Building hypermedia clients

Todd Brackley @toddbNZ
Hypr @hyprNZ

Business issues & tech goals

- We need to provision an entire network of data (in minutes—not days)
- Really needed to expose real data (to show what's going)
- Model the business processes but defer GUI (in-place editing)

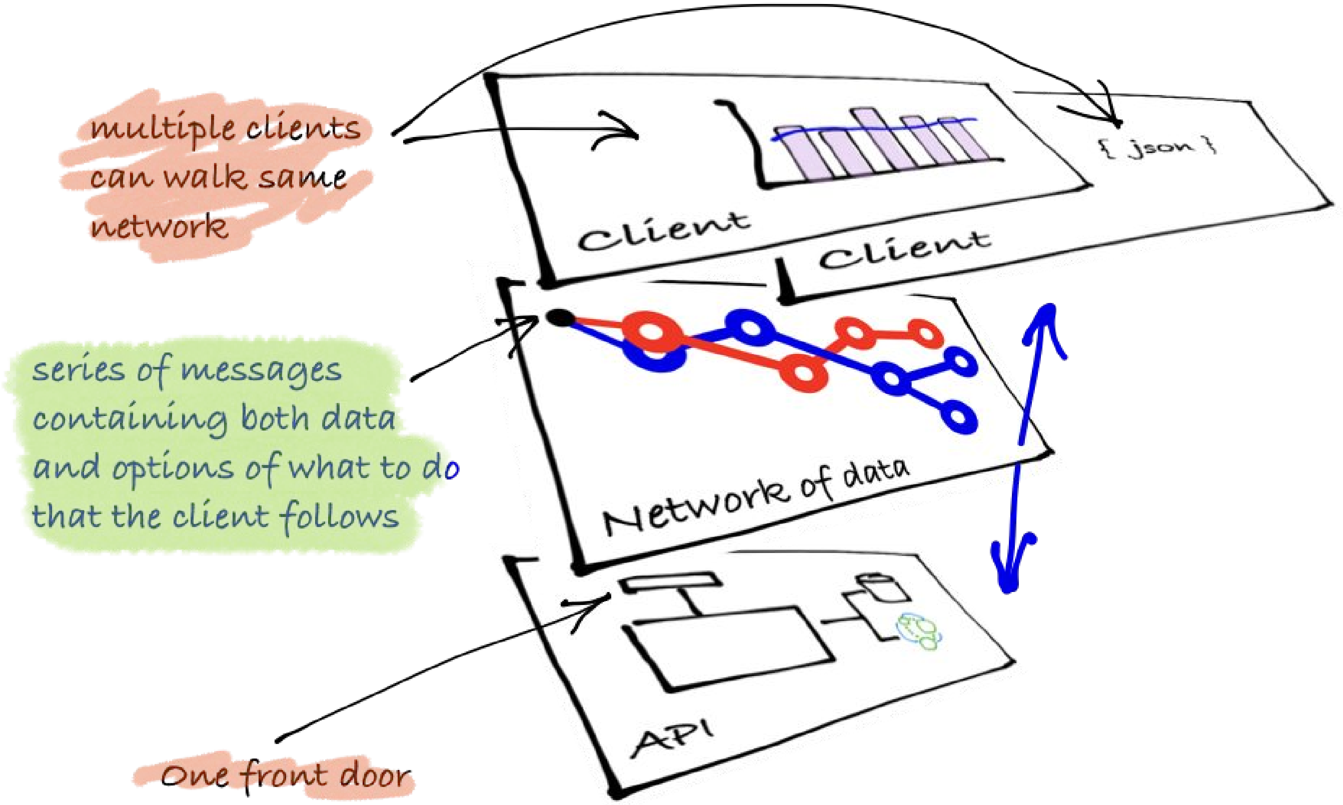
- Underlying engine should be the same regardless of presentation
- Changes to the server can extend the client (use forms as affordances)

Structure

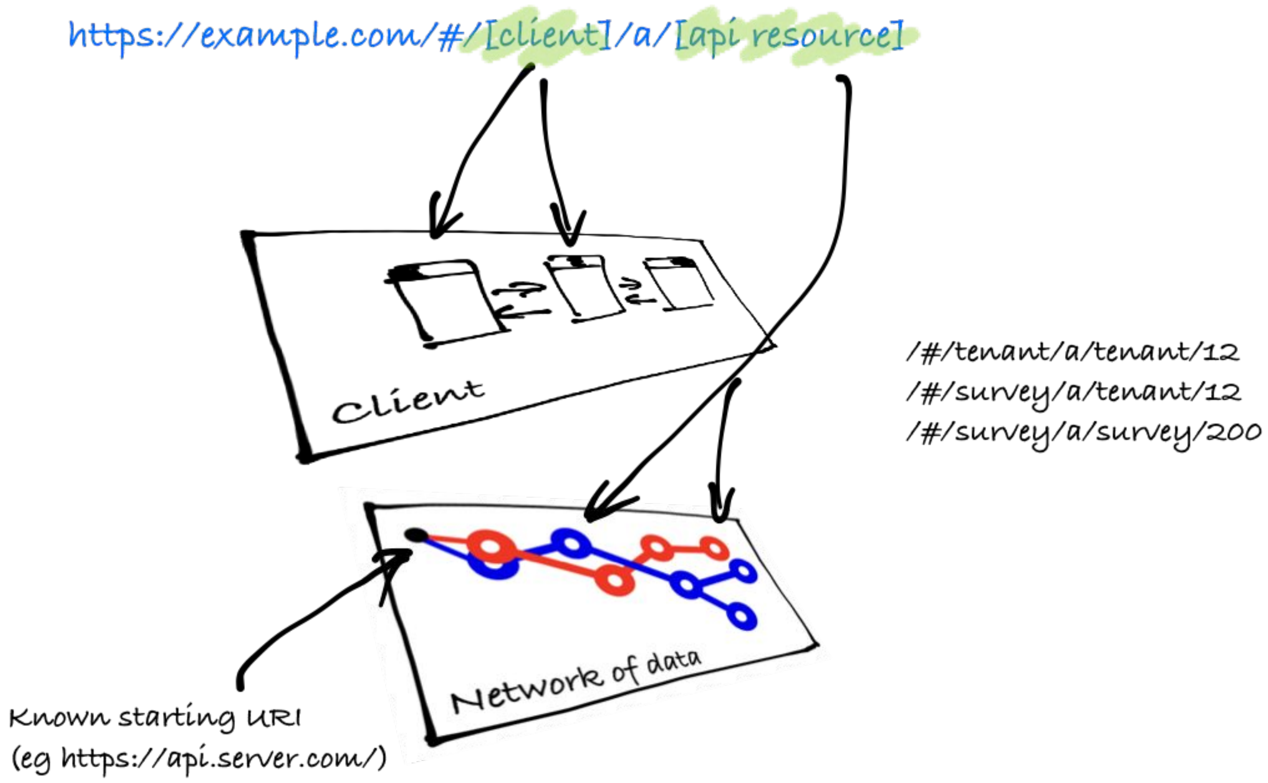
1. Walking and updating network of data
2. Using forms to know what to process
3. Some client design issues

Walking the network of data

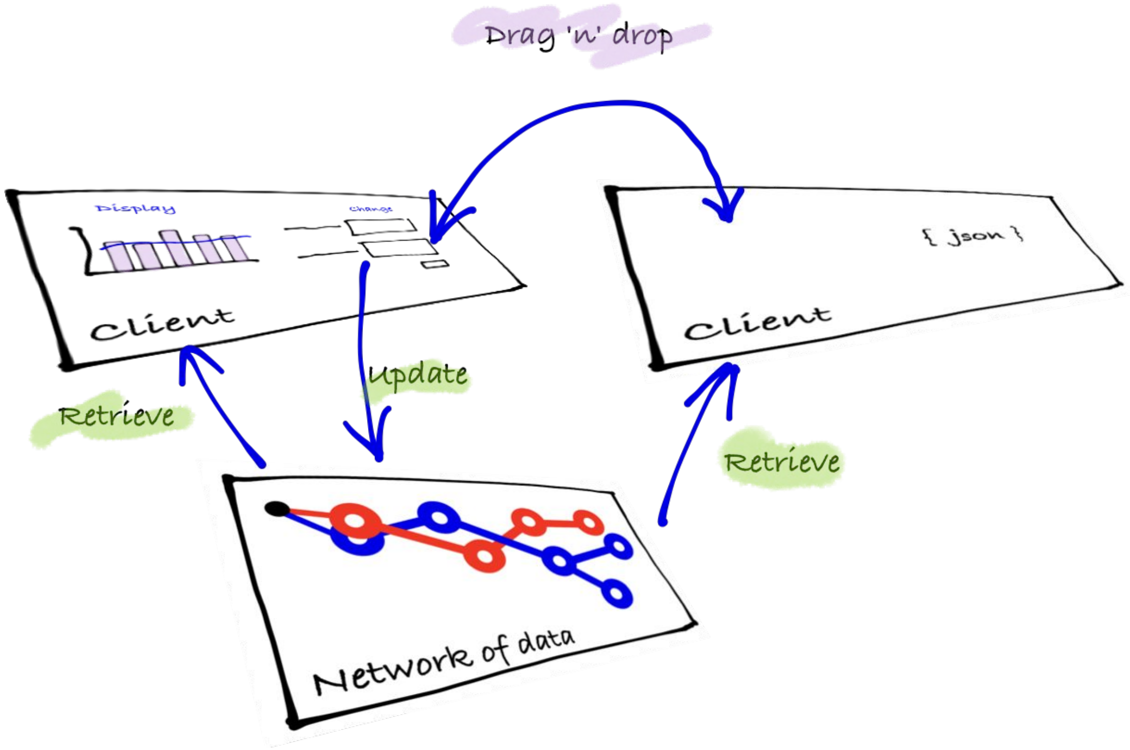
Keep a separation between client and API


multiple clients can walk same network
series of messages containing both data and options of what to do that the client follows
One front door
{ json }
Client
Client
Network of data
API

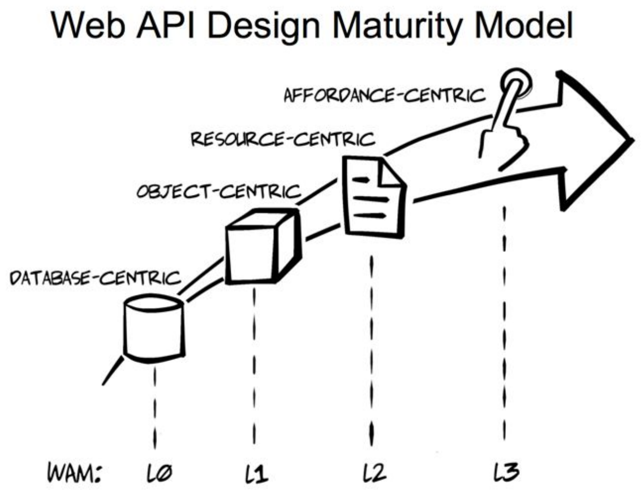
Bookmarkable URI holding state



Lots of ways to walk the API



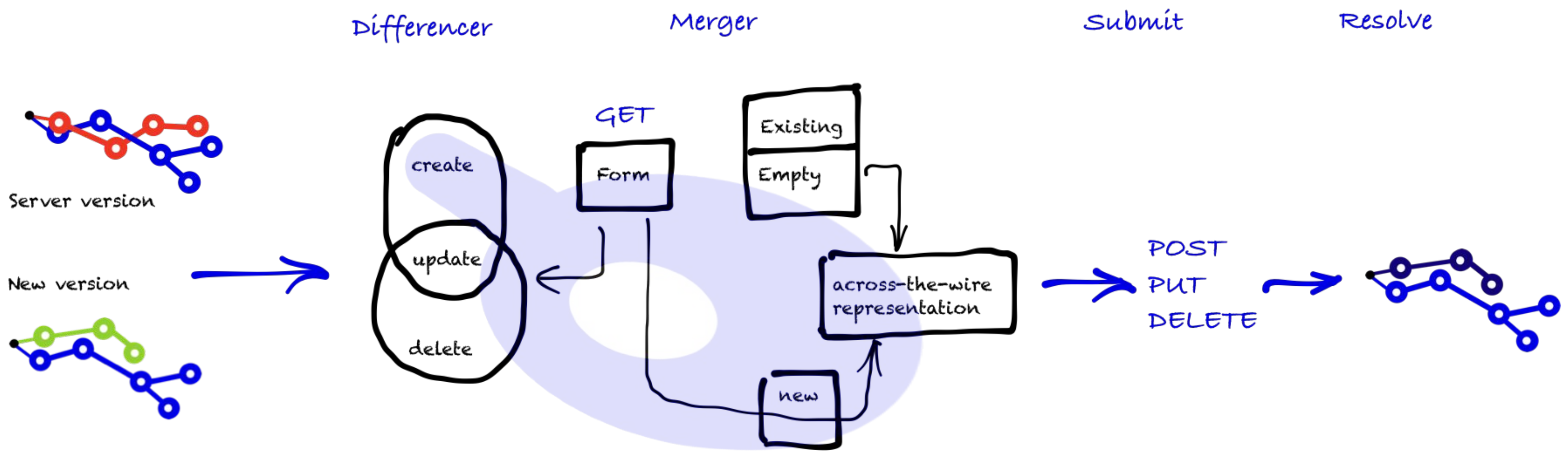
Using forms as affordances



- updates to resources and forms are controlled at the server
- no changes needed on the client except for changes to the network structure
- forms tell the client how to process properties on in-memory resources

Amundsen, http://amundsen.com/talks/2016-06-wsrest/2016-06-WADM.pdf

General engine



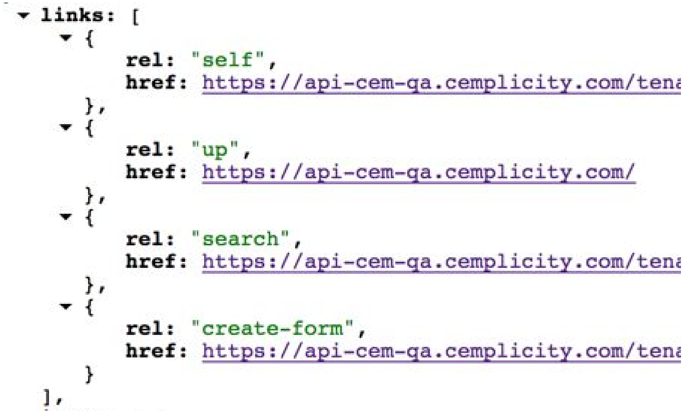
Forms: you’ll need to cater for

- Single (by value) [text/numbers/passwords/creditcard)
- Enumeration (select by value or by reference) [single/multiple]
- Groups (containers of above–including recursive)

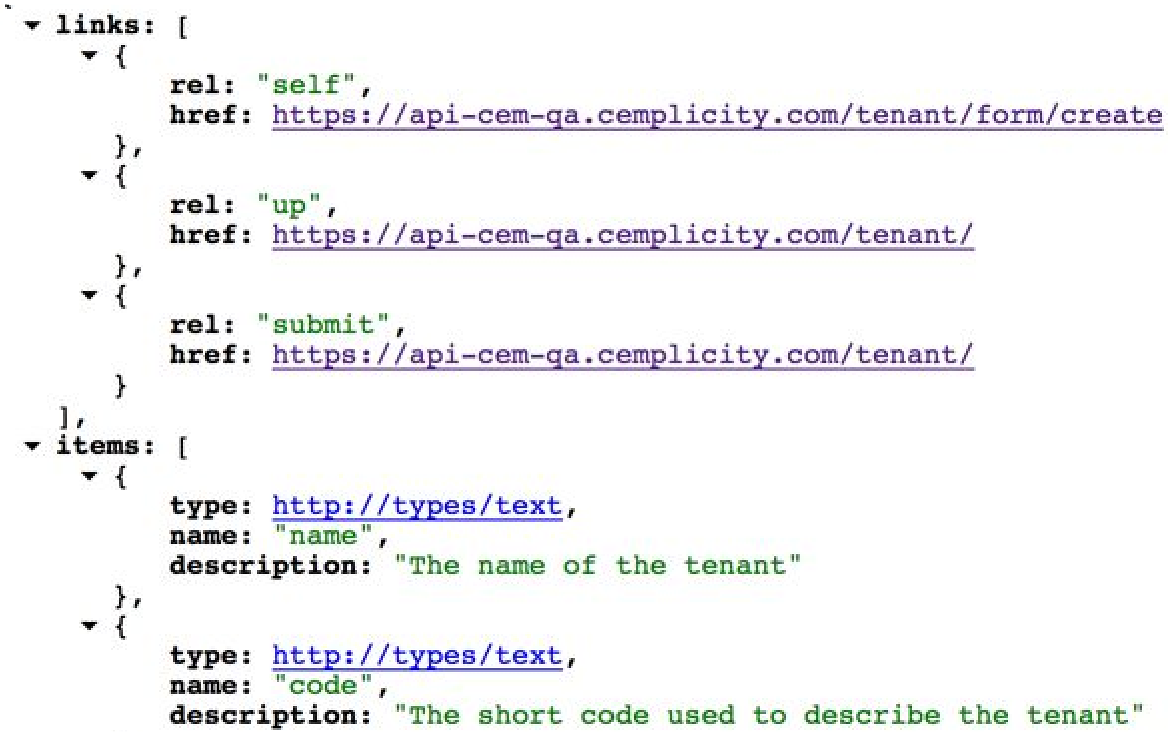
There are plenty of forms specifications

- HAL-FORMS
- Cj
- SIREN
- JSON-LD + hydra
- UBER
- We are using atom-like+json

Here’s our (create) form—single by value

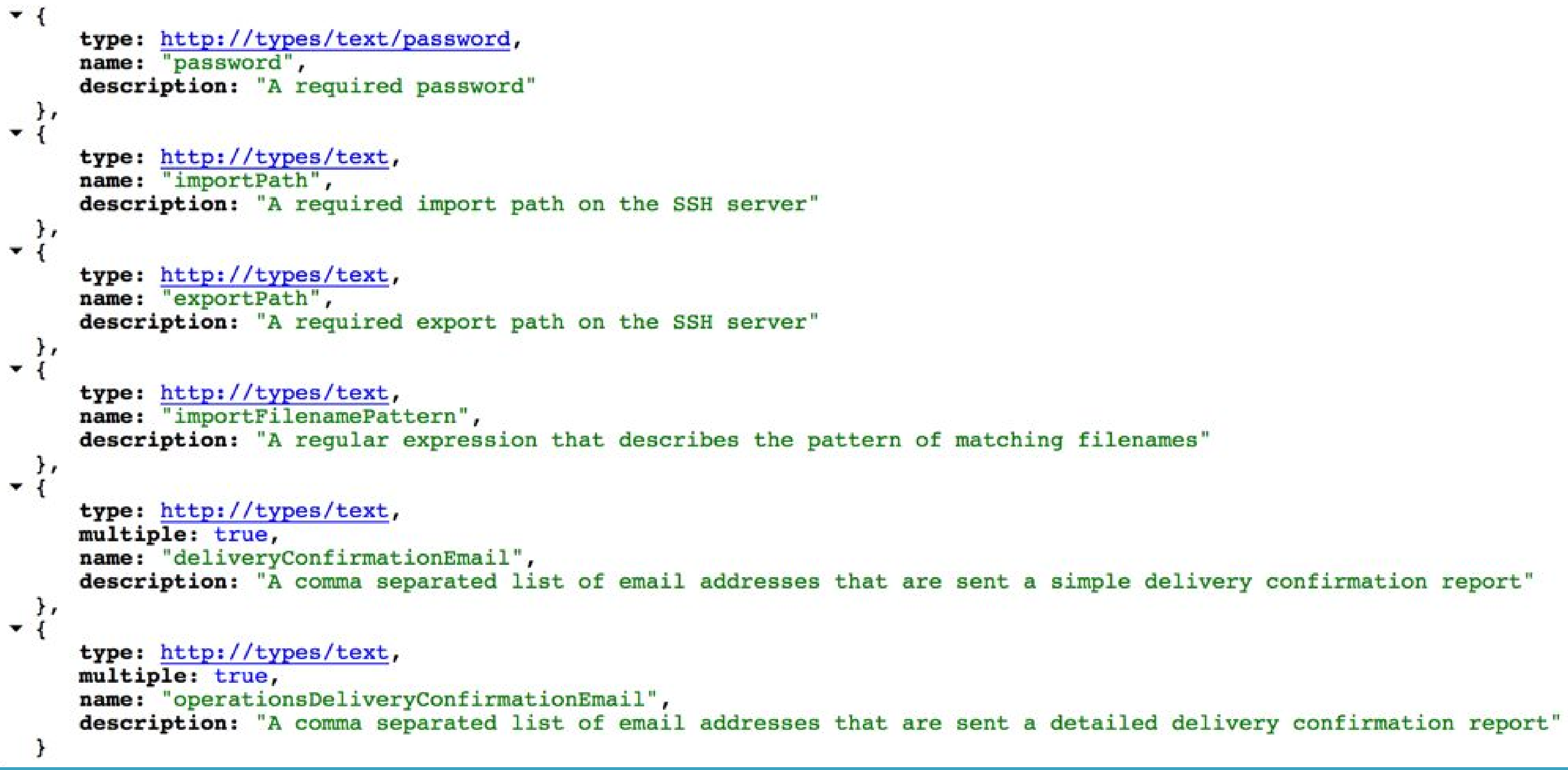


```
▾ links: [
   ▾ {
          rel: "self",
          href: https://api-cem-qa.cemplicity.com/tena
     },
   ▾ {
          rel: "up",
          href: https://api-cem-qa.cemplicity.com/
     },
   ▾ {
          rel: "search",
          href: https://api-cem-qa.cemplicity.com/tena
     },
   ▾ {
          rel: "create-form",
          href: https://api-cem-qa.cemplicity.com/tena
     }
  ],
```



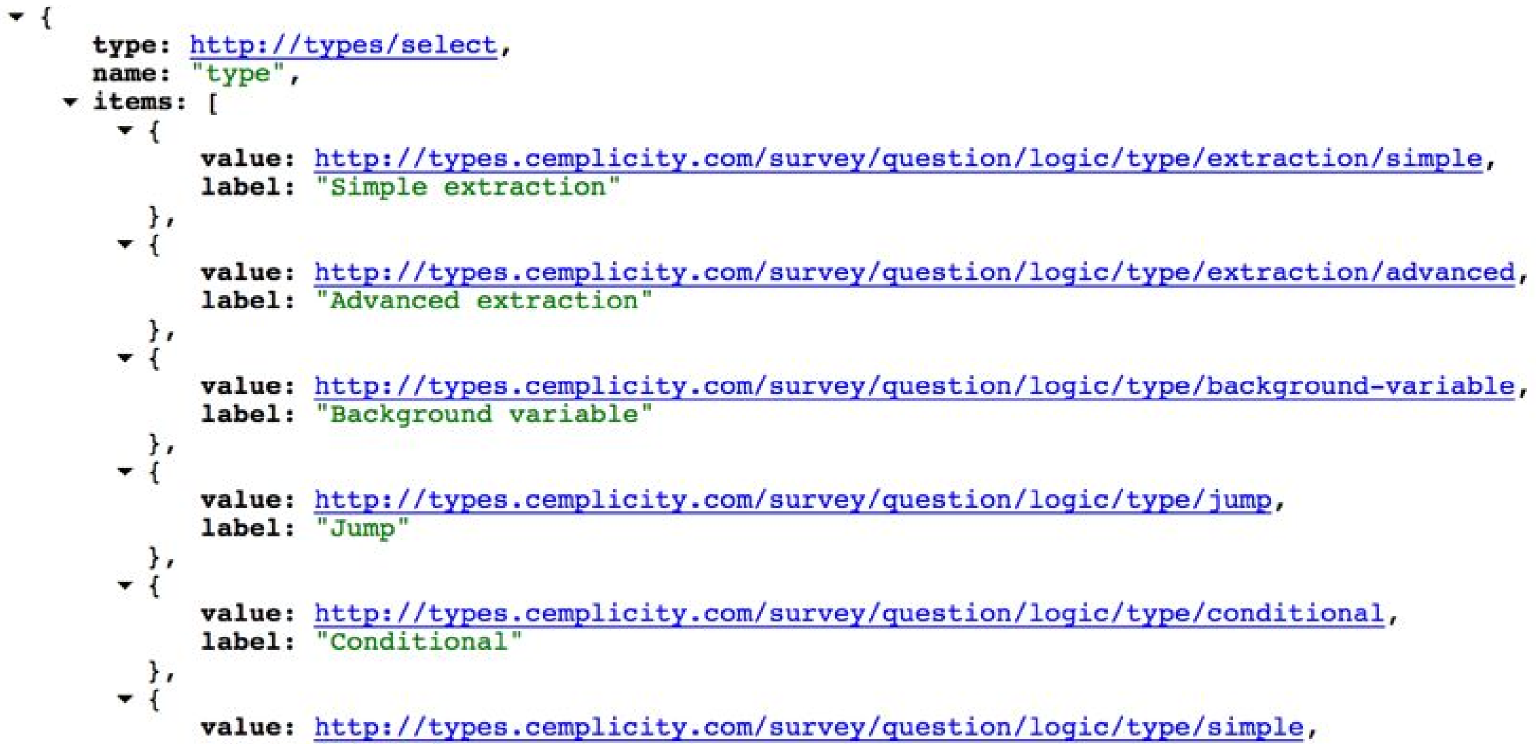
```
▾ links: [
   ▾ {
          rel: "self",
          href: https://api-cem-qa.cemplicity.com/tenant/form/create
     },
   ▾ {
          rel: "up",
          href: https://api-cem-qa.cemplicity.com/tenant/
     },
   ▾ {
          rel: "submit",
          href: https://api-cem-qa.cemplicity.com/tenant/
     }
  ],
▾ items: [
   ▾ {
          type: http://types/text,
          name: "name",
          description: "The name of the tenant"
     },
   ▾ {
          type: http://types/text,
          name: "code",
          description: "The short code used to describe the tenant"
```

… more by value



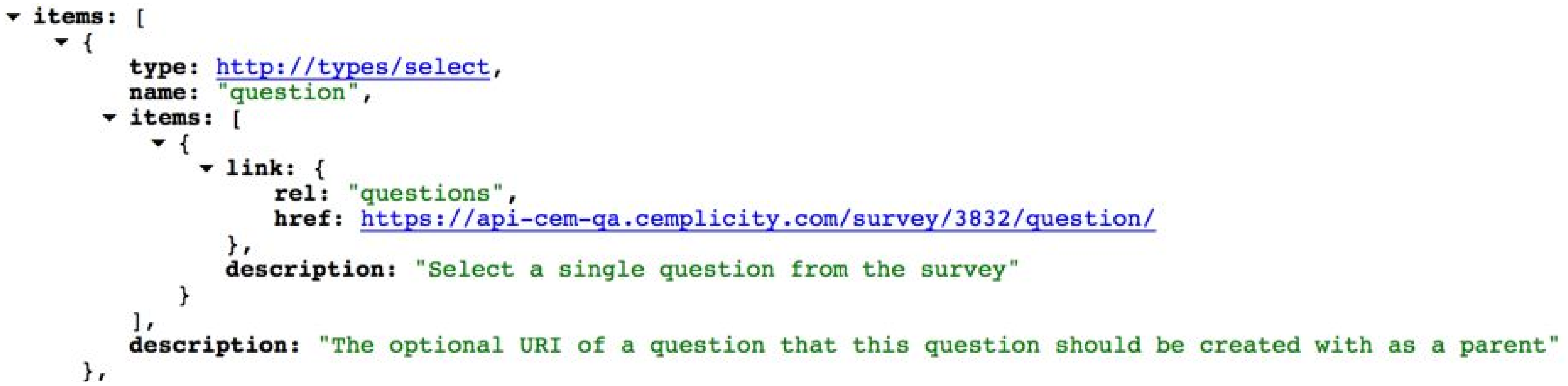
```
▼ {
      type: http://types/text/password,
      name: "password",
      description: "A required password"
  },
▼ {
      type: http://types/text,
      name: "importPath",
      description: "A required import path on the SSH server"
  },
▼ {
      type: http://types/text,
      name: "exportPath",
      description: "A required export path on the SSH server"
  },
▼ {
      type: http://types/text,
      name: "importFilenamePattern",
      description: "A regular expression that describes the pattern of matching filenames"
  },
▼ {
      type: http://types/text,
      multiple: true,
      name: "deliveryConfirmationEmail",
      description: "A comma separated list of email addresses that are sent a simple delivery confirmation report"
  },
▼ {
      type: http://types/text,
      multiple: true,
      name: "operationsDeliveryConfirmationEmail",
      description: "A comma separated list of email addresses that are sent a detailed delivery confirmation report"
  }
```

… enumeration by value



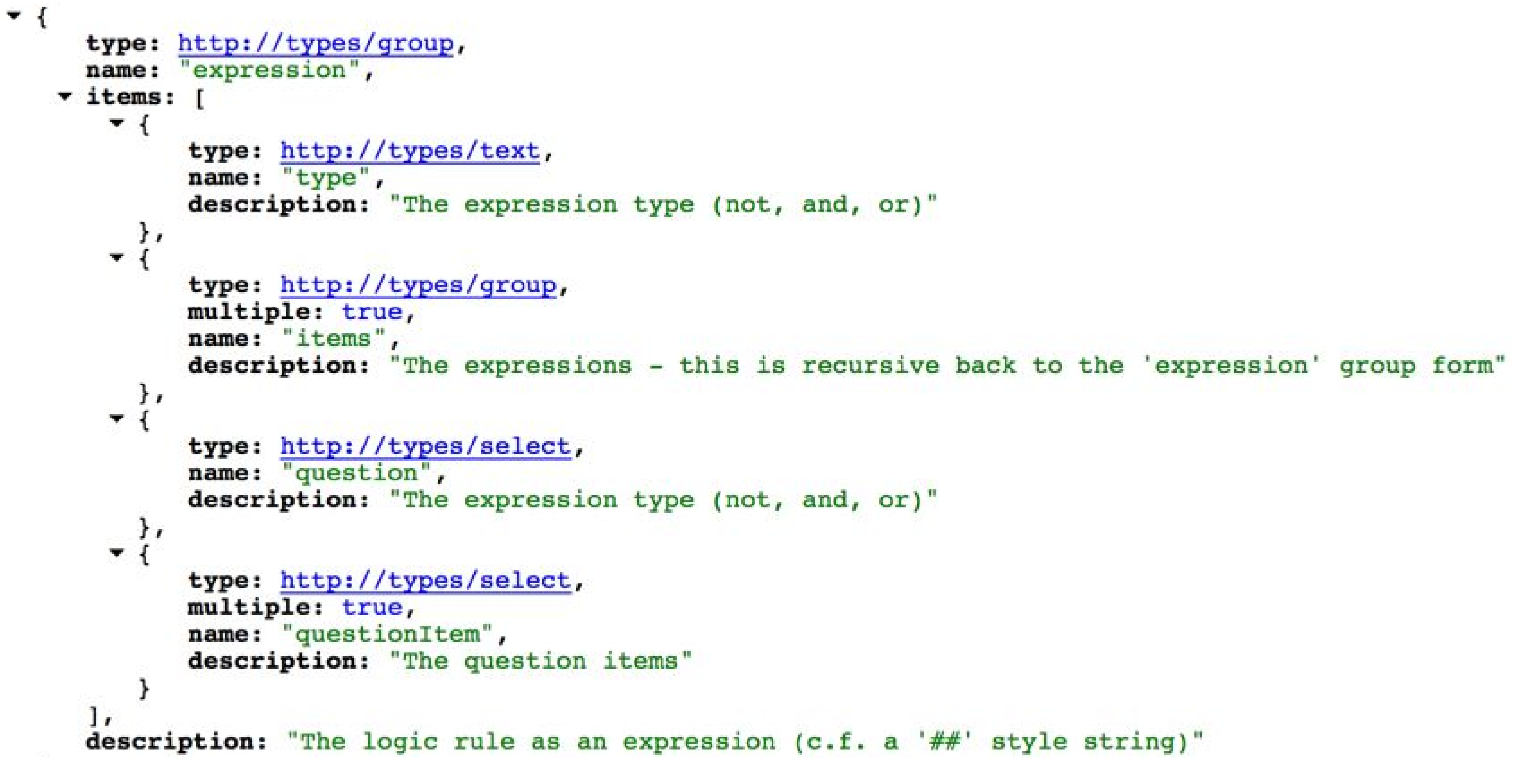
```
▾ {
      type: http://types/select,
      name: "type",
    ▾ items: [
        ▾ {
              value: http://types.cemplicity.com/survey/question/logic/type/extraction/simple,
              label: "Simple extraction"
          },
        ▾ {
              value: http://types.cemplicity.com/survey/question/logic/type/extraction/advanced,
              label: "Advanced extraction"
          },
        ▾ {
              value: http://types.cemplicity.com/survey/question/logic/type/background-variable,
              label: "Background variable"
          },
        ▾ {
              value: http://types.cemplicity.com/survey/question/logic/type/jump,
              label: "Jump"
          },
        ▾ {
              value: http://types.cemplicity.com/survey/question/logic/type/conditional,
              label: "Conditional"
          },
        ▾ {
              value: http://types.cemplicity.com/survey/question/logic/type/simple,
```

… enumeration by reference (spot the problems)



```
▾ items: [
    ▾ {
          type: http://types/select,
          name: "question",
        ▾ items: [
            ▾ {
                ▾ link: {
                      rel: "questions",
                      href: https://api-cem-qa.cemplicity.com/survey/3832/question/
                  },
                  description: "Select a single question from the survey"
              }
          ],
          description: "The optional URI of a question that this question should be created with as a parent"
      },
```

… group

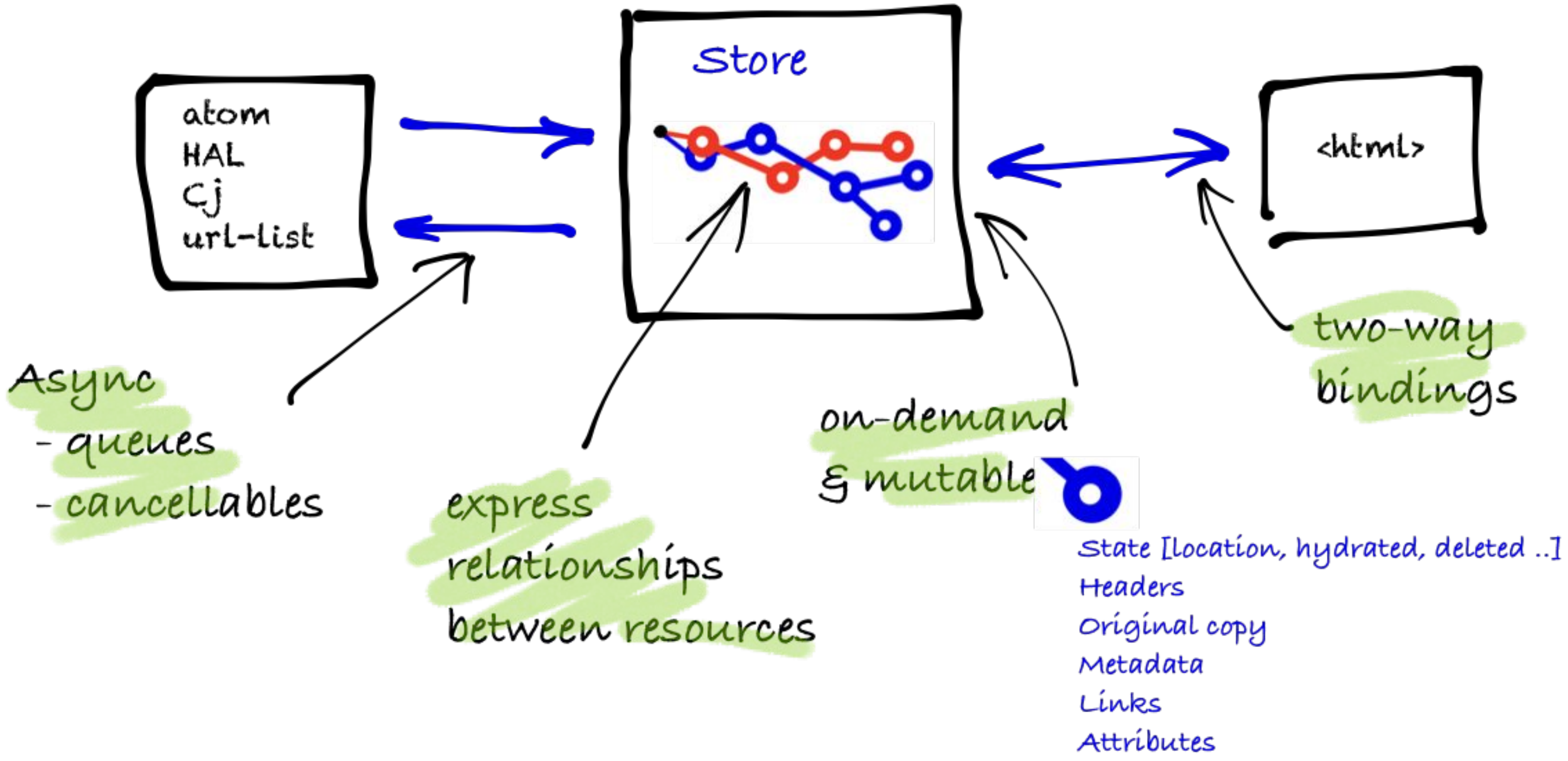


```
▾ {
      type: http://types/group,
      name: "expression",
  ▾   items: [
      ▾ {
            type: http://types/text,
            name: "type",
            description: "The expression type (not, and, or)"
        },
      ▾ {
            type: http://types/group,
            multiple: true,
            name: "items",
            description: "The expressions - this is recursive back to the 'expression' group form"
        },
      ▾ {
            type: http://types/select,
            name: "question",
            description: "The expression type (not, and, or)"
        },
      ▾ {
            type: http://types/select,
            multiple: true,
            name: "questionItem",
            description: "The question items"
        }
      ],
      description: "The logic rule as an expression (c.f. a '##' style string)"
```

Five design issues

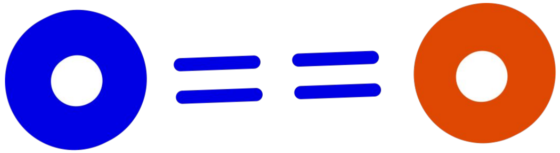
- Client store
- Identity
- Hydration
- Mappings
- Caching

1. Single In-memory client-side resource store



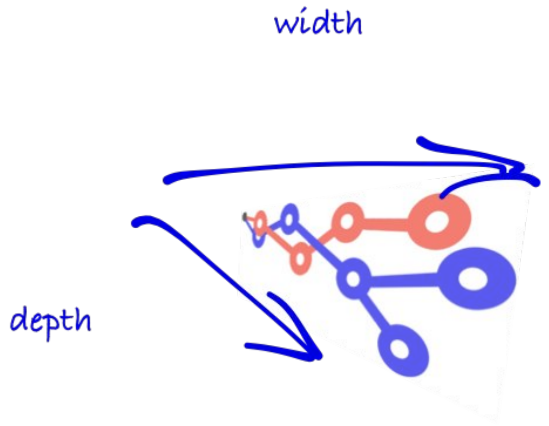
2. Identity of resources

- you need to be able to choose some exactly from link relations (eg 'self') and sometimes more loosely from an attribute (eg 'name') and sometime with a mixture (eg other a combination of link relations)
- need collection utilities to aid and abet this—you'll need to map identity on collection (references) as well as values



3. Hydration strategies: width-first and depth-first

- walking widely vs deeply (eg create a parent collection of items before going deeply into each item)
- we've needed collection aware async map and reduce utils (in both parallel and sequential



4. Remember mappings for making copies of trees

- taking a copy or part of a known tree or a disconnected tree (and grafting into the graph)
- simple string comparisons (because they are URIs)—implementation is dictionary
- as you are walking the tree, you'll need to be able to do substitutions
- can't have forward references (so we need to know the order to avoid recursive-lazy-loading problems)
- early loading forward references (eg metadata because it will be used by others to create themselves, ancestors before descendants)

5. Pushing through the cache

- avoid thinking about best shot at not having stale representations (force loads flags)
- hold cache headers in the in-memory state and use them to decided—so the server decides
- remember you'll still need to get through the browser, http, reverse-client proxy, persistence caches though!

Five design issues

We’ve found useful to know about and then incrementally add parts of each as you need them

- Client store
- Identity
- Hydration
- Mappings
- Caching

Conclusion

- This is about being affordance-centric, use forms to instruct the client
- We haven't talked about affordance-centric based workflow as a GUI
- We aren't getting this for free yet (ie examples and tooling support), so you'll need to think about your app complexity when building out clients
- Stay simple and explainable

Thanks!

Todd Brackley @toddb*NZ*
Hypr @hypr*NZ*