Creating Integrated API Documentation Experiences for Users

Chris Smith @zxchris

Rob Woodgate @agiledoc

Companies House

A lot of APIs are **created and documented** by developers

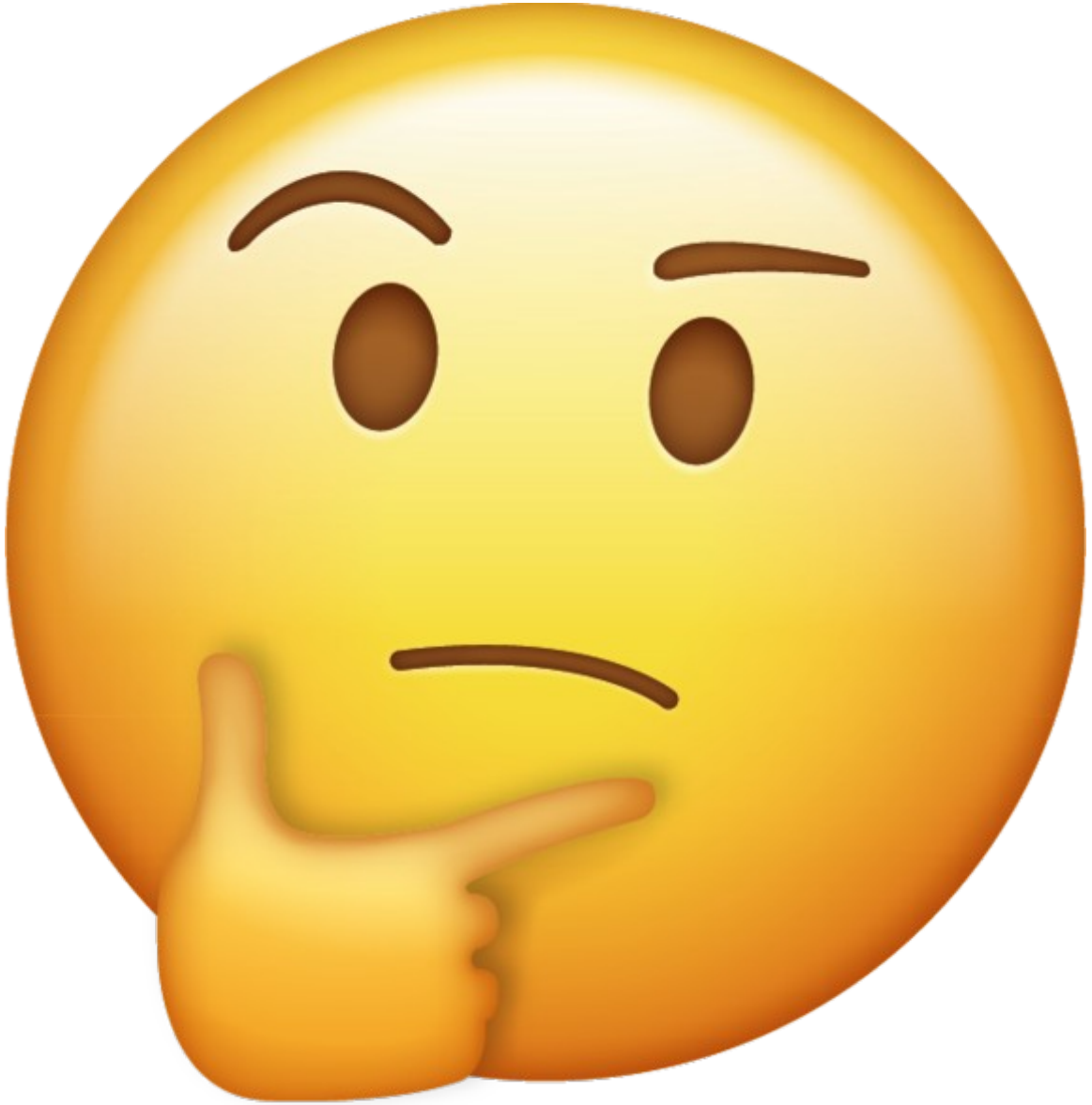
Is the documentation written by developers…

- Good?
- Useful?
- Easy to comprehend?

Yes?

Honestly?

No?



You can be part of a better system!

Hooray!!!

How?

1. Understand the pitfalls

2. Recognise your goal

3. Get to your goal

Common API Pitfalls

1. Designed at the keyboard
2. API specs produced from the code
3. No standard structure
4. No naming conventions
5. No UX designer
6. No content writer
7. Specification and docs are separate deliverables
8. Developer does everything

This leads to a disconnect.....

Formal API Spec **Authored Docs**

1. Understand the pitfalls ✓

2. Recognise your goal

What is your goal?

To deliver a documented API that meets the customer need

What does this mean in reality?

Working together

It’s not about sitting next to each other.

It’s about sharing the same goal.

Your goal is to produce a useful, comprehensible deliverable

If the API specification is difficult to understand, you've already failed

You need to help your developer understand the user needs

If the API is easy to comprehend then it requires less documentation

You should be an integral part of the API design process

Get API developers & writers working together

This should be standard practise

It’s not easy

It’s not quick

Some people will RESIST

But, resistance….

...is futile

You need to be part of a collective

Get API developers & writers DESIGNING together

This approach avoids a lot of the common pitfalls

1. Understand the pitfalls ✓

2. Recognise your goal ✓

3. Get to your goal

How?

Spoiler:

We don’t have all the answers

But we can help!

Let’s look at an example….

```
{
  “name”  : “string”
  “lfp”   : “bool”
  “sent”  : “datetime”
  “made_up_date” : “date”
}
```

Let’s look at another example….

```
{
  “financial_accounts_type” : “string”
  “has_late_filing_penalty” : “bool”
  “penalty_sent_at” : “datetime”
  “period_ends_on”  : “date”
}
```

```
        {
          "name" : "string"
          "lfp"  : "bool"
          "sent" : "datetime"
          "made_up_date" : "date"
        }


{
  "financial_accounts_type" : "string"
  "has_late_filing_penalty" : "bool"
  "penalty_sent_at" : "datetime"
  "period_ends_on"  : "date"
}
```

Some suggestions to get you thinking:

1. Enumerations suffixed with **_type**
2. Dates suffixed with **_on**
3. Date-times suffixed with **_at**
4. Booleans prefixed with **has_** (or **is_**)

Tactic 1:

Consistency

Tactic 2:

Comprehensibility

Tactic 3:

Specs as UX/UI

Tactic 4:

Specification before code

Tactic 5:

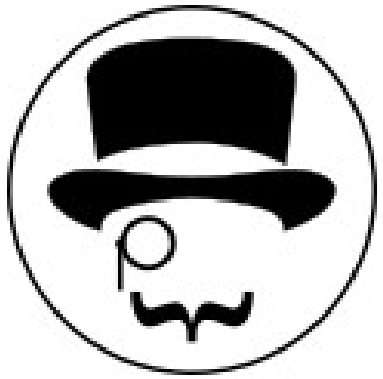
Teamwork

Tactic 6:

Single consumable

A single consumable?

We use this:



DapperDox

Why?

It combines specs with docs and graphics…

...in a single consumable...

...that devs and writers can work on together!

Time for a demo