THE LORD OF API DESIGNS

BY

ARNAUD LAURET

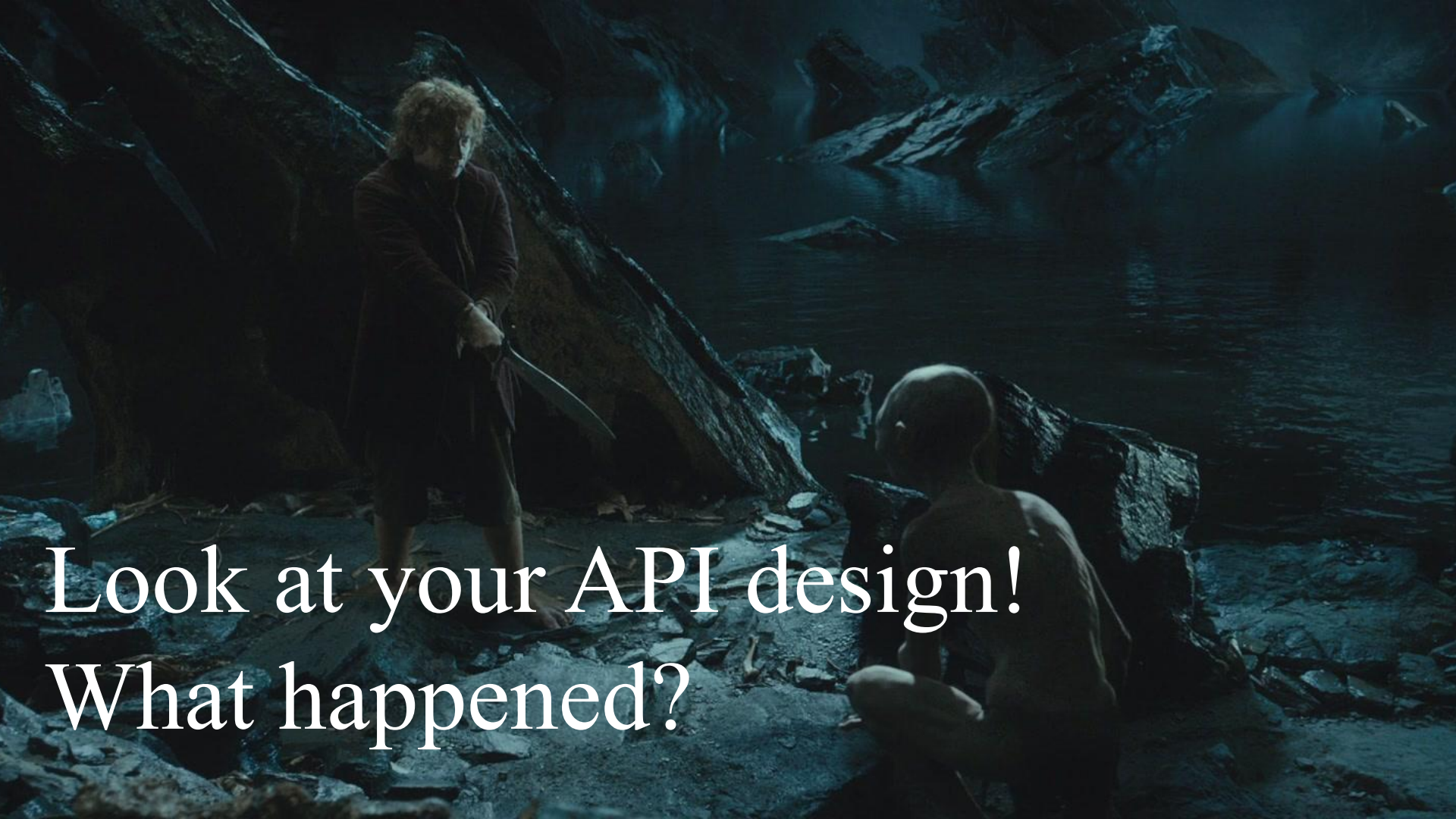
@APIHANDYMAN

APIHANDYMAN.IO

APISTYLEBOOK.COM

Chapter I

THE SHADOW OF API DESIGN


Look at your API design!
What happened?

My precious
API design!
What have we
done?

First API

But after that
the nightmare
started

First API
updated

More and
more public
and private
APIs!

Second API

Third API ...

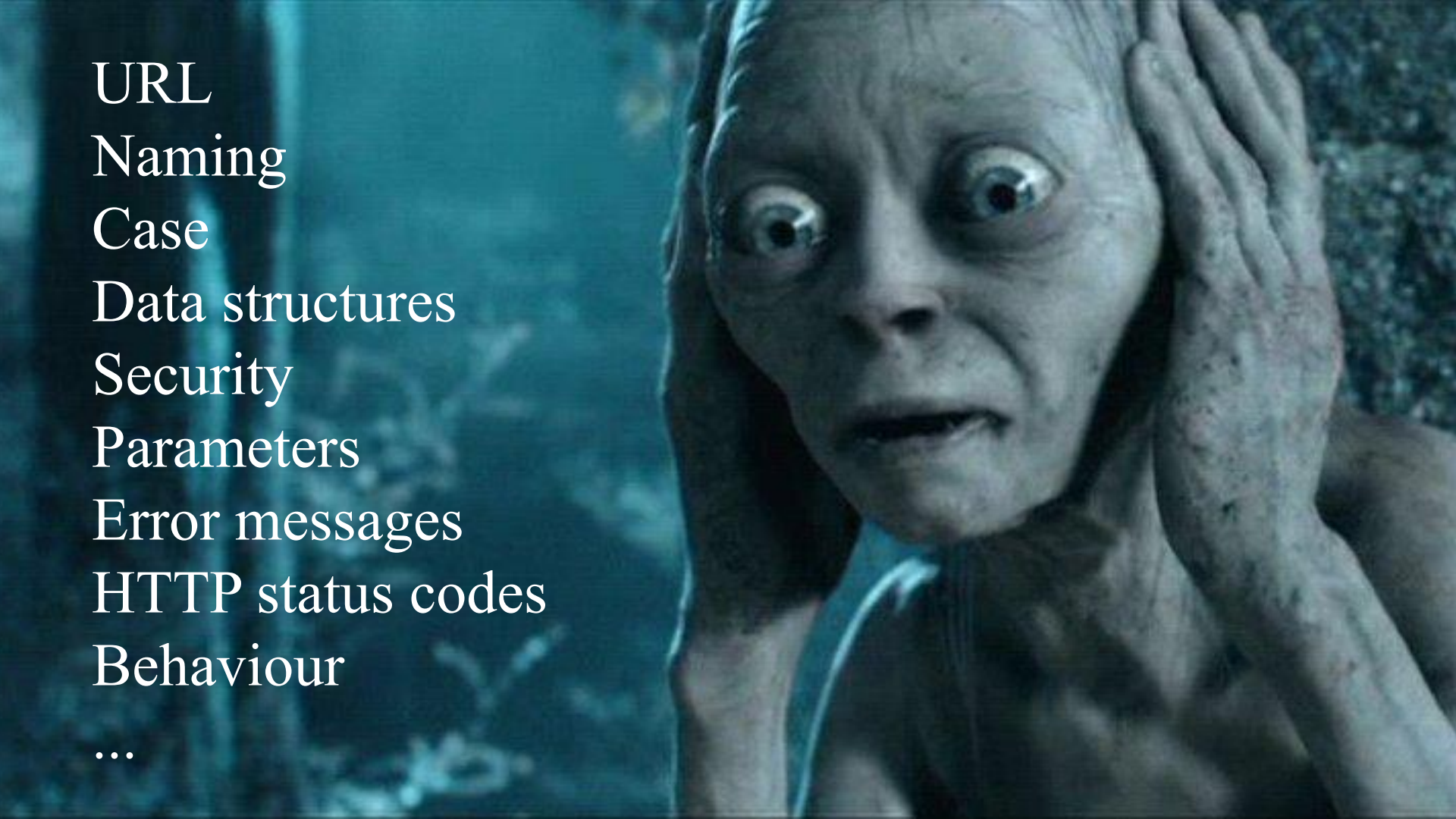
Fourth API ...

… Nth API

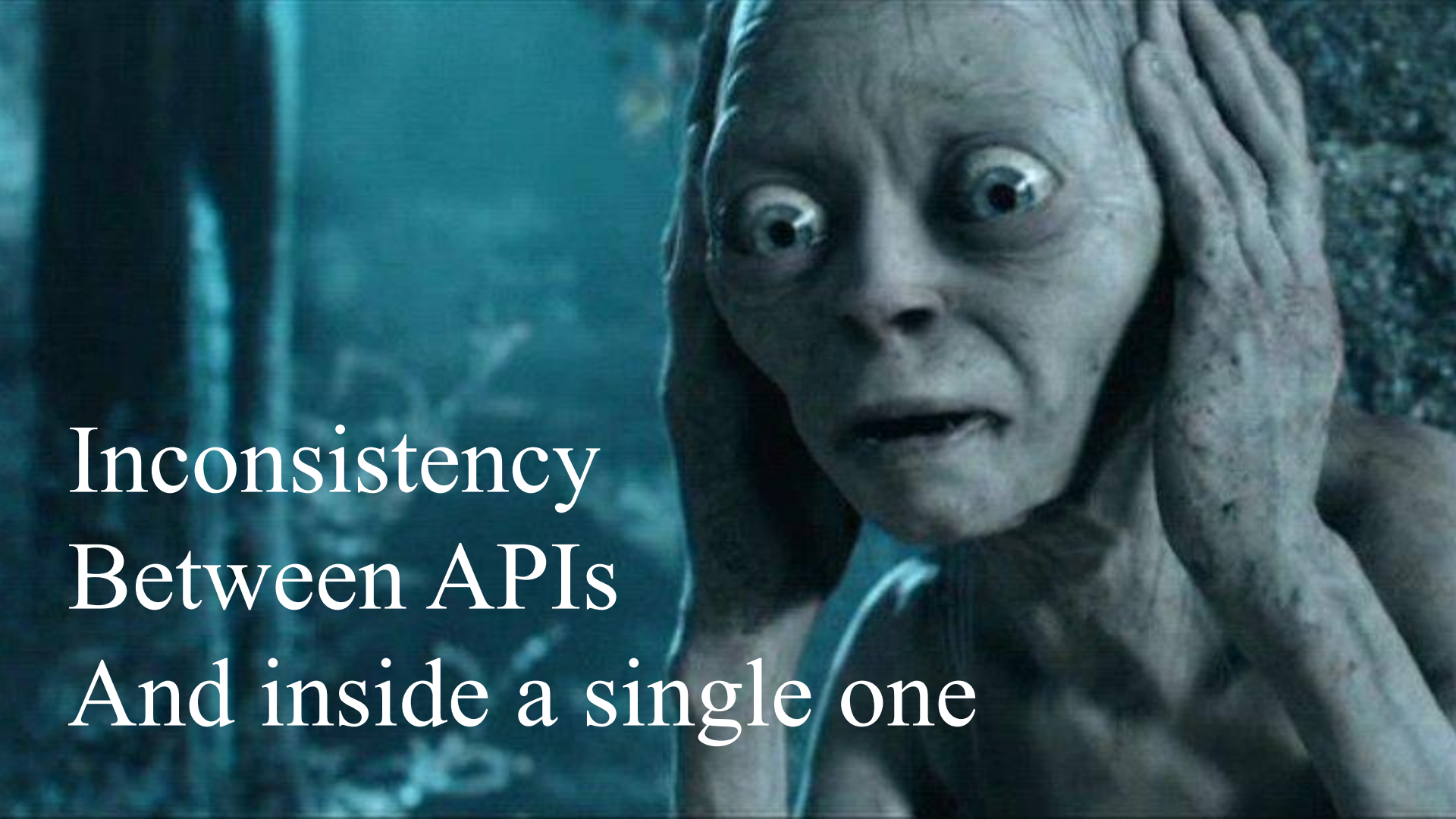
What’s the
problem with
these APIs?

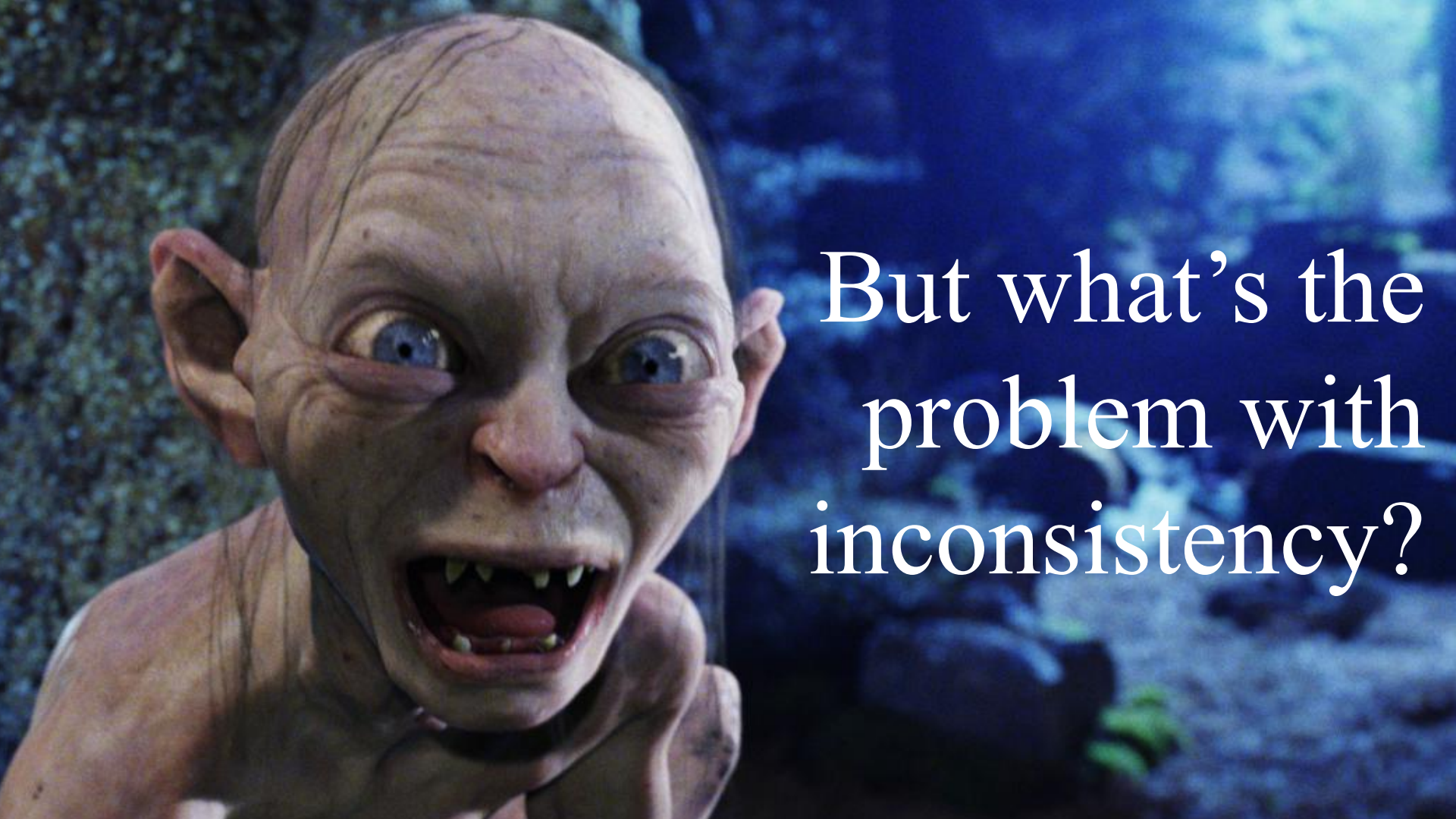
/heroes/{bilbo}
/characters/heroes/{bilbo}
/api/hero/{bilbo}

“name”: “Baggins, Bilbo”
“FirstnameLastname”: “Bilbo Baggins”
“character-name”: “bilbo baggins”
“character_name”: “BILBO BAGGINS”
“fnLn”: “B. Baggins”


URL
Naming
Case
Data structures
Security
Parameters
Error messages
HTTP status codes
Behaviour
...

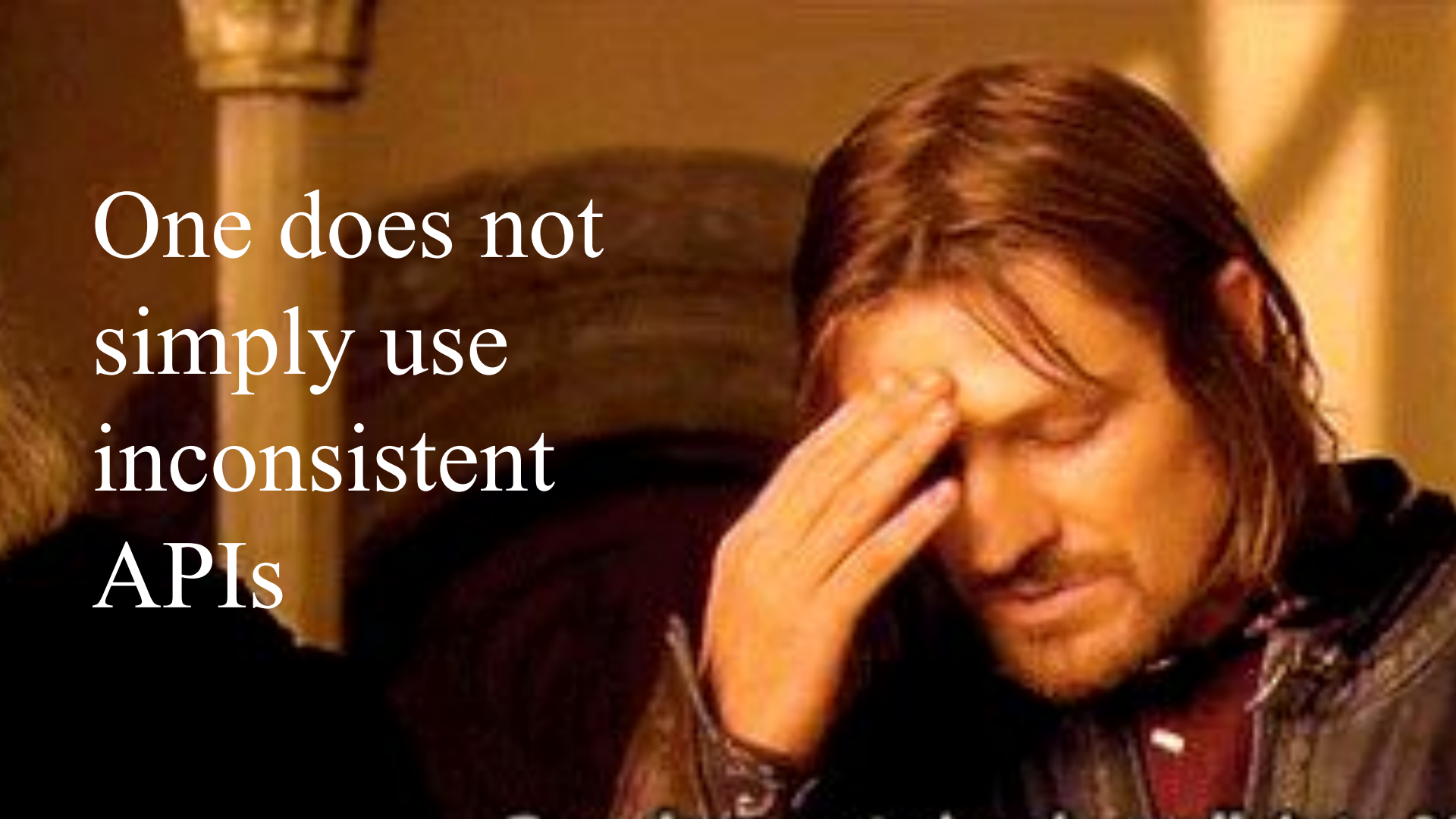
My precious!
Too much differences!
Too much inconsistency!


Inconsistency
Between APIs
And inside a single one

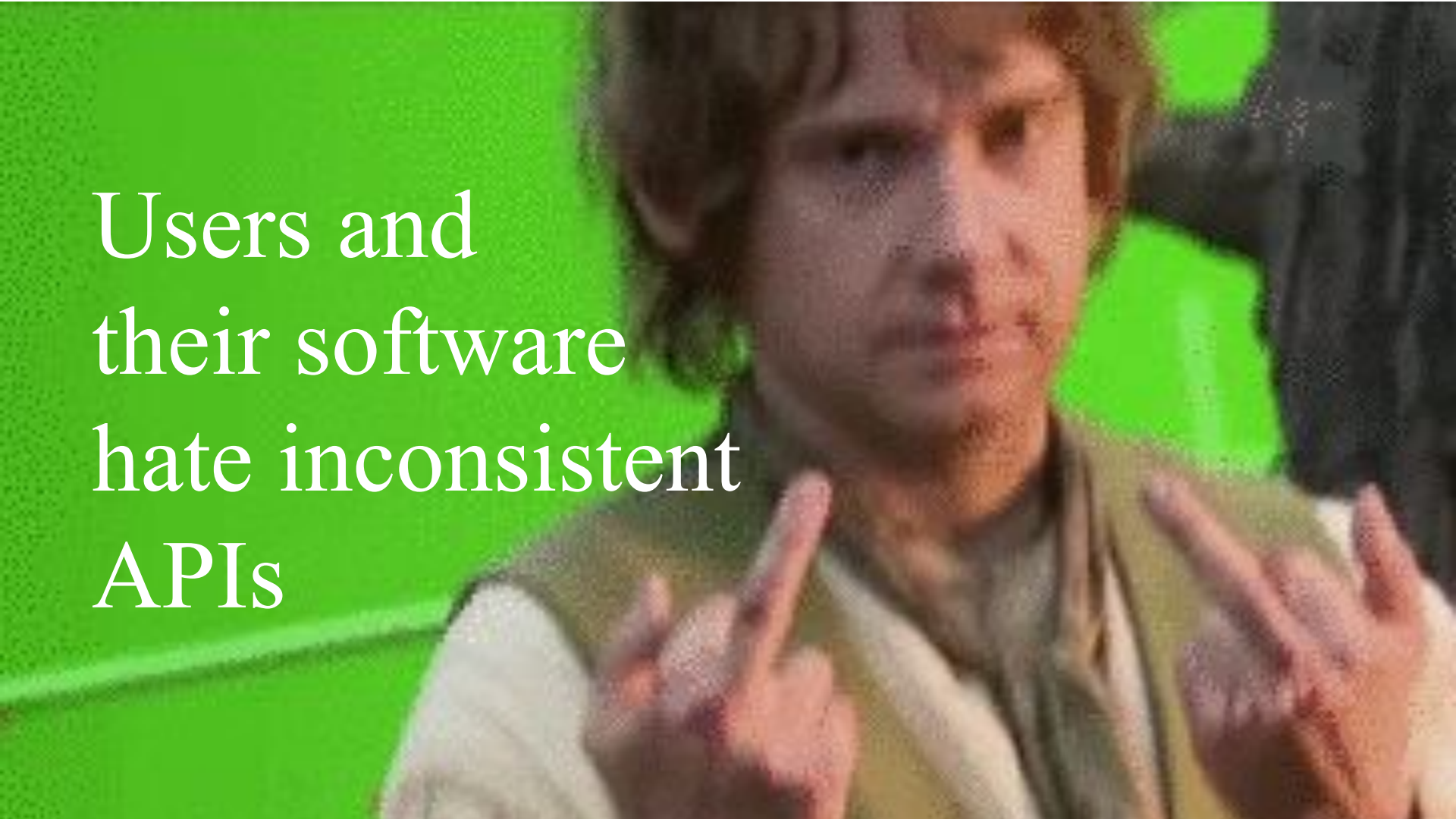

But what's the problem with inconsistency?

Why do you almost always know how to open a door?

Why Gandalf hates
Moria’s door?


One does not
simply use
inconsistent
APIs

Using each new endpoint or
API is a totally new adventure


Users and
their software
hate inconsistent
APIs

So much
time lost!
So much
users lost!
So much
money lost!

Is that the work of a single API designer?

At the
beginning
yes…

Let’s
forget what I’ve
just doooone!!!!

… But we needed more and more API designers and things get even worse!

Endless discussions

Repeated endless discussions

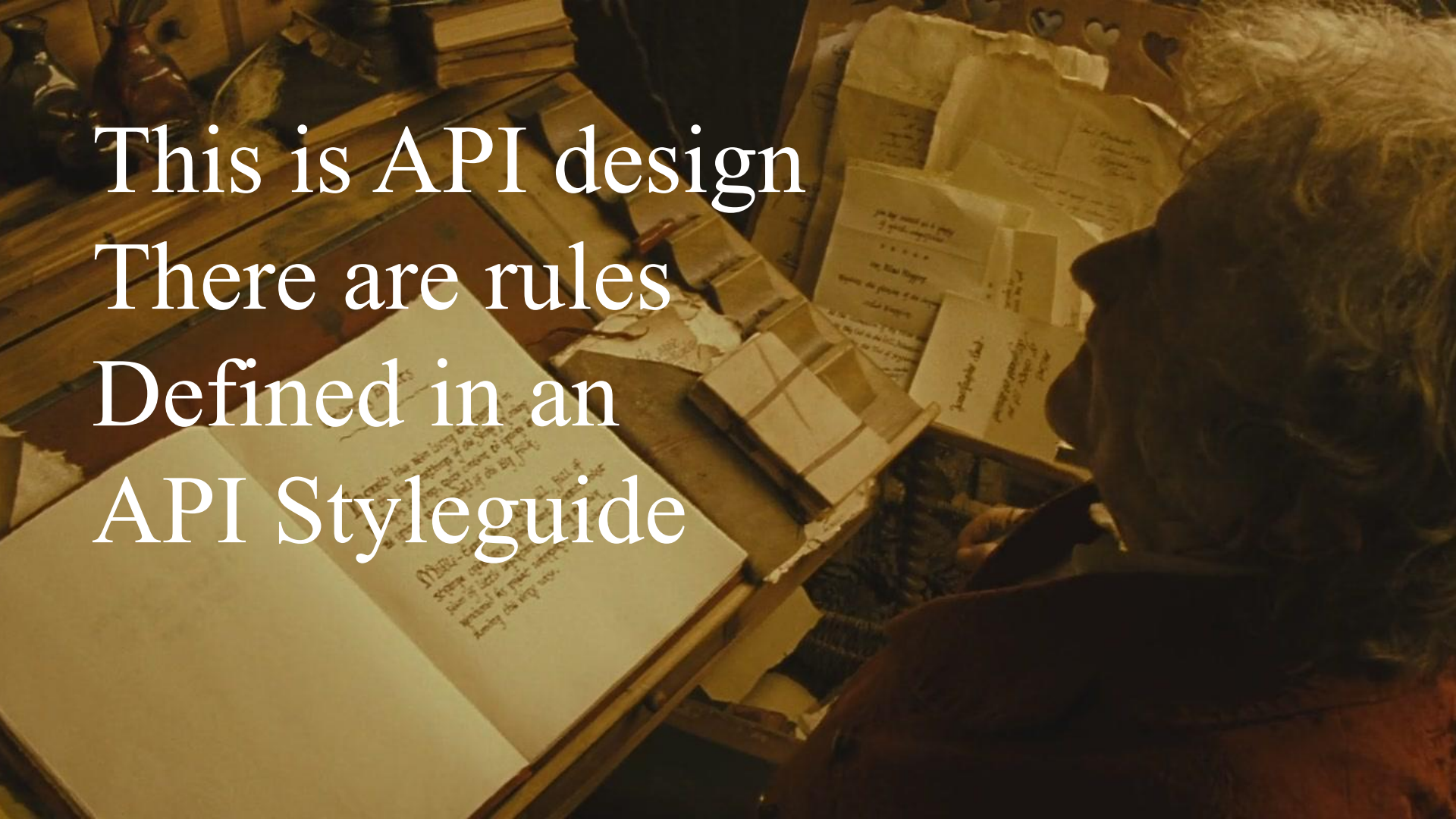
Let’s
reinvent
the wheeeeeeel!

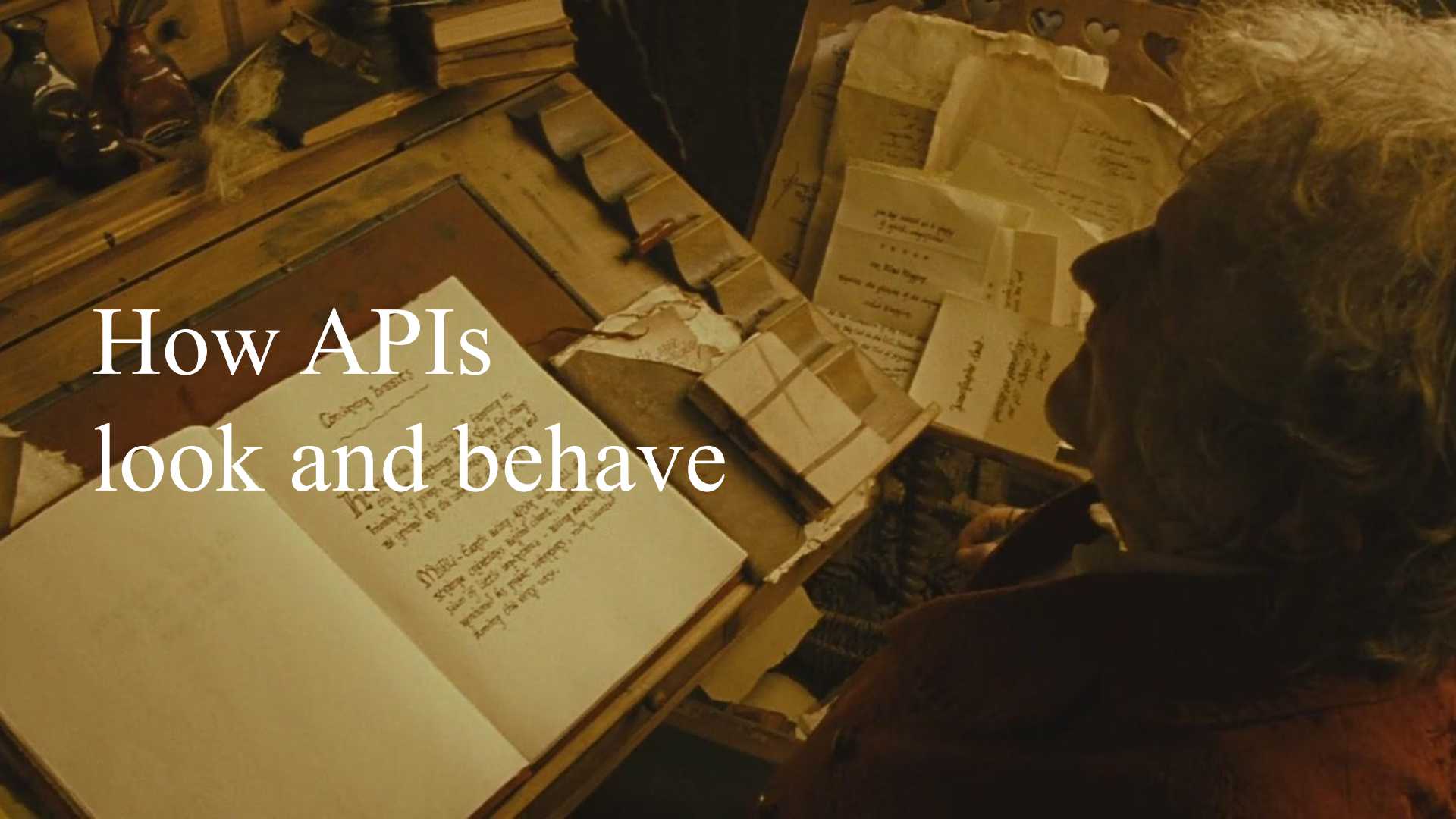
My precious!
More inconsistency!
And so much time loosed!

But what can we do my precious?

Chapter II

A LONG-EXPECTED API STYLEGUIDE


This is API design
There are rules
Defined in an
API Styleguide



How APIs look and behave

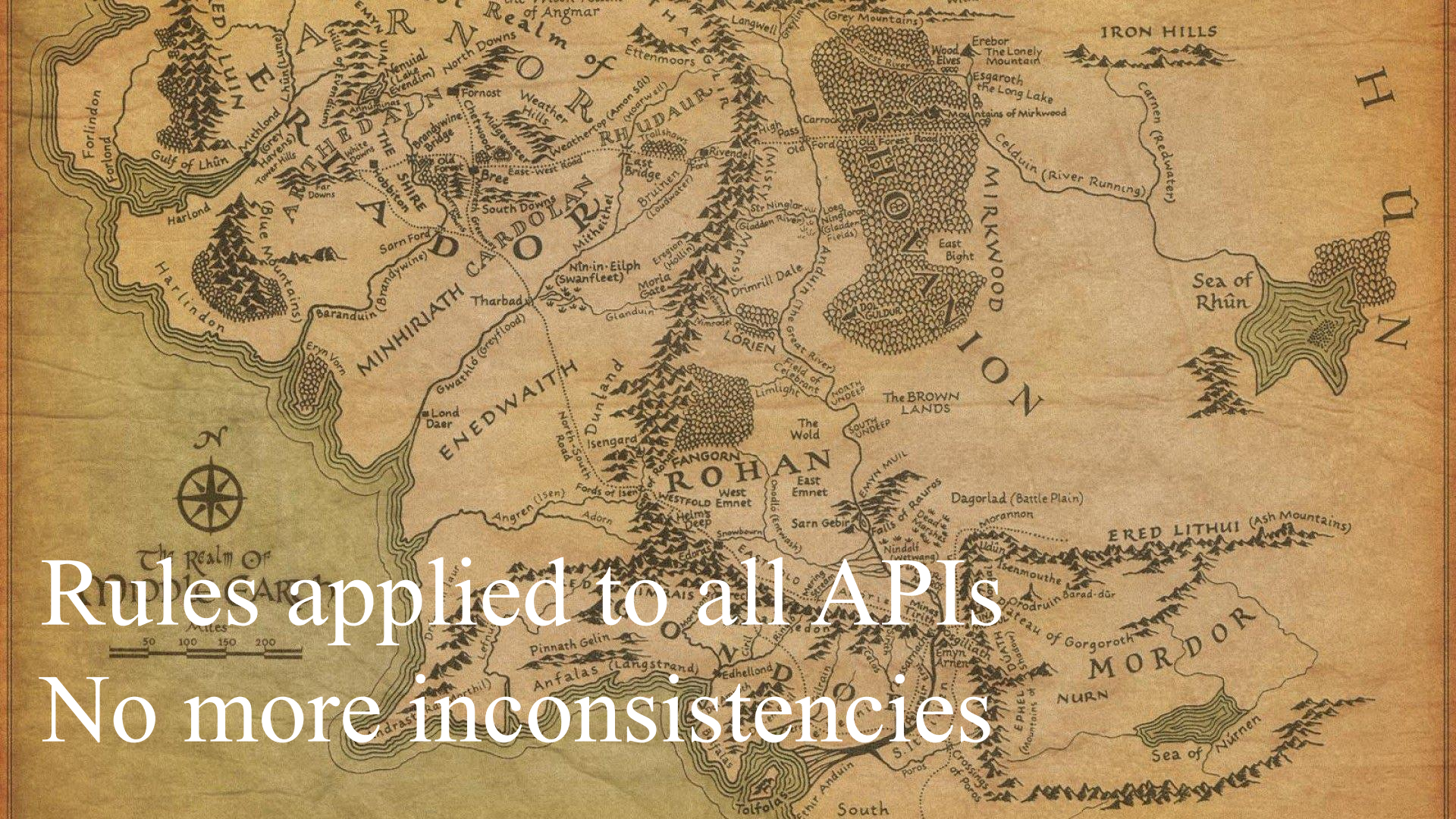
How to design APIs

How to create but also update an API

How to handle the API lifecycle

All needed rules
to ensure consistency

Rules discussed once
No more time loosed
Focus on what’s really important

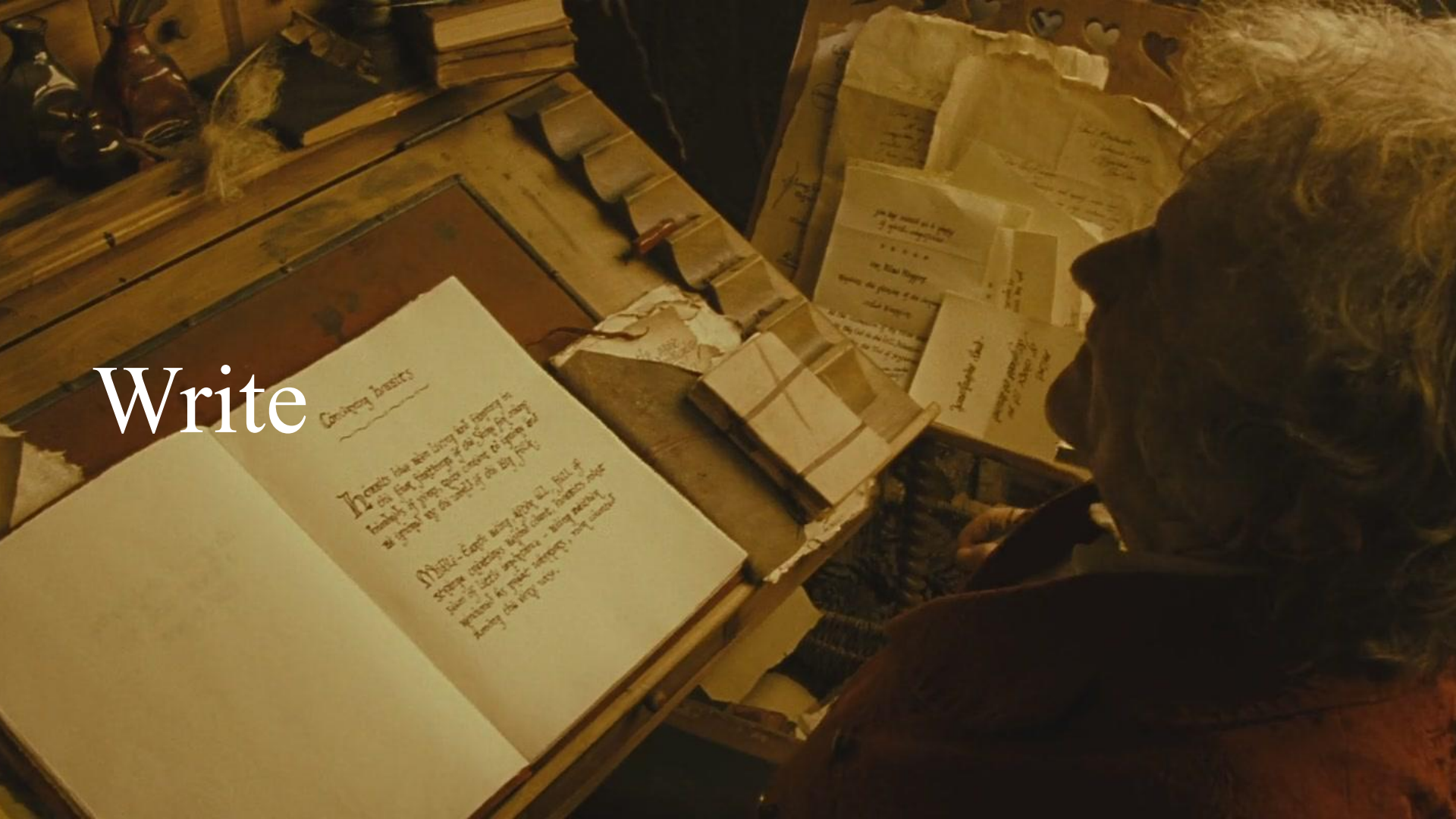

Rules applied to all APIs
No more inconsistencies

Both users and designers
are happy

So all we
have to do is
describe rules
in a styleguide?

Chapter III

THE BATTLE OF THE API STYLEGUIDE



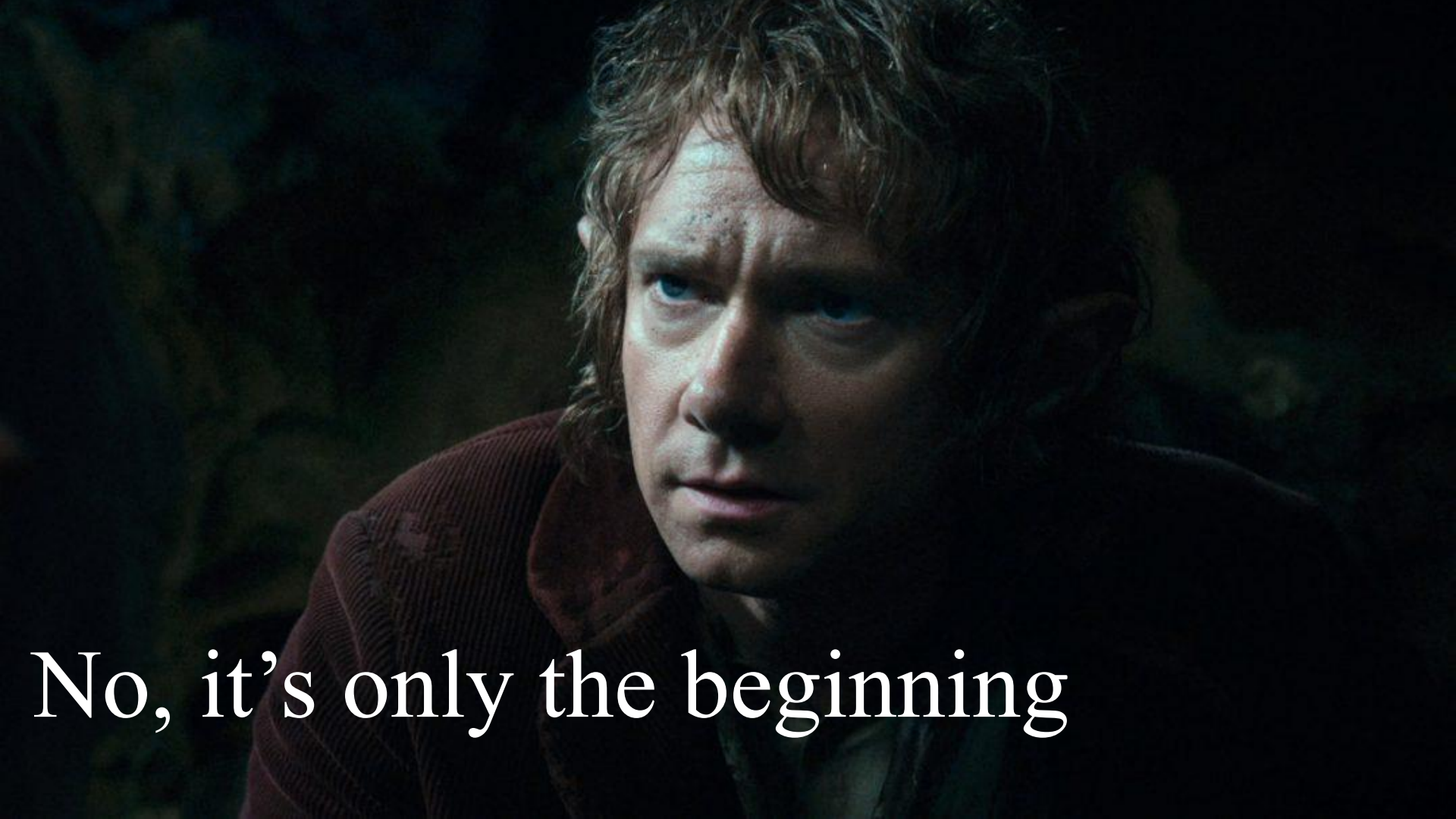
Write

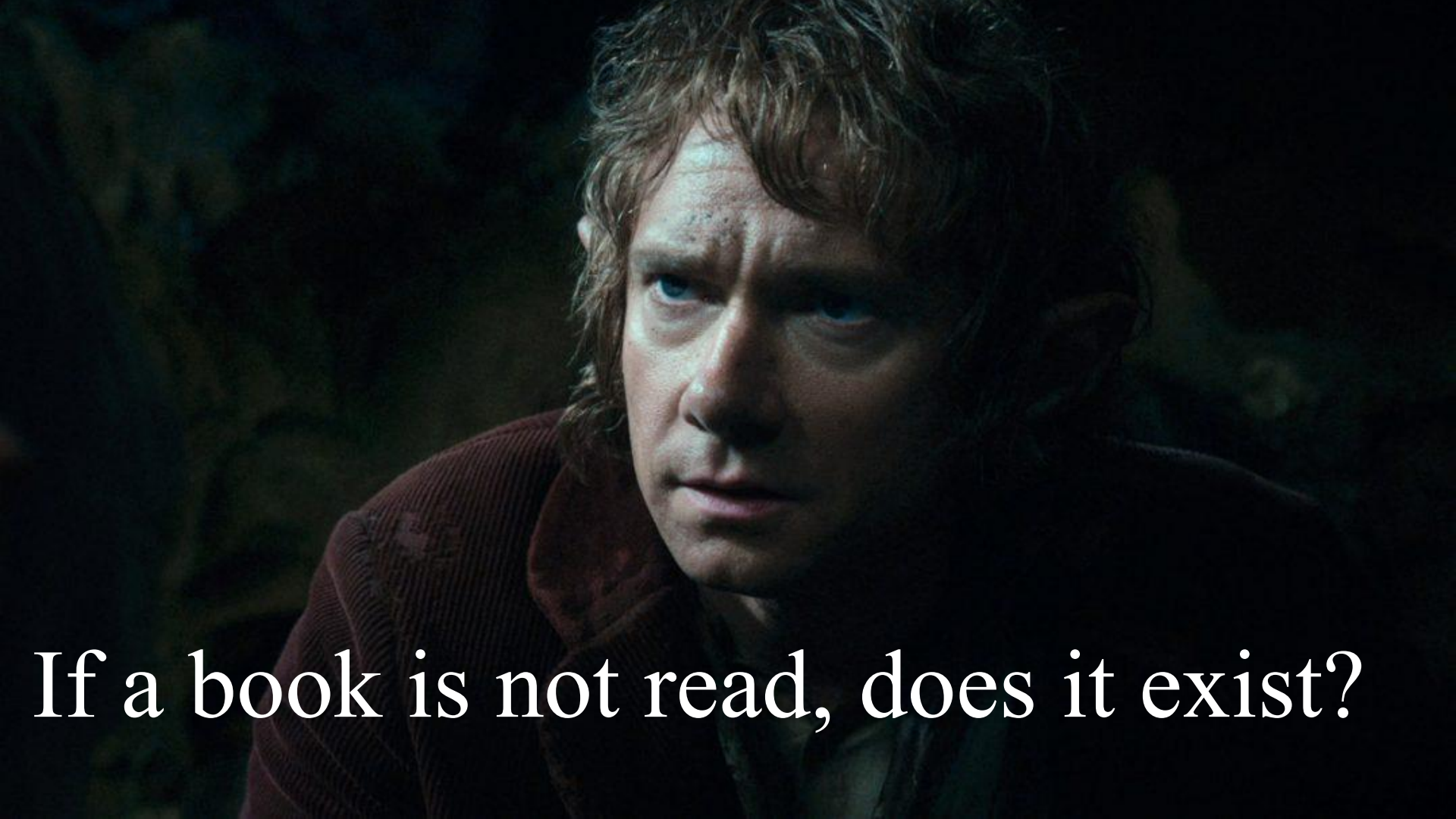
Rules based on
your existing APIs
or others

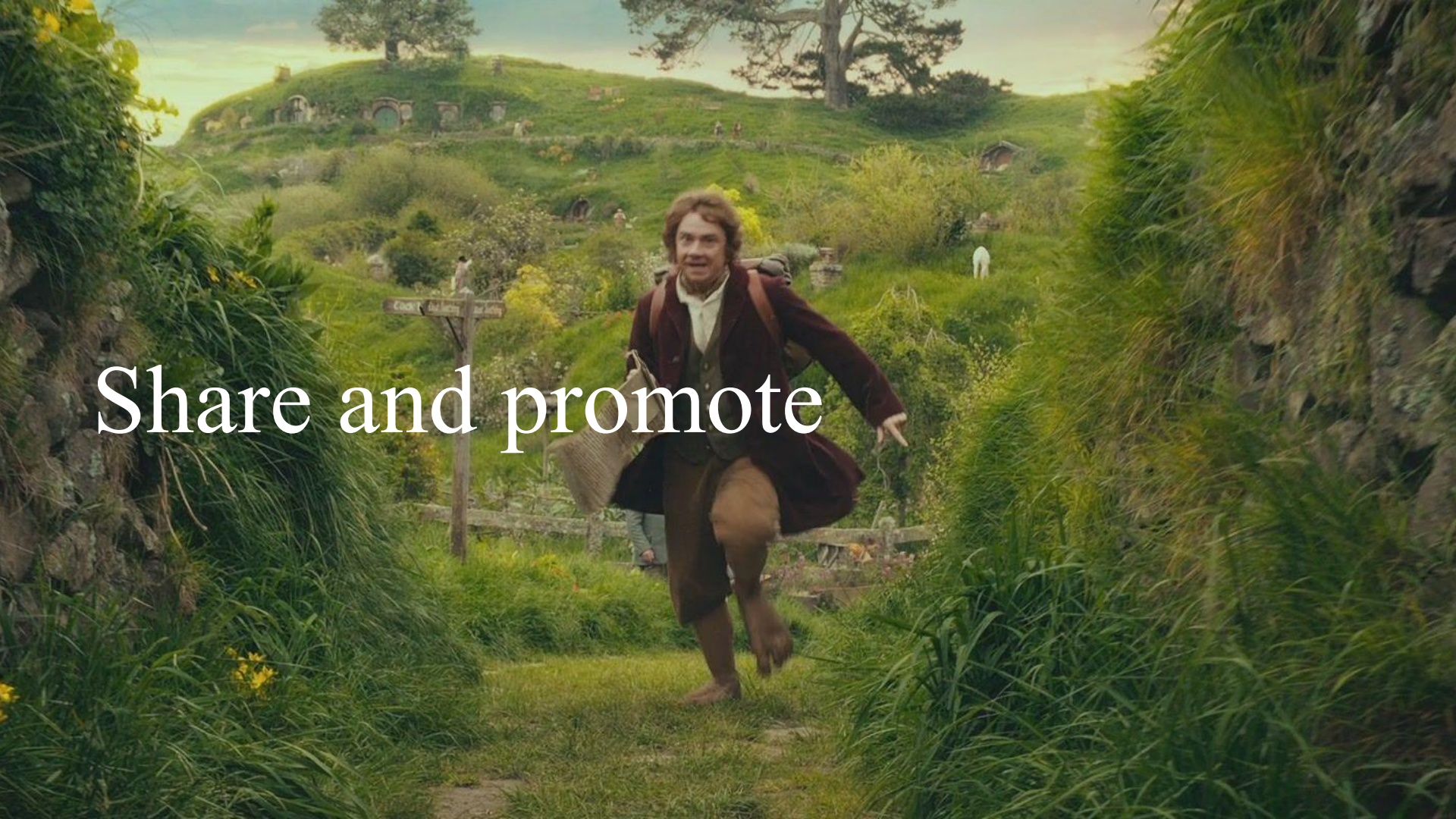
Based on existing
API styleguides
apistylebook.com
(shameless plug)

And don’t forget your
fellow API designers

And it’s done?

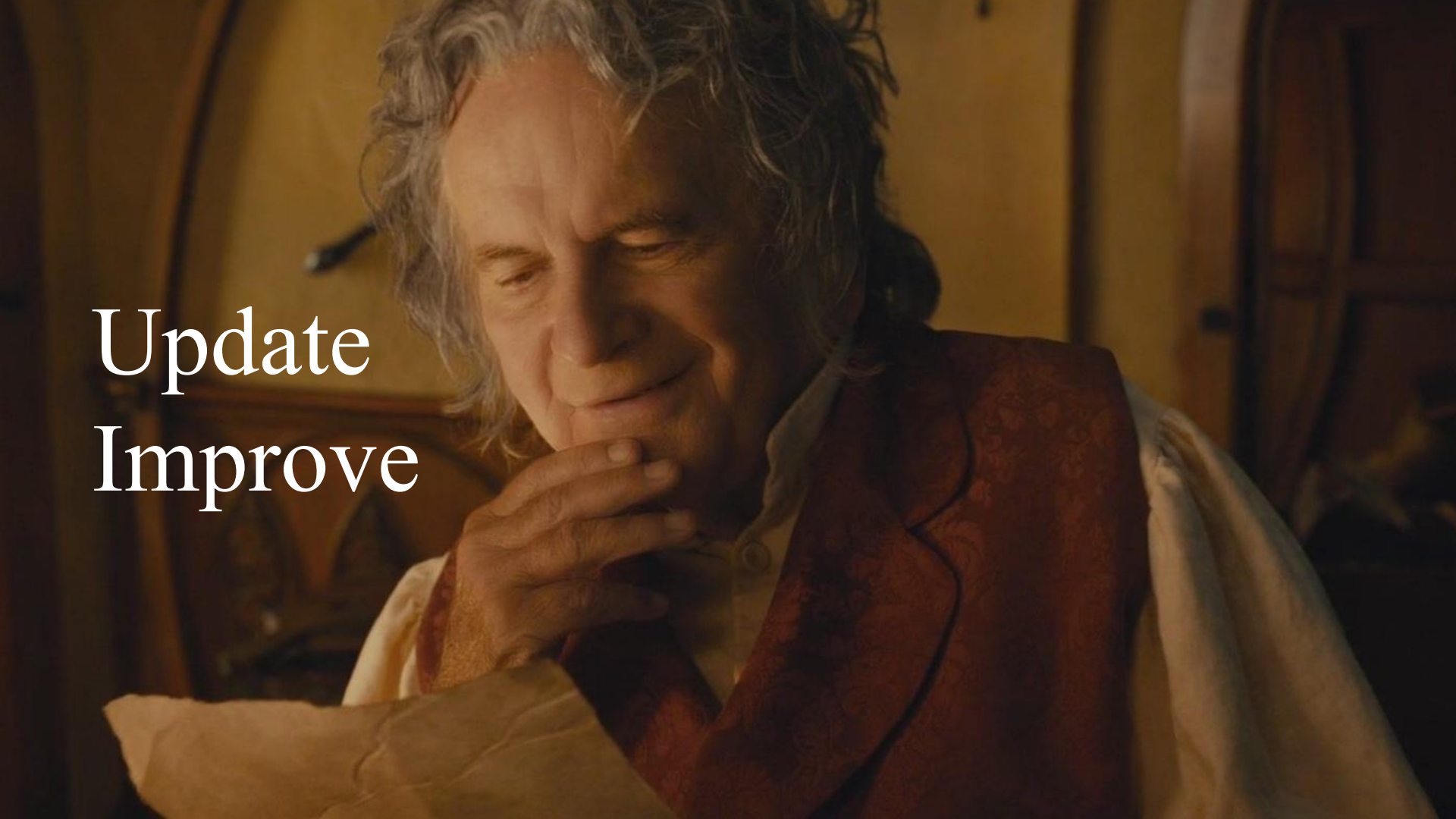

No, it’s only the beginning


If a book is not read, does it exist?


Share and promote

What else?

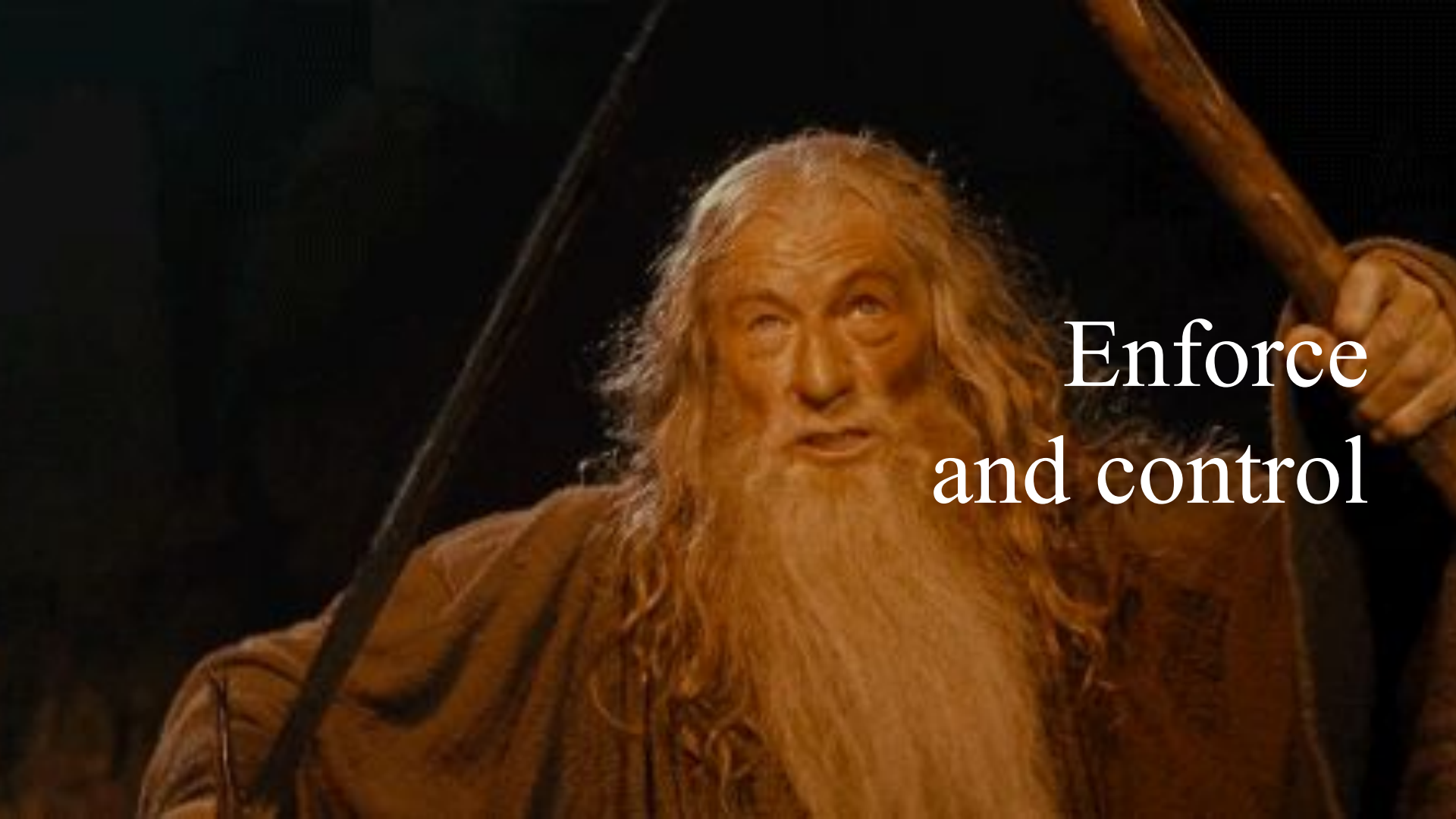
API styleguide will need to evolve


Update
Improve

Is that all?

One last thing

Rules may not be followed


Enforce
and control

Is this
control only
done by people?

Mostly but some try to automate
part of these controls...

Chapter IV

THE API DESIGN HAVENS

OPEN API
INITIATIVE

```yaml
paths:
  /persons:
    get:
      summary: Gets some persons
      description: Returns a list containing all persons. The list supports paging.
      parameters:
       - $ref: "#/parameters/pageSize"
       - $ref: "#/parameters/pageNumber"
      responses:
        200:
          description: A list of Person
          schema:
            $ref: "#/definitions/Persons"
        500:
          $ref: "#/responses/Standard500ErrorResponse"
    post:
      summary: Creates a person
      description: Adds a new person to the persons list.
      parameters:
        - name: person
          in: body
          description: The person to create.
          schema:
            $ref: "#/definitions/Person"
```

Machine readable API description

Zalando's Zally
A minimalistic API linter

https://github.com/zalando-incubator/zally

Zally API Linter

Check if your SWAGGER Schema conforms to Zalando's REST API Guidelines

BY URL EDITOR RULES

ENTER FULL PATH TO YOUR SWAGGER FILE

https://raw.githubusercontent.com/OAI/OpenAPI-Specification/master/examples/v2.0/json/petstore.json

VALIDATE

Rule #174 is violated

VIOLATIONS

- MUST – Use common field names

#/definitions/Pet: field 'id' has type 'integer' (expected type 'string')

Rule: /#174

Paths:

- #/definitions/Pet

- MUST – Define and Assign Access Rights (Scopes)

API Guidelines

Table of Contents
1. Introduction
2. Principles
3. General Guidelines
4. Security
5. Compatibility
6. JSON Guidelines
7. API Naming
8. HTTP
9. Performance
10. Pagination
11. Hypermedia
12. Data Formats
13. Common Data Types
14. Common Headers
15. Proprietary Headers
16. Deprecation
17. API Operation
18. Events

Appendix A: References
Appendix B: Tooling
Appendix C: Changelog

Must: Use common field names and semantics [174]

There exist a variety of field types that are required in multiple places. To achieve consistency across all API implementations, you must use common field names and semantics whenever applicable.

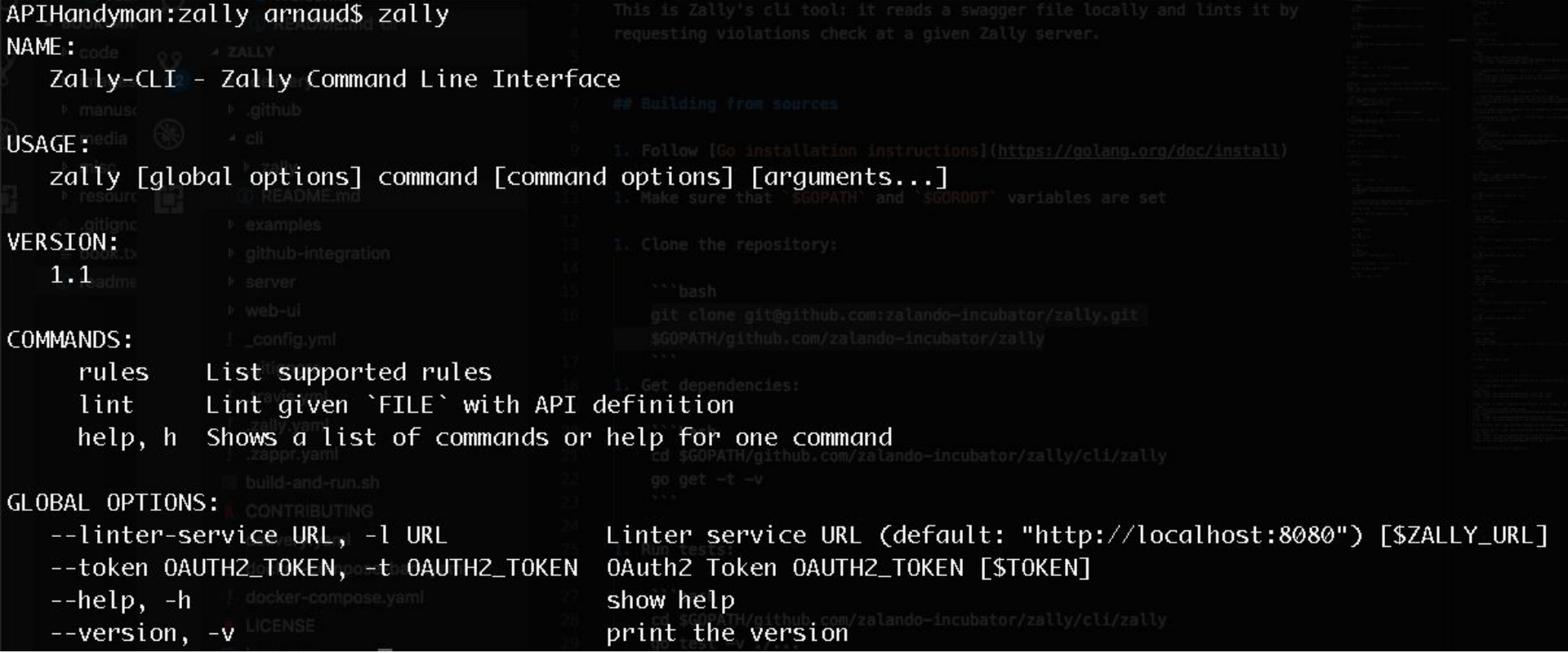
Generic Fields

There are some data fields that come up again and again in API data:

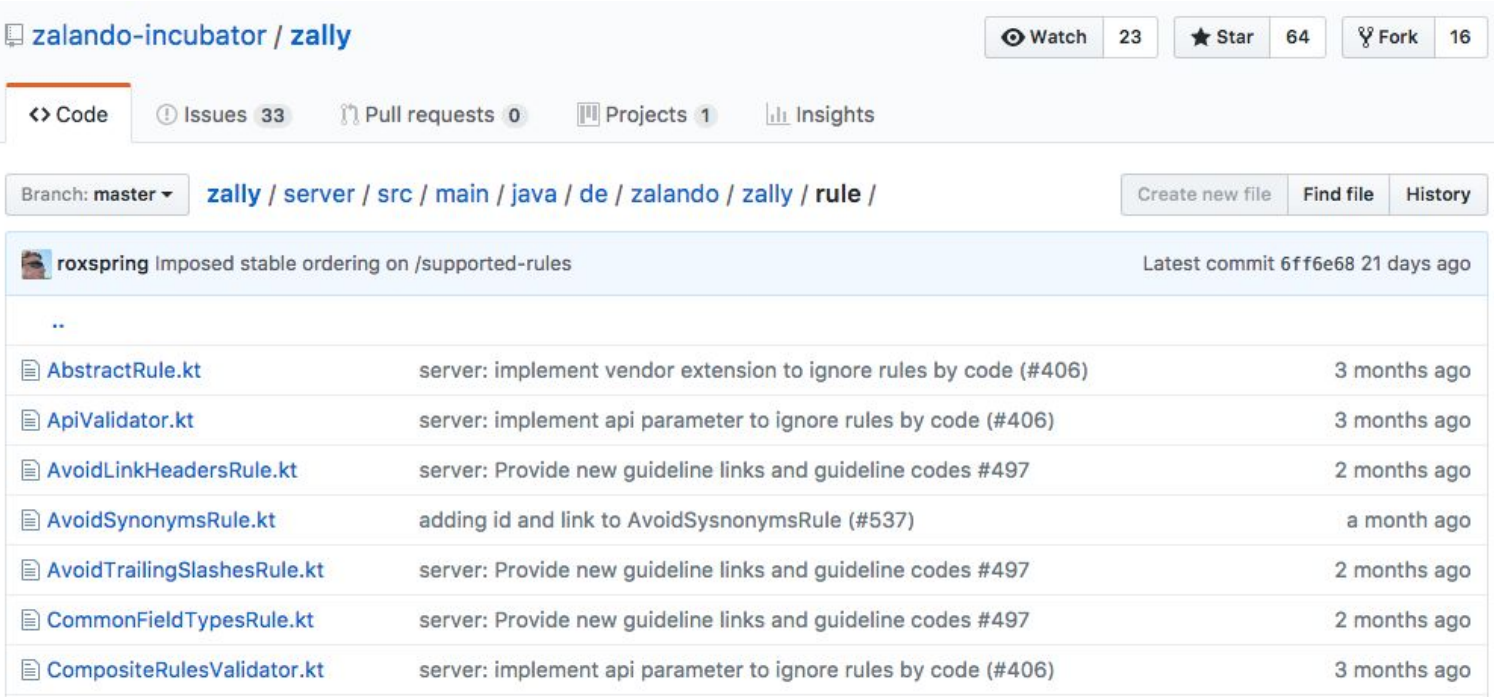
- `id` : the identity of the object. If used, IDs must opaque strings and not numbers. IDs are unique within some documented context, are stable and don't change for a given object once assigned, and are never recycled cross entities.
- `xyz_id` : an attribute within one object holding the identifier of another object must use a name that corresponds to the type of the referenced object or the relationship to the referenced object followed by `_id` (e.g. `customer_id` not `customer_number` ; `parent_node_id` for the reference to a parent node from a child node, even if both have the type `Node`)
- `created` : when the object was created. If used, this must be a `date-time` construct.
- `modified` : when the object was updated. If used, this must be a `date-time` construct.
- `type` : the kind of thing this object is. If used, the type of this field should be a string. Types allow runtime information on the entity provided that otherwise requires examining the Open API file.

Example JSON schema:

Rule #174 is defined in guidelines



There’s also an API and a CLI



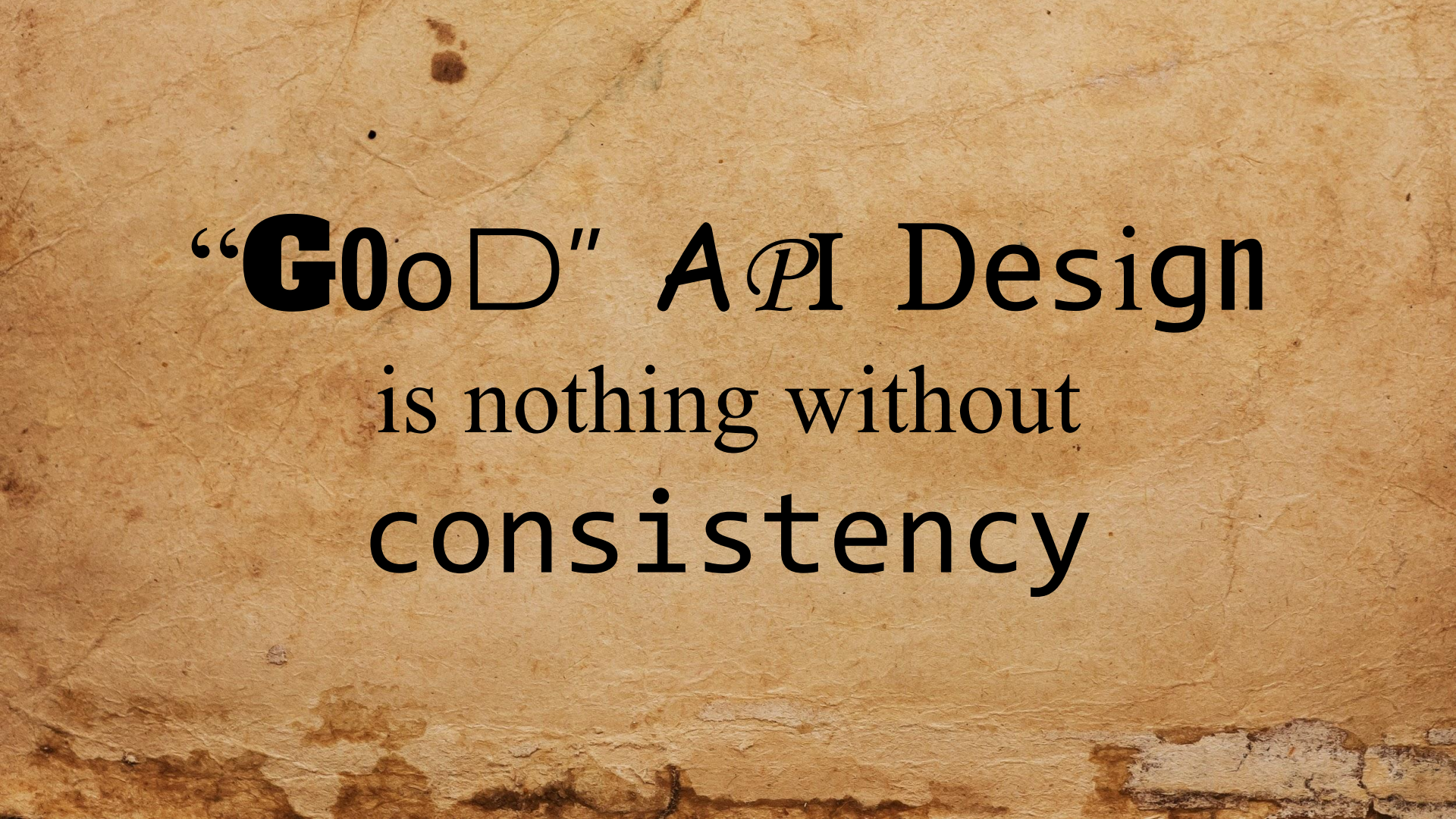
You can define your own rules

So, no more
humans?

Such tool really helps
But we still need humans

Epilogue

THE FELLOWSHIP OF API DESIGNERS


"GOoD" API Design
is nothing without
consistency

No consistency
without
clearly defined rules

Rules

are nothing without

governance

(establish, share, update, control)

It’s time to build your fellowship of API designers

THE LORD OF API DESIGNS

BY

ARNAUD LAURET @APIHANDYMAN

THE END