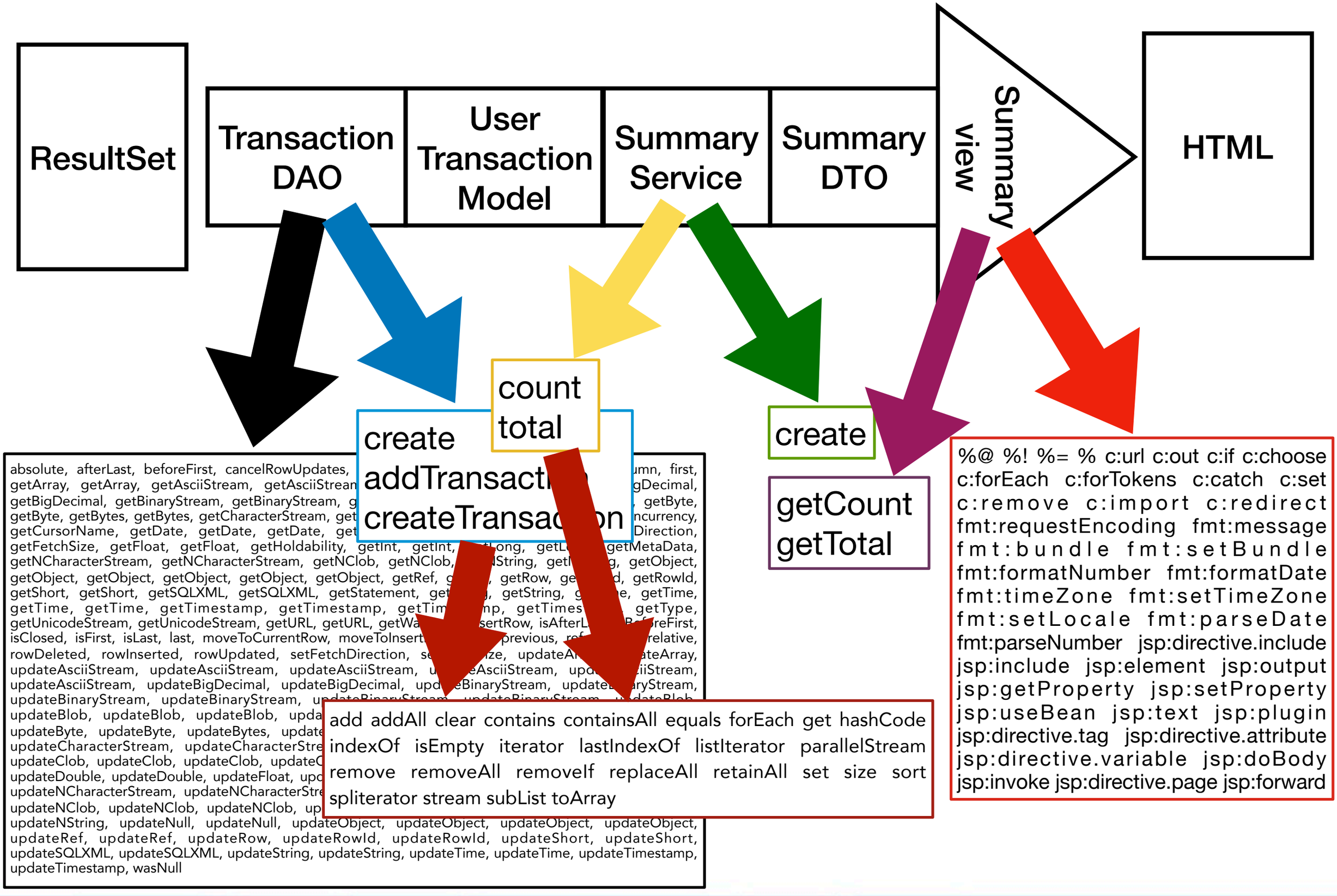

```
[{:id 1 :amount 3}
 {:id 2 :amount 7}
 {:id 3 :amount 4}
 {:id 4 :amount 6}]
```

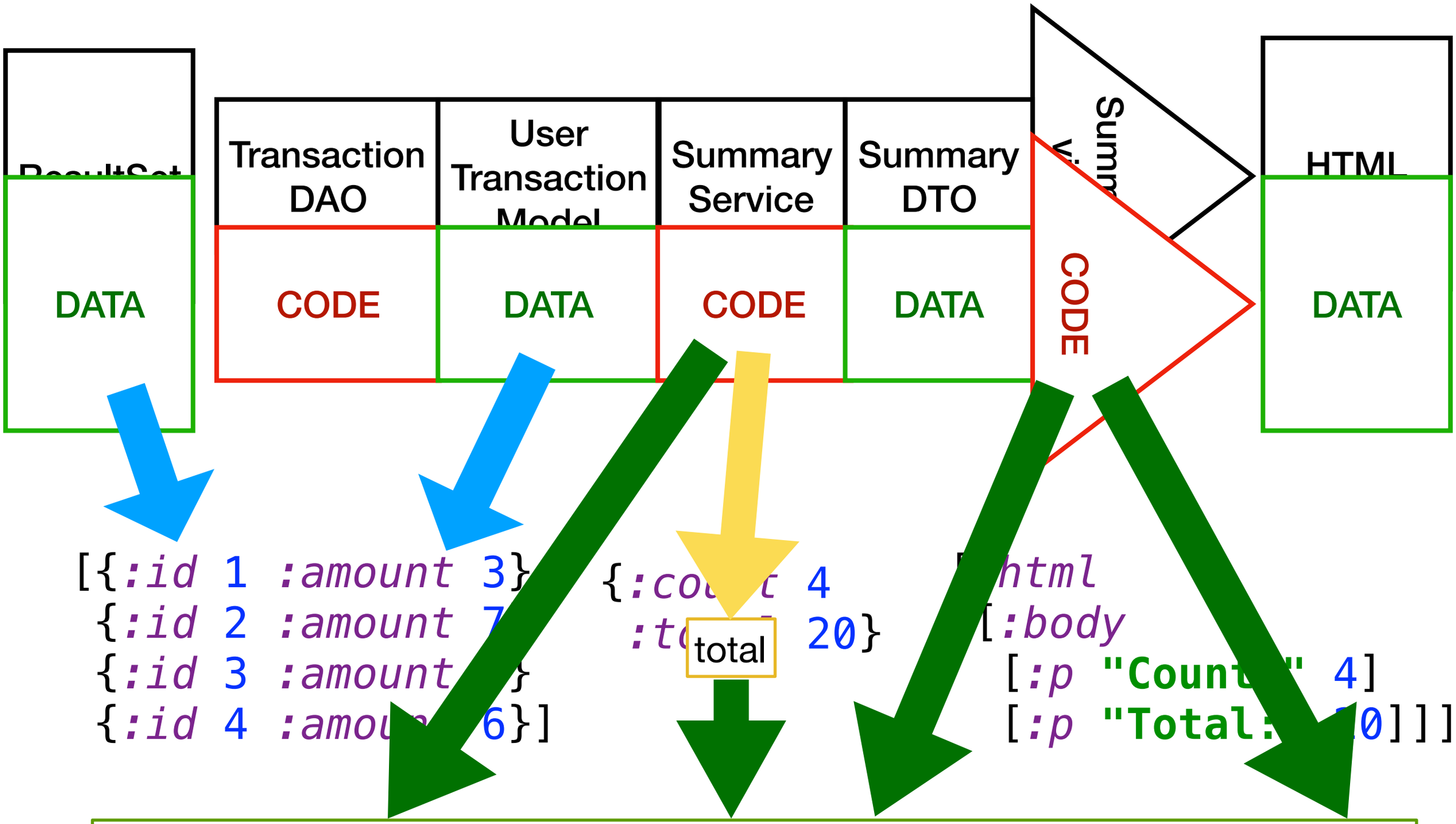
```
{:count 4
 :total 20}
```

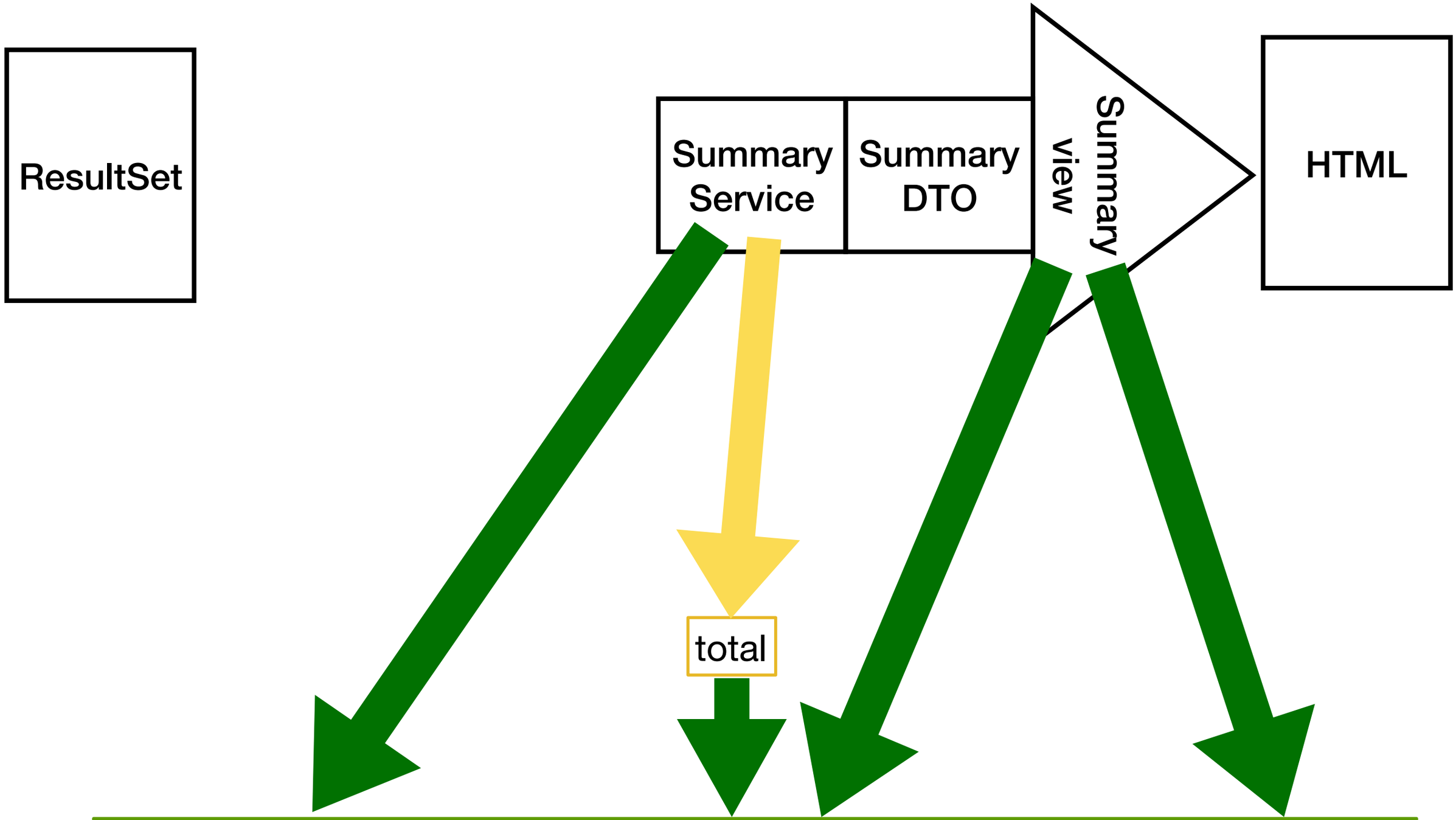
```
[:html
 [:body
  [:p "Count:" 4]
  [:p "Total:" 20]]]
```

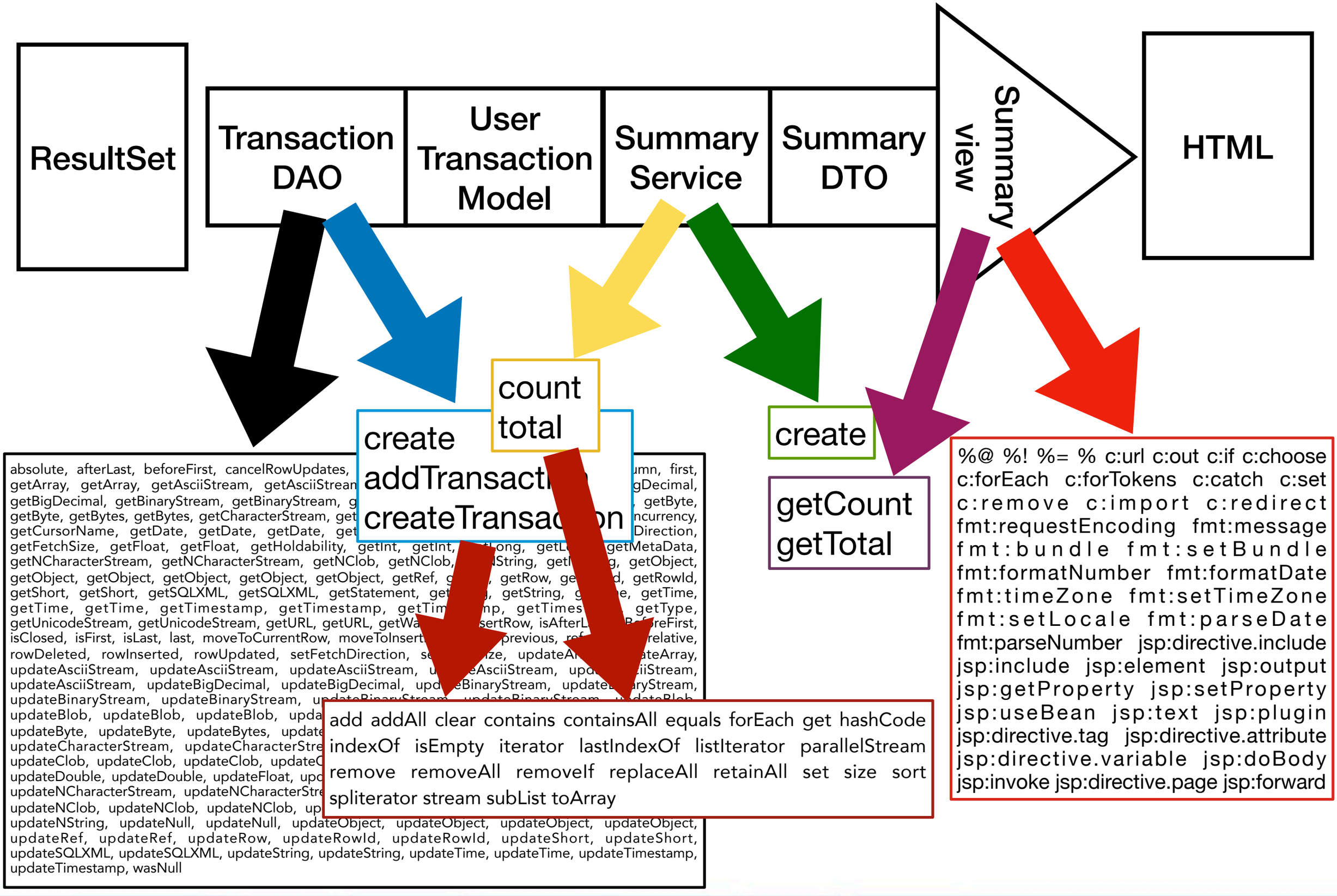
```
[{:id 1 :amount 3}
 {:id 2 :amount 7}
 {:id 3 :amount 4}
 {:id 4 :amount 6}]
```


```
[:html
 [:body
  [:p "Count:" 4]
  [:p "Total:" 20]]]
```


ResultSet
Transaction DAO
User Transaction Model
Summary Service
Summary DTO
Summary view
HTML
count
total
create
addTransacti
createTransac on
create
getCount
getTotal
absolute, afterLast, beforeFirst, cancelRowUpdates,
getArray, getArray, getAsciiStream, getAsciiStrean
getBigDecimal, getBinaryStream, getBinaryStream, g
getByte, getBytes, getBytes, getCharacterStream, get
getCursorName, getDate, getDate, getDate, get
getFetchSize, getFloat, getFloat, getHoldability, getInt, getInt,
getNCharacterStream, getNCharacterStream, getNClob, getNClob,
getObject, getObject, getObject, getObject, getObject, getRef,
getShort, getShort, getSQLXML, getSQLXML, getStatement,
getTime, getTime, getTime, getTimestamp, getTimestamp,
getUnicodeStream, getUnicodeStream, getURL, getURL,
isClosed, isFirst, isLast, last, moveToCurrentRow,
rowDeleted, rowInserted, rowUpdated, setFetchDirection,
updateAsciiStream, updateAsciiStream, updateAsciiStream,
updateAsciiStream, updateBigDecimal, updateBigDecimal,
updateBinaryStream, updateBinaryStream,
updateBlob, updateBlob, updateBlob,
updateByte, updateByte, updateBytes,
updateCharacterStream,
updateClob, updateClob, updateClob,
updateDouble, updateDouble, updateFloat,
updateNCharacterStream,
updateNClob, updateNClob, updateNClob,
updateNString, updateNull, updateNull, updateObject, updateObject, updateObject, updateObject,
updateRef, updateRef, updateRow, updateRowId, updateRowId, updateShort, updateShort,
updateSQLXML, updateSQLXML, updateString, updateString, updateTime, updateTime, updateTimestamp,
updateTimestamp, wasNull
add addAll clear contains containsAll equals forEach get hashCode indexOf isEmpty iterator lastIndexOf listIterator parallelStream remove removeAll removeIf replaceAll retainAll set size sort spliterator stream subList toArray
%@ %! %= % c:url c:out c:if c:choose c:forEach c:forTokens c:catch c:set c:remove c:import c:redirect fmt:requestEncoding fmt:message fmt:bundle fmt:setBundle fmt:formatNumber fmt:formatDate fmt:timeZone fmt:setTimeZone fmt:setLocale fmt:parseDate fmt:parseNumber jsp:directive.include jsp:include jsp:element jsp:output jsp:getProperty jsp:setProperty jsp:useBean jsp:text jsp:plugin jsp:directive.tag jsp:directive.attribute jsp:directive.variable jsp:doBody jsp:invoke jsp:directive.page jsp:forward


ResultSet
DATA
Transaction DAO
CODE
User Transaction Model
DATA
Summary Service
CODE
Summary DTO
DATA
Summary View
CODE
HTML
DATA
[{:id 1 :amount 3}
{:id 2 :amount 7
{:id 3 :amount
{:id 4 :amou 6}]
{:count 4
:to 20}
total
[:html
[:body
[:p "Count" 4]
[:p "Total: 20]]]
apply butlast concat cons count cycle diff distinct distinct? drop drop-last drop-while empty empty? every? ffirst filter first flatten fnext for frequencies group-by interleave interpose into into-array keep keep-indexed last lazy-cat map map-indexed mapcat next nfirst nnext not-any? not-empty not-every? nth nthnext partition partition-all partition-by pmap postwalk prewalk rand-nth reduce reductions remove replace rest reverse second seq? seque set shuffle some sort sort-by split-at split-with take take-nth take-while to-array-2d vec walk when-first assoc pop subvec replace conj rseq update-in update get get-in contains? find keys vals assoc assoc-in dissoc merge merge-with select-keys update-in update rename-keys map-invert reduce-kv dissoc-in disj join select project union difference intersection index


ResultSet
Summary Service
Summary DTO
Summary view
HTML
total
apply butlast concat cons count cycle diff distinct distinct? drop drop-last drop-while empty empty? every? ffirst filter first flatten fnext for frequencies group-by interleave interpose into into-array keep keep-indexed last lazy-cat map map-indexed mapcat next nfirst nnext not-any? not-empty not-every? nth nthnext partition partition-all partition-by pmap postwalk prewalk rand-nth reduce reductions remove replace rest reverse second seq? seque set shuffle some sort sort-by split-at split-with take take-nth take-while to-array-2d vec walk when-first assoc pop subvec replace conj rseq update-in update get get-in contains? find keys vals assoc assoc-in dissoc merge merge-with select-keys update-in update rename-keys map-invert reduce-kv dissoc-in disj join select project union difference intersection index


ResultSet
Transaction DAO
User Transaction Model
Summary Service
Summary DTO
Summary view
HTML
count
total
create
addTransacti
createTransac on
create
getCount
getTotal
absolute, afterLast, beforeFirst, cancelRowUpdates,
getArray, getArray, getAsciiStream, getAsciiStrean
getBigDecimal, getBinaryStream, getBinaryStream, g
getByte, getBytes, getBytes, getCharacterStream, get
getCursorName, getDate, getDate, getDate, get
getFetchSize, getFloat, getFloat, getHoldability, getInt, getInt,
getNCharacterStream, getNCharacterStream, getNClob, getNClob,
getObject, getObject, getObject, getObject, getObject, getRef,
getShort, getShort, getSQLXML, getSQLXML, getStatement,
getTime, getTime, getTime, getTimestamp, getTimestamp,
getUnicodeStream, getUnicodeStream, getURL, getURL,
isClosed, isFirst, isLast, last, moveToCurrentRow,
rowDeleted, rowInserted, rowUpdated, setFetchDirection,
updateAsciiStream, updateAsciiStream, updateAsciiStream,
updateAsciiStream, updateBigDecimal, updateBigDecimal,
updateBinaryStream, updateBinaryStream,
updateBlob, updateBlob, updateBlob,
updateByte, updateByte, updateBytes,
updateCharacterStream,
updateClob, updateClob, updateClob,
updateDouble, updateDouble, updateFloat,
updateNCharacterStream,
updateNClob, updateNClob, updateNClob,
updateNString, updateNull, updateNull, updateObject, updateObject, updateObject, updateObject,
updateRef, updateRef, updateRow, updateRowId, updateRowId, updateShort, updateShort,
updateSQLXML, updateSQLXML, updateString, updateString, updateTime, updateTime, updateTimestamp,
updateTimestamp, wasNull
add addAll clear contains containsAll equals forEach get hashCode indexOf isEmpty iterator lastIndexOf listIterator parallelStream remove removeAll removeIf replaceAll retainAll set size sort spliterator stream subList toArray
%@ %! %= % c:url c:out c:if c:choose c:forEach c:forTokens c:catch c:set c:remove c:import c:redirect fmt:requestEncoding fmt:message fmt:bundle fmt:setBundle fmt:formatNumber fmt:formatDate fmt:timeZone fmt:setTimeZone fmt:setLocale fmt:parseDate fmt:parseNumber jsp:directive.include jsp:include jsp:element jsp:output jsp:getProperty jsp:setProperty jsp:useBean jsp:text jsp:plugin jsp:directive.tag jsp:directive.attribute jsp:directive.variable jsp:doBody jsp:invoke jsp:directive.page jsp:forward

"It is better to have 100 functions operate on one data structure than 10 functions on 10 data structures." — *Alan Perlis*

```
[{:id 1 :amount 3}
 {:id 2 :amount 7}
 {:id 3 :amount 4}
 {:id 4 :amount 6}]
```


```
[:html
 [:body
  [:p "Count:" 4]
  [:p "Total:" 20]]]
```

DATA?

ResultSet

```
[{:id 1 :client 32 :amount 3}
 {:id 2 :client 87 :amount 7}
 {:id 3 :client 32 :amount 4}
 {:id 4 :client 40 :amount 6}]
```

UserTransactionsModel

```
[{:id 1 :client 32 :amount 3}
 {:id 2 :client 87 :amount 7}
 {:id 3 :client 32 :amount 4}
 {:id 4 :client 40 :amount 6}]
```

UI Table

```
[{:id 1 :client 32 :amount 3}
 {:id 2 :client 87 :amount 7}
 {:id 3 :client 32 :amount 4}
 {:id 4 :client 40 :amount 6}]
```

CSV

```
[{:id 1 :client 32 :amount 3}
 {:id 2 :client 87 :amount 7}
 {:id 3 :client 32 :amount 4}
 {:id 4 :client 40 :amount 6}]
```

JMS Queue

```
[{:id 1 :client 32 :amount 3}
 {:id 2 :client 87 :amount 7}
 {:id 3 :client 32 :amount 4}
 {:id 4 :client 40 :amount 6}]
```

HTML

```
[:html
     [:body
          [:p "Count: 4"]
          [:p "Total: 20"]]]
```

CSS

```
[:h1
 [:a
  {:font-weight      :normal
   :text-decoration :none}
  [:&:hover
   {:font-weight      :bold
    :text-decoration :underline}]]]
```

HTTP Request

```
{:request-method :get
 :uri                  “/foobaz"
 :query-params         {"somekey" “somevalue"}
 :headers              {"accept-encoding" "gzip, deflate"
                        "connection"      "close"}
 :body                 nil
 :scheme               :http
 :content-length 0
 :server-port          8080
 :server-name          “localhost"
}
```

Client HTTP Request

```
{:method          :get
 :url             “http://localhost:8080/foobaz"
 :query-params    {"somekey" "somevalue"}
 :headers         {"accept-encoding" "gzip, deflate"
                   "connection"      “close"}
 :body            nil
}
```

HTTP Request

```
{:request-method :get
 :uri                  “/foobaz"
 :query-params         {"somekey" “somevalue"}
 :headers              {"accept-encoding" "gzip, deflate"
                        "connection"      "close"}
 :body                 nil
 :scheme               :http
 :content-length 0
 :server-port          8080
 :server-name          “localhost"
}
```

Client HTTP Request

```
{:method          :get
 :url             “http://localhost:8080/foobaz"
 :query-params    {"somekey" "somevalue"}
 :headers         {"accept-encoding" "gzip, deflate"
                   "connection"      “close"}
 :body            nil
}
```

HTTP Response

```
{:status  200
 :headers {"Content-Type" "text/html"}
 :body    "Hello World"}
```

Client HTTP Response

```
{:status  200
 :headers {"Content-Type" "text/html"}
 :body    "Hello World"}
```

SQL Query

```
{:select [:id :client :amount]
 :from   [:transactions]
 :where  [:= :client "a"]}
```

SQL Insert

```
{:insert-into :transactions
 :values      [{:id 1, :client "a" :amount 3}
               {:id 2, :client "a" :amount 7}
               {:id 3, :client "b" :amount 4}
               {:id 4, :client "c" :amount 6}]}
```

Project

```
{:name           "my-project"
 :version        "0.0.1-SNAPSHOT"
 :compile-path   "target/classes"
 :repositories   [["central" {:url "https://repo1.maven.org/maven2/"
                              :snapshots false}]]
 :source-paths   ["src"]
 :resource-paths ["resources"]
 :test-paths     ["test"]
 :target-path    "target"
 :prep-tasks     ["javac" "compile"]
 :dependencies   [[http.async.client/http.async.client "1.1.0"]
                  [ch.qos.logback/logback-classic "1.1.5"]
                  [ch.qos.logback/logback-core "1.1.5"]
                  [org.slf4j/slf4j-api "1.7.16"]
                  [com.google.guava/guava "19.0"]]}
```

Configuration

```
{:web-server                {:listen 8080}
 :db-config                 {:dbname   "mypgdatabase"
                             :host     "xxxx"
                             :user     "xxxx"
                             :password "xxxx"}
 :http-defaults             {:connection-timeout 10000
                             :request-timeout    10000
                             :max-connections    2000}
 :transaction-service       {:url "http://transactions"}
 :user-service              {:url                "http://transactions"
                             :connection-timeout 1000}}
```

XML

```
{:tag     :Person
 :content [{:tag     :name
            :content ["Dan"]}
           {:tag     :user
            :content ["ABC123"]}
          {:tag     :transactions
           :content [{:tag     :transaction
                      :content [{:tag     :id
                                 :content [1]}
                                {:tag     :amount
                                 :content [10]}]}]}]}
```

DATA!

Metadata

```
{:id          :string
 :name        :string
 :description :string
 :ingredients [{:id       :string
                :quantity :int
                :unit     #{:LITRE
                            :SPOON
                            :CUP
                            :GALLON}}]}
```

Metadata

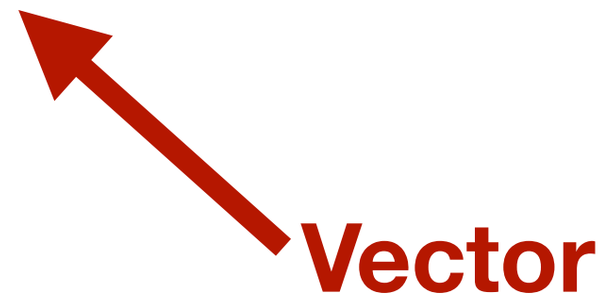
```
RecipeSummary = selectKeys(RecipeDto,[:id :name])
RecipeSummary =  remove(RecipeDto,[:ingredients])
RecipeSummary =
              putAll(
                selectKeys(RecipeDto,[:id :name]),
                selectKeys(SocialDto,[:stars :value]))

RecipeSummaryV2 = put(RecipeSummary, :time, :long)
```

CODE

```
selectKeys(user [:id :name])
```

```clojure
(selectKeys user [:id :name])
```



List

```
selectKeys(user [:id :name])
```

Vector

List

```
selectKeys(user [:id :name])
```

Vector

List

```
selectKeys(user [:id :name])
```

Symbol

Vector

List

```
selectKeys(user [:id :name])
```

Symbol

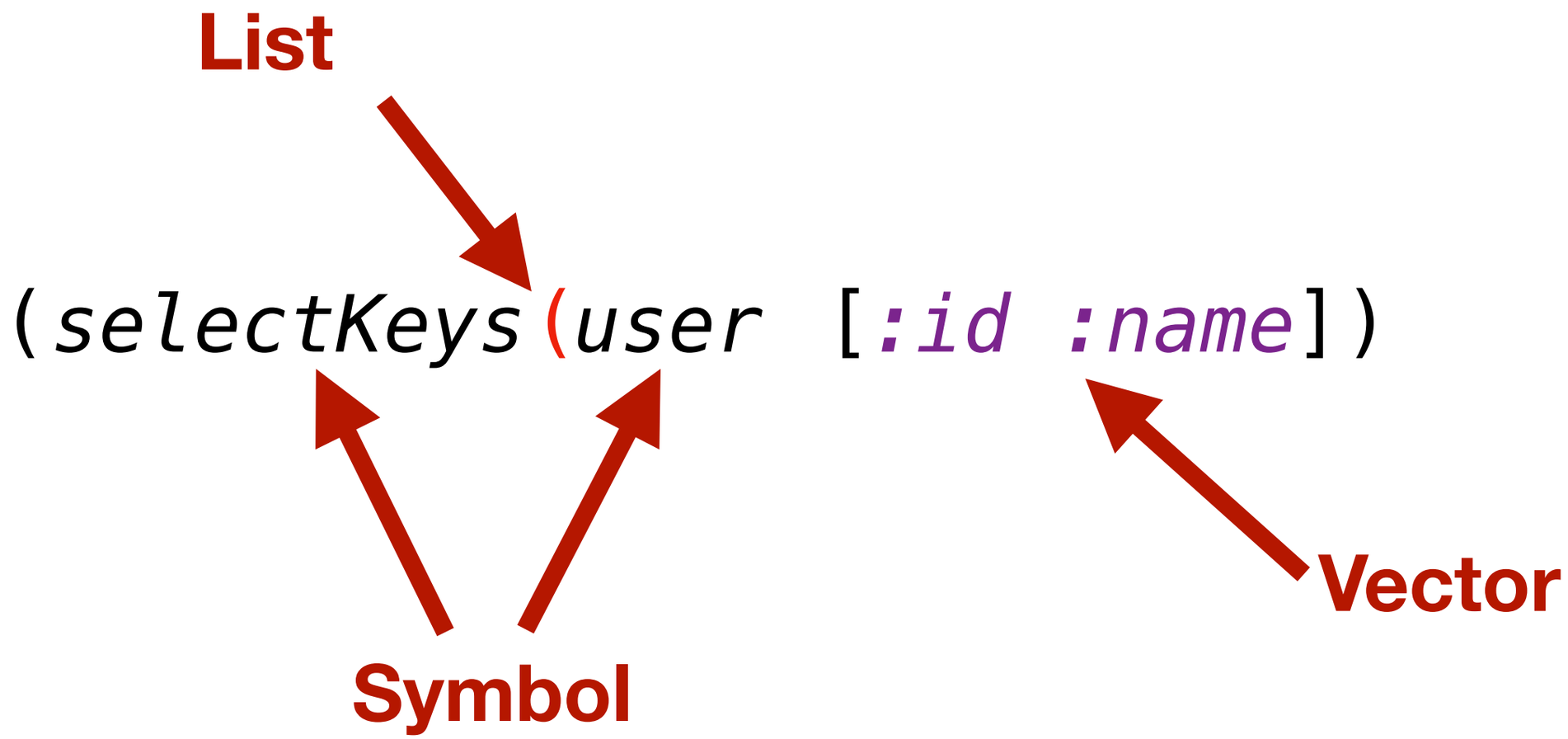
Vector

List

```
selectKeys(user [:id :name])
```

Symbol

Vector


List
(selectKeys (user [:id :name])
Symbol
Vector

Code

```
(function param1 param2 …)
```

Code

```
(defclass User
  :string name
  :string surname)
```

Metaprogramming

```
($= 1 + 2)
```

```
(+ 1 2)
```

```
(defmacro $= [op1 op op2]
  (list op op1 op2))
```

```java
BufferedReader br = new BufferedReader(new FileReader(""));
try {
    return br.readLine();
} finally {
    br.close();
}
```

```java
try (BufferedReader br = new BufferedReader(new FileReader(""))) {
    return br.readLine();
}
```

```clojure
(defmacro with-open [bindings & body]
  (cond
    (= (count bindings) 0) `(do ~@body)
    (symbol? (bindings 0)) `(let ~(subvec bindings 0 2)
                              (try
                                (with-open ~(subvec bindings 2) ~@body)
                                (finally
                                  (. ~(bindings 0) close))))))
```

```clojure
(match [(mod n 3) (mod n 5)]
       [0 0] "FizzBuzz"
       [0 _] "Fizz"
       [_ 0] "Buzz"
       :else n)
```

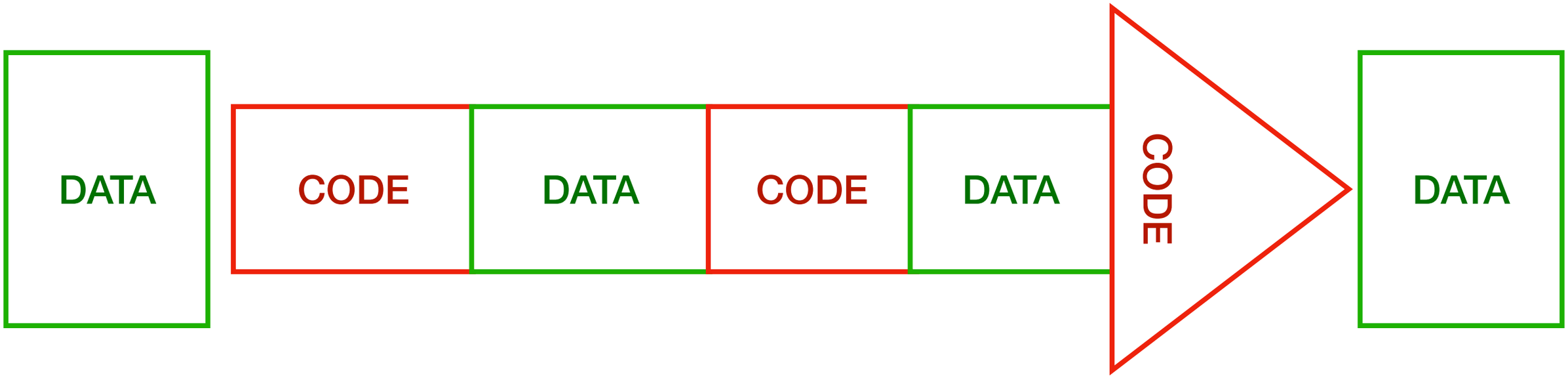
```clojure
(defun say-hi
    ([:dennis] "Hi,good morning, dennis.")
    ([:catty] "Hi, catty, what time is it?")
    ([:green] "Hi,green, what a good day!")
    ([other] (str "Say hi to " other)))
```

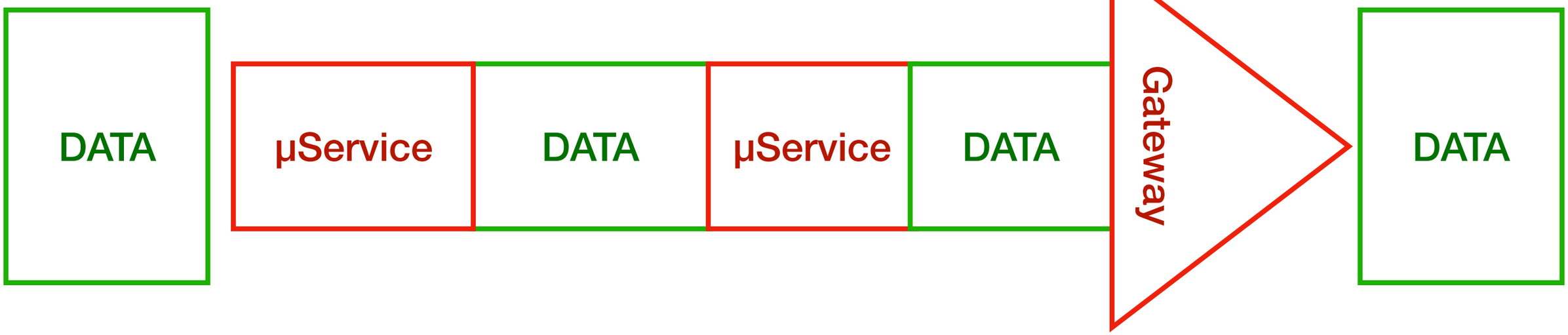
```clojure
(try+
  ...
  (catch [:status 404] _ (log "404!"))
  (catch [:status 500] _ (log "500!"))
  (catch IOException _ (log "ops!"))
  (catch Object _ (log "booo")))
```

```
(let-flow [a (http-call-1)
           b (http-call-2)]
  (+ a b))
```

```
(go
  (let [recipe (<! (find-recipe))
        price (<! (get-price (:id recipe)))]
    (assoc recipe :price price)))
```

DATA ALL THE THINGS!


DATA
CODE
DATA
CODE
DATA
CODE
DATA

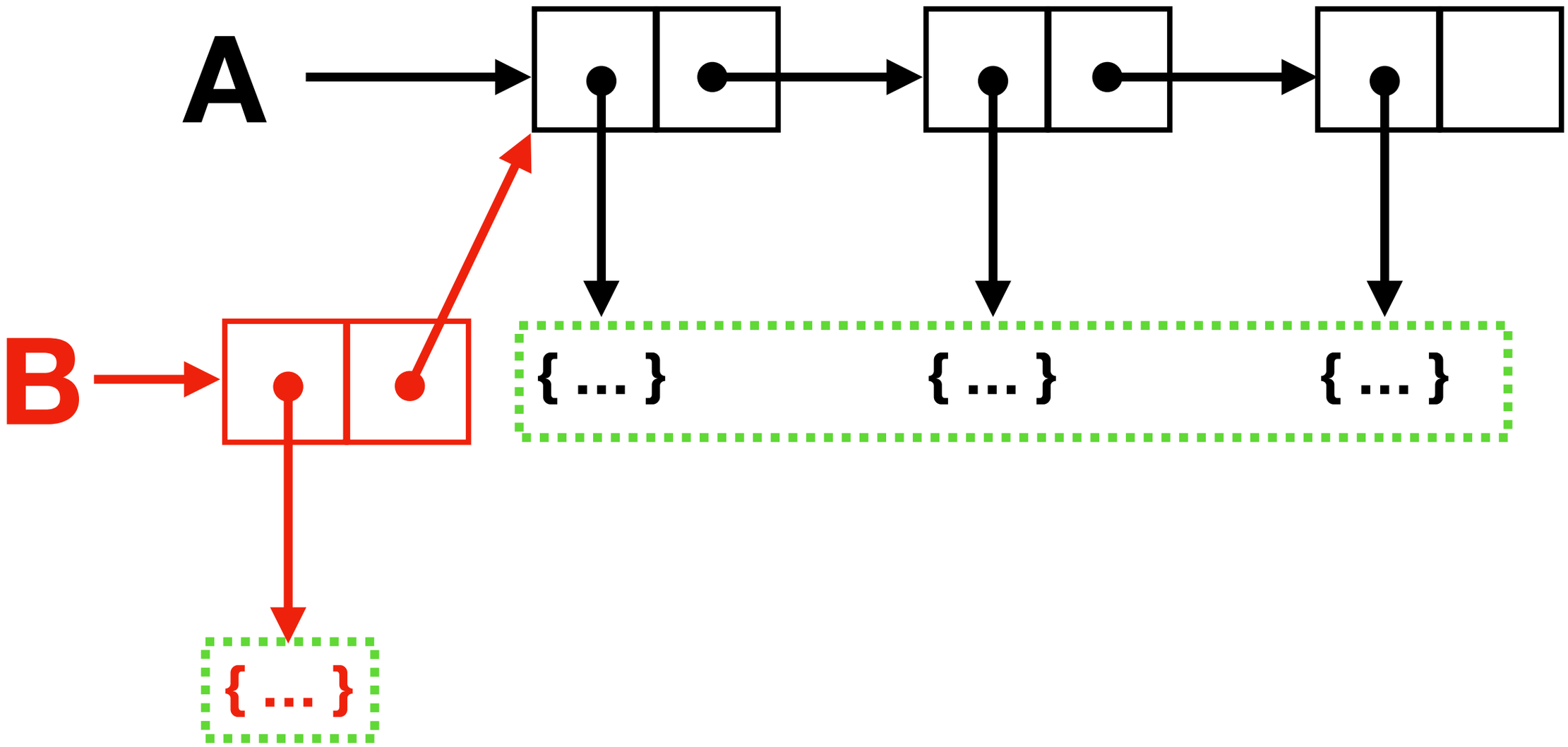

DATA
µService
DATA
µService
DATA
Gateway
DATA

DATA

IS

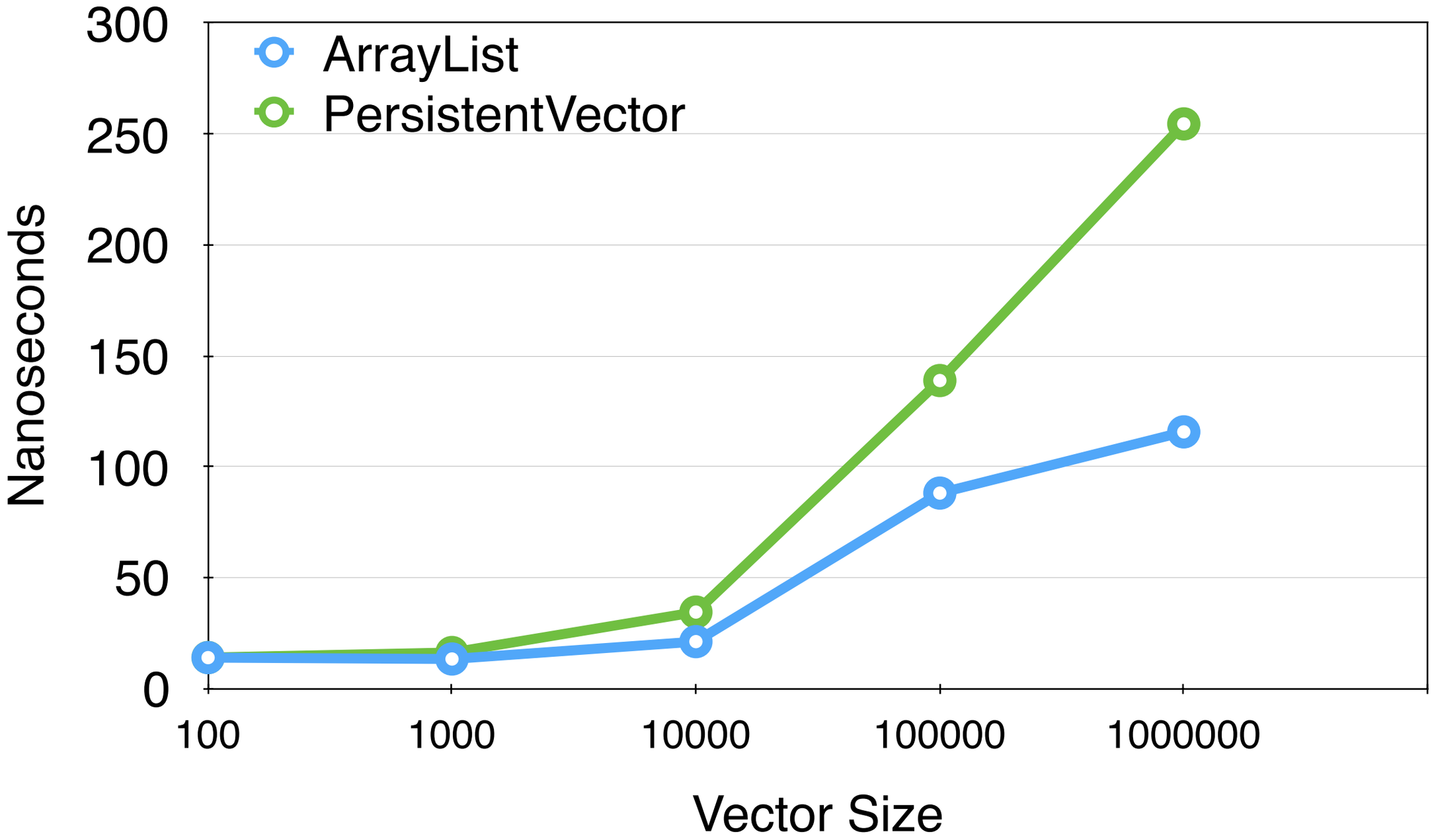
IMMUTABLE

```
"I am a String!"
```

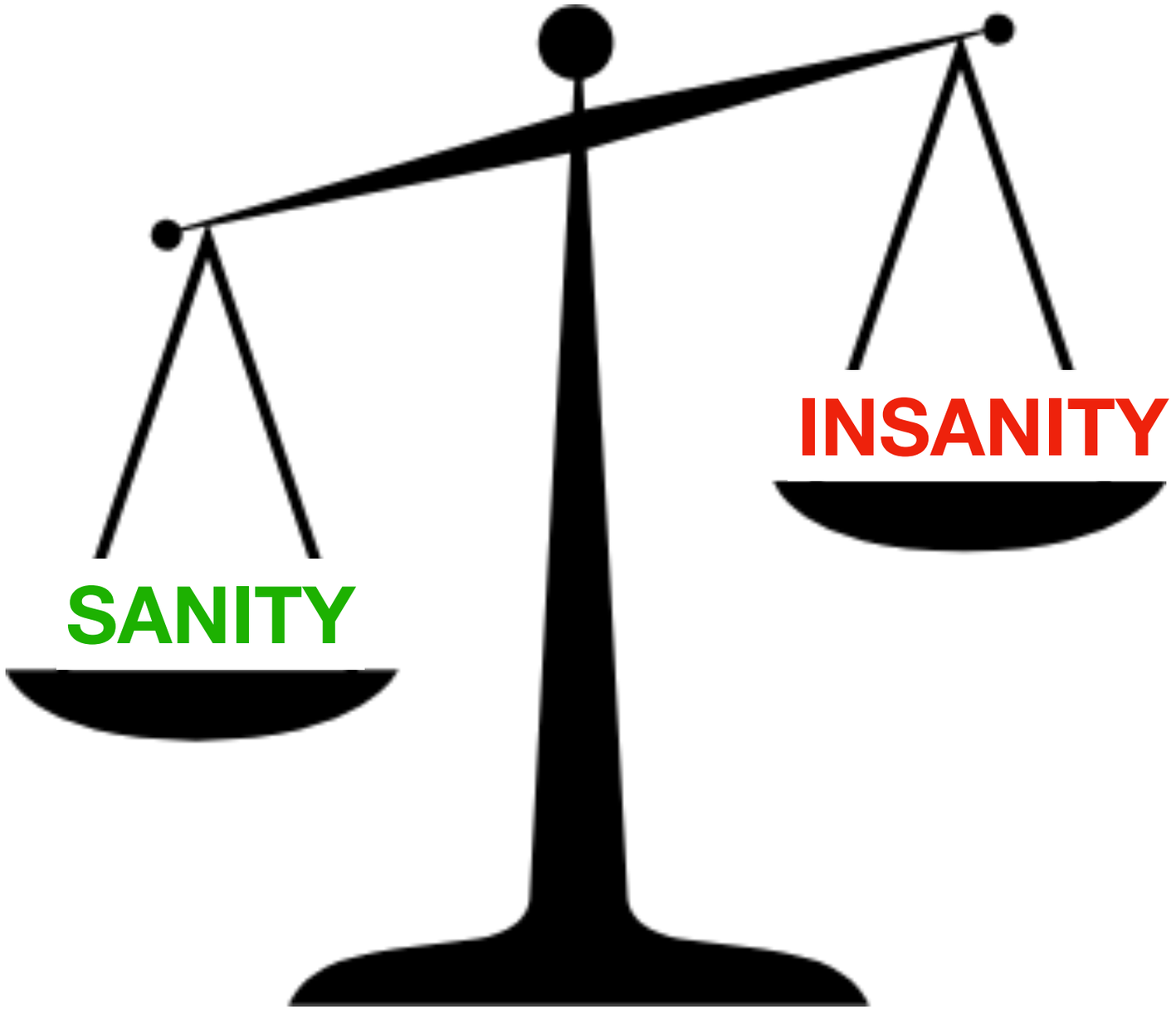

A
B
{ … }
{ … }
{ … }
{ … }

	Persistent List	Persistent Vector	Persistent Map
ADD	O(1) head	O(1) tail	O(log32 n)
REMOVE	O(1) head	O(1) tail	O(log32 n)
GET	O(n)	O(log32 n)	O(log32 n)

O(log32 n)


ArrayList
PersistentVector
Nanoseconds
300
250
200
150
100
50
0
100
1000
10000
100000
1000000
Vector Size

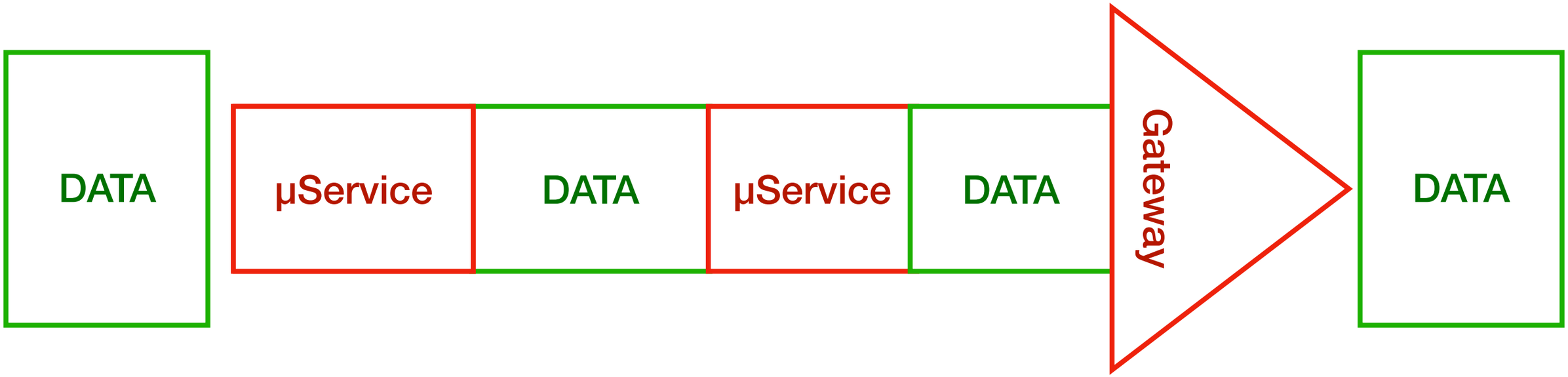

PERFORMANCE
PRODUCTIVITY


INSANITY
SANITY

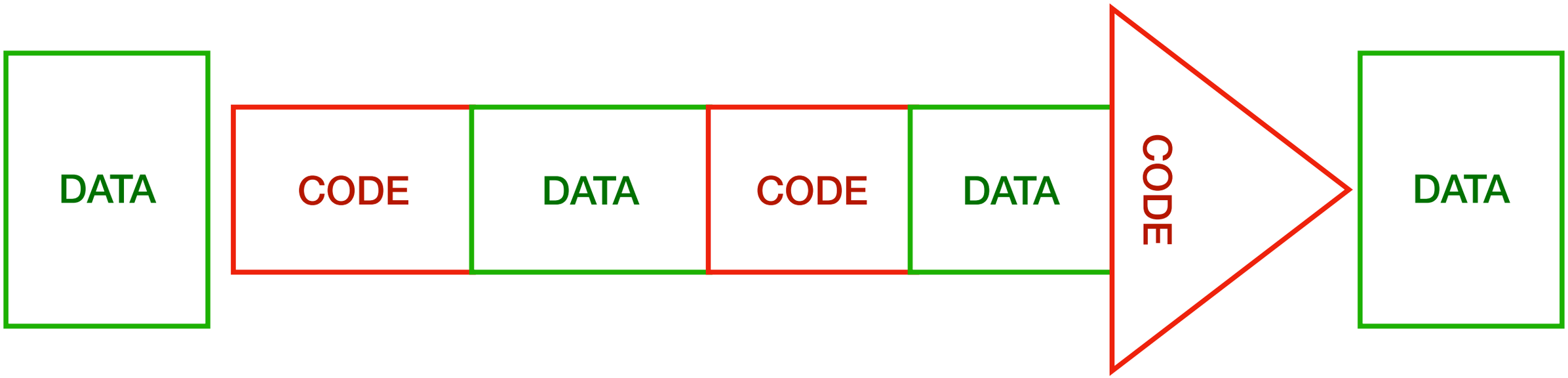
Data Oriented Programming

Data Oriented Programming

Data Oriented Programming



Data Oriented Programming



DATA

Data Oriented Language

numbers → 10, 1.1

boolean → **true** / **false**

string → **“foo”**

keyword → **:color**, **:red**

lists → (1 2 3)

vector → [1 2 3]

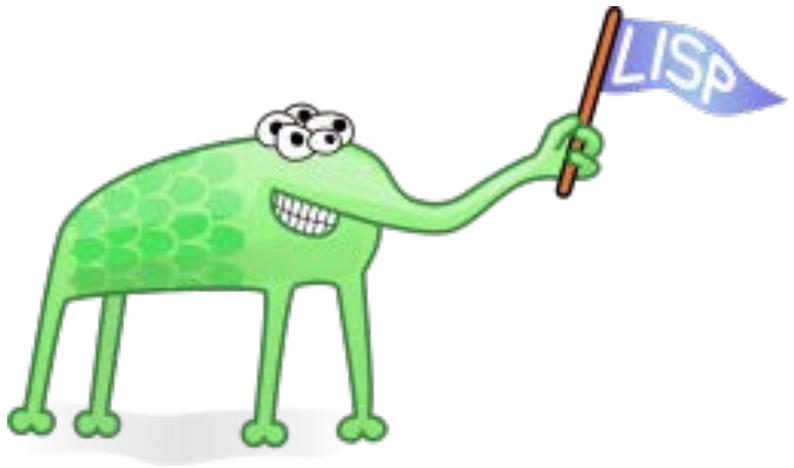
map → {key1 val1, key2 val2}

set → #{e1, e2}

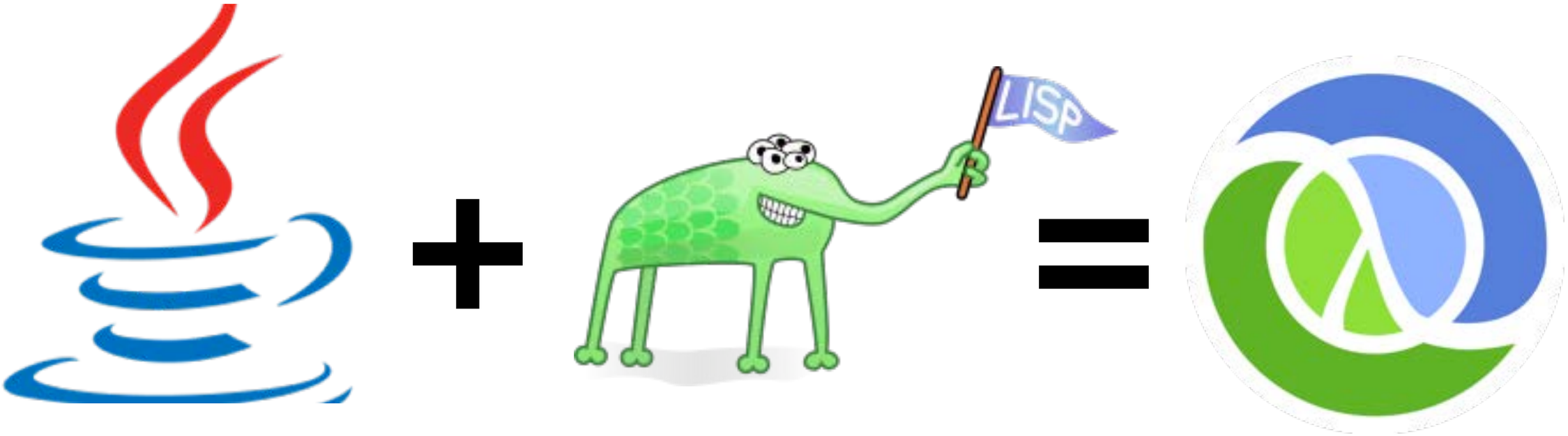
nothing → **nil**

symbol → +, user

Data Oriented Language



Data Oriented Language



Clojure Syntax

numbers → 10, 1.1

boolean → **true** / **false**

string → **“foo”**

keyword → **:color**, **:red**

lists → (1 2 3)

vector → [1 2 3]

map → {key1 val1, key2 val2}

set → #{e1, e2}

nothing → **nil**

symbol → +, user

???

@DanLebrero

dlebrero@gmail.com

danlebrero.com

akvo.org

@Akvo

https://akvo.org/