Challenges in Evolving APIs

Pragmatic Steps to Overcoming Challenges

NATIONAL FOOTBALL LEAGUE

Outline

- Introduction
- Principles
- Practice
- Plugs
- Personal Challenges (Q & A)

Arash Shokoufandeh

- 30+ years experience
- Valkyrie Team Lead
- Primarily responsible for **api.nfl.com**

Earl Nolan

- 30 years experience
- Meta Team Lead
- Lifetime Member National Ski Patrol
- **8.21.2017**

Outline

- Introduction
- Principles
- Practice
- Plugs
- Personal Challenges (Q & A)

Principles

- API Manifesto
- GraphQL

API MANIFESTO

- There are two hard things in computer science: cache invalidation, naming things, and off-by-one errors
  - Variation on Phil Karlton: http://martinfowler.com/bliki/TwoHardThings.html
- Functionality should be easy to explain. If it is hard to name, that's generally a bad sign
- Good names drive good design. Prefer specific names to general names. Names occupy API real estate.

API MANIFESTO

- An API should do one thing and do it well
- An API should force the client to do the right thing
- Complete auto-discovery by the API user
- Avoid boolean parameters. Use enumerated types instead to allow for future possibilities
- Power to Weight Ratio: when in doubt, leave it out YAGNI
- Implementation should not impact the API

GraphQL

- Data Query Language released by Facebook in 2015
- Supported by multiple languages
- Strongly typed
- Client Specifies exactly what they want (projection and joins)
- Graphql.org has a really nice summary

GraphQL

- NOT REST
- Not specific to HTTP, can use any RPC mechanism
- Different API for reads vs. mutations
- Discoverable via introspection query
- Graphiql is a killer app on the Mac
- Queries are not JSON but responses are

GraphQL Reads

- Field selection (projection)
  - Reduces payload (important for mobile)
  - Provides exactness of what is in use (safely drop unused fields)
  - No loops
- Arguments can be provided at any level
- Datafetcher(s) may be specified at the field level

GraphQL Read Examples

```
query {
  viewer {
    schemas(first:1, after:"c2ltcGxlLWN1cnNvci0x") {
      edges {
        node {
          name
        }
      }
    }
  }
}
```

GraphQL Read Response


```
{
  "data": {
    "viewer": {
      "schemas": {
        "edges": [
          {
            "node": {
              "name": "Bozo The Clown"
            }
          },
          {
            "node": {
              "name": "Curious George"
            }
          }
        ]
      }
    }
  }
}
```

No Loops Example

```
query {
  viewer {
    instances {
      edges {
        node {
          oneness,
          goDeeper {
            oneness,
            goDeeper {
              oneness,
              goDeeper {
                oneness,
                goDeeper {
                  oneness,
                  goDeeper {
                    oneness,
                    goDeeper {
                      oneness,
                      goDeeper {
                        oneness
                      }
                    }
}}}}}}}}}}
```

GraphQL Mutations

- API developer specifies what can be changed
- Granularity can be one field to anything
- Arguments can be provided at any level
- Asymmetry between mutation request and response

GraphQL Mutation Example

```
mutation  {
  upsertSchemaDefinition(schemaDef: {
    name: "skibob",
    idGeneration: Client,
    domainFields: [{
        memberType:       String,
        memberFieldName: "resort",
        memberDescription: "Where to ski."
    }]
  })
  {
    name
  }
}
```

Outline

- Introduction
- Principles
- Practice
- Personal Challenges (Q & A)

Practice

- Discoverability
- API Additions
- API Deprecation/Deletion
- API Modification

HAL

- Makes API Explorable/Discoverable
- Expresses API Hyperlinks + plain old JSON
- Default representation for Spring Boot/Spring Data
- Read more about it here:
  - http://stateless.co/hal_specification.html
- Other option: SIREN
- HAL is good for read only applications. SIREN is better for PUT/POST/… definitions

EXAMPLE: TOP LEVEL


```
{
  "_links":{
    "video":{
      "href":"http://prod.ais.clubs.nfl.com/video{?page,size,
sort}",
      "templated":true
    },
    "profile":{
      "href":"http://prod.ais.clubs.nfl.com/alps"
    }
  }
}
```

EXAMPLE: SEARCH


```
{
  "_links":{
    "findByLast7days":{
      "href":"http://prod.ais.clubs.nfl.com/video/search/findByLast7days{?clubCode}",
      "templated":true
    },
    "findByNameAndClubCode":{
      "href":"http://prod.ais.clubs.nfl.com/video/search/findByNameAndClubCode{?name,clubCode}",
      "templated":true
    }
  }
}
```

GraphQL Discoverability

- Introspection Query
  - Discover all types
  - How to view
  - How to mutate

Practice

- Discoverability
- API Additions
- API Deprecation/Deletion
- API Modification

Outline

- Introduction
- Principles
- Practice
- Plugs
- Personal Challenges (Q & A)

NFL Open source

- GOLD: Dynamic Domain driven by GraphQL
- GOLD Starter: Try GOLD quickly
- GraphQL Mediator: Convert introspection query into usable objects
- GLiTR: POJOs to GraphQL schema made easy
- Audible: POJO Mapper using Java 8 Lambda and Orika

- For all our open source projects, see:
  - https://github.com/NFL

Outline

- Introduction
- Principles
- Practice
- Plugs
- Personal Challenges (Q & A)

Questions???

- Arash Shokoufandeh: arash.shokoufandeh@nfl.com
- Earl Nolan: earl.nolan@nfl.com
- https://github.com/nfl/
- http://graphql.org/