



Porting a Django View to an MFE

An **Open edX** lightning talk by **Nathan Sprenkle**



Micro-frontend application

Why MFES?

- + Standalone React / Redux applications
- + **Outside of edx-platform** ←———— *Deconstructing the Monolith*
- + Client heavy instead of server heavy
- + Paragon: component reuse, branding, and accessibility



Talk Inspiration:

Create a new Student Dashboard

Page is high visibility, high complexity.

Requires near feature parity.

~ 1K lines of code for gathering & composing page context.



Browse recently launched courses and see what's new in your favorite subjects.

🔍 Explore New Courses

My Courses



Defense Against the Dark Arts

Hogwarts - D.A.D.A 601
Starts - Jan 24, 2024



Resume Course



Potions

Hogwarts - POT 418
Ended - Dec 31, 2022



View Archived Course



Herbology

Hogwarts - HERB 512
Started - Dec 31, 2019



Resume Course



edX

Legal

Connect

Problem!

Challenging to reimplement from scratch and meet requirements.

But we found a *useful pattern*...

—

... from investigating these 2 architectures



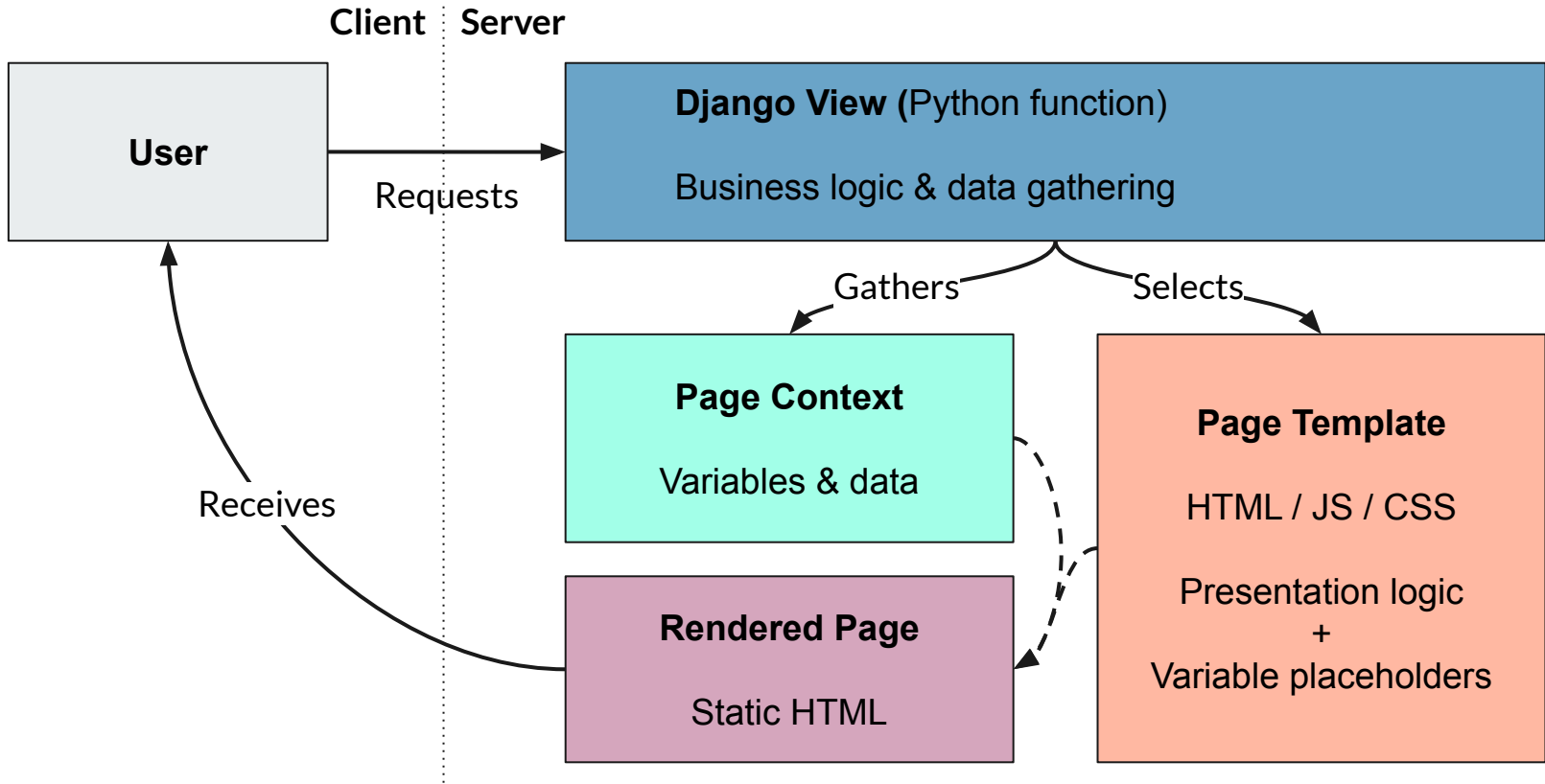
Mirror as much of the core logic and structure as possible...

... while allowing for clearly scoped refactors & following best-practices.

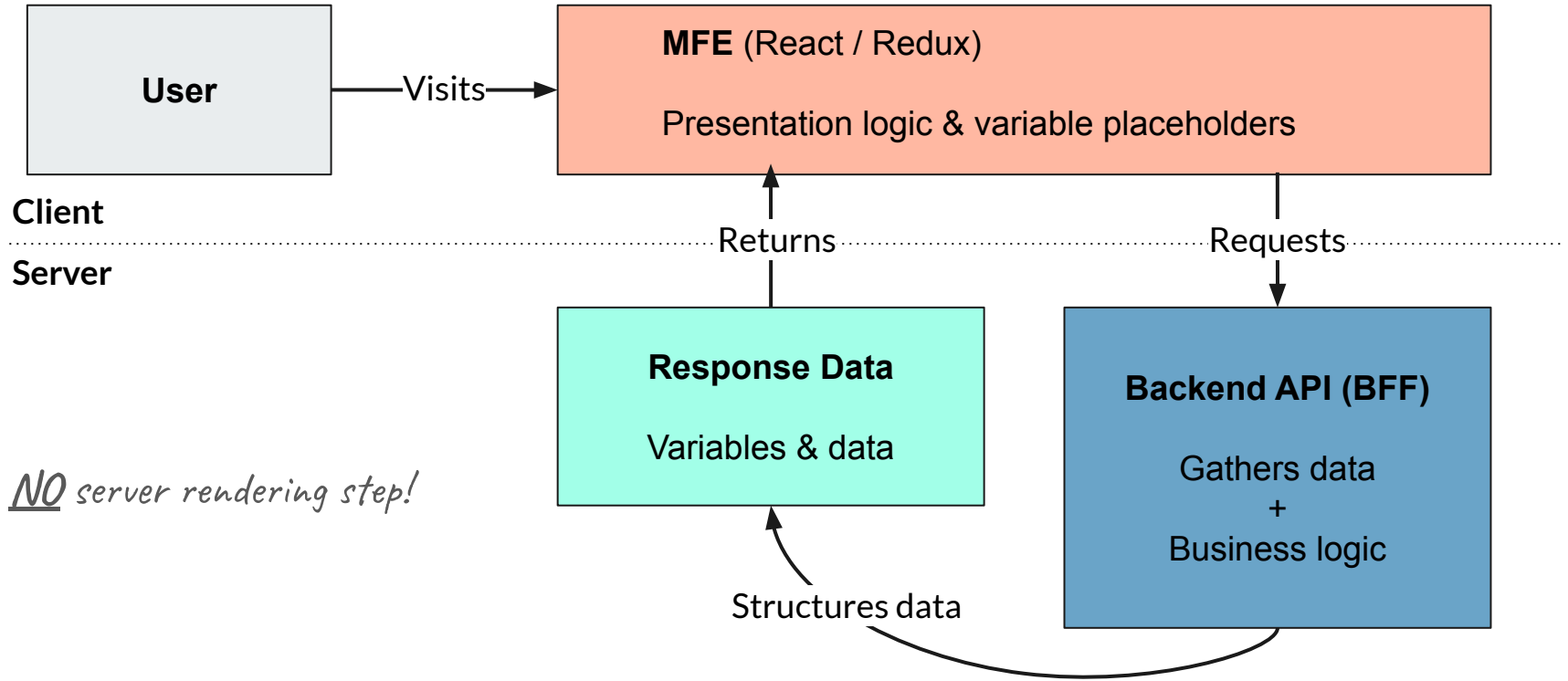


Let's see how by digging into those architectures...





Django views! What are?



But what about Micro-frontend Applications (MFEs)?!?!

Hopefully you're thinking...

“Wow, these have some clear similarities!”

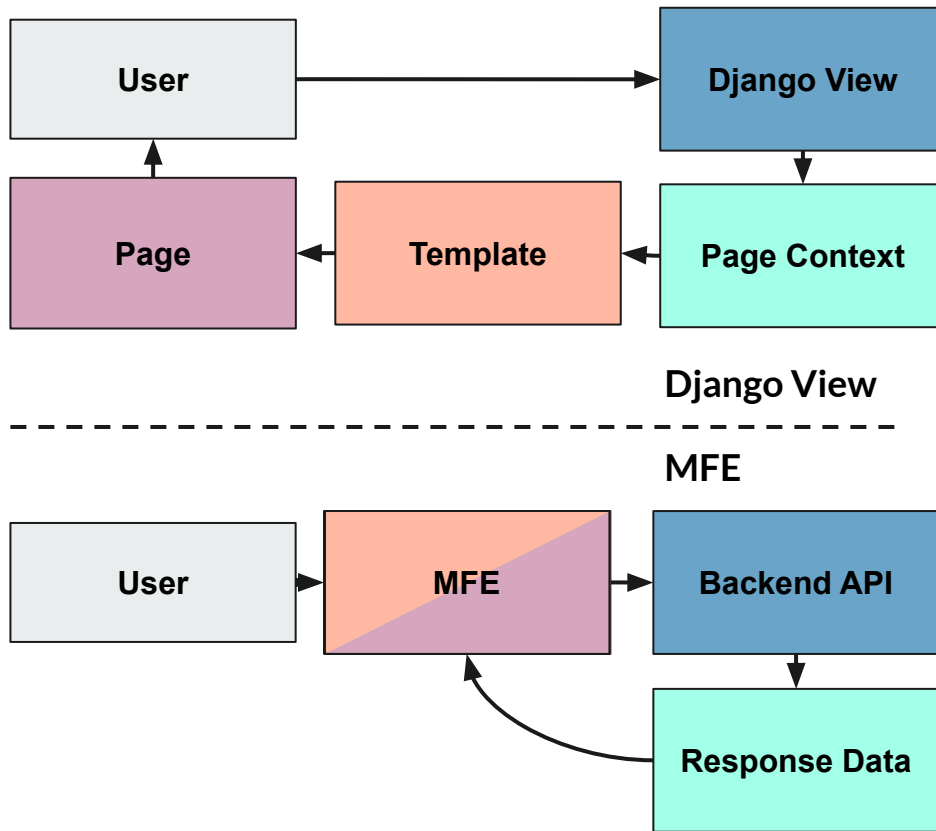


Yes!

And we can use those similarities...

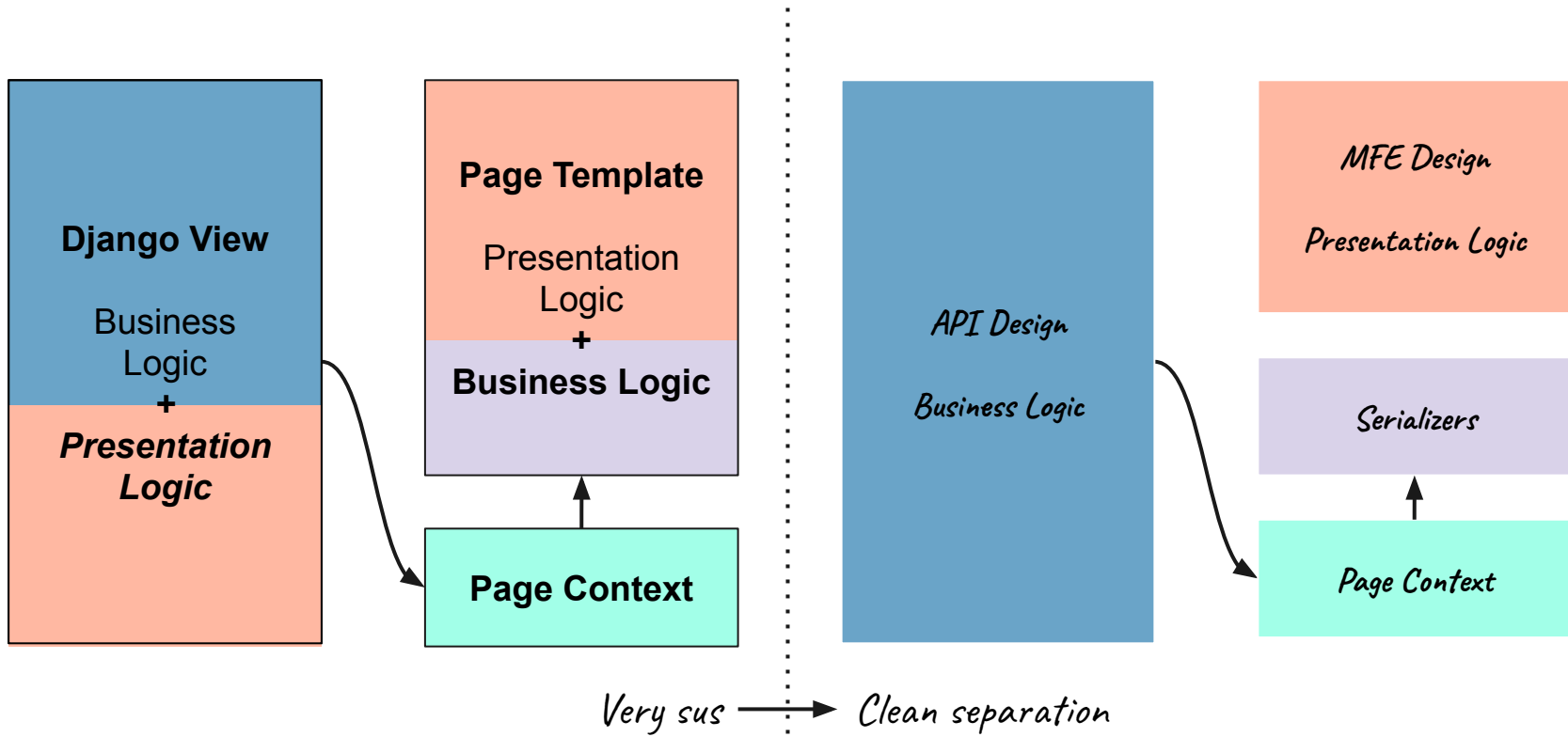
To map logic to our new implementation

1. BFF : Django View
2. MFE : Template / Rendered Page
3. Page context : Response Data

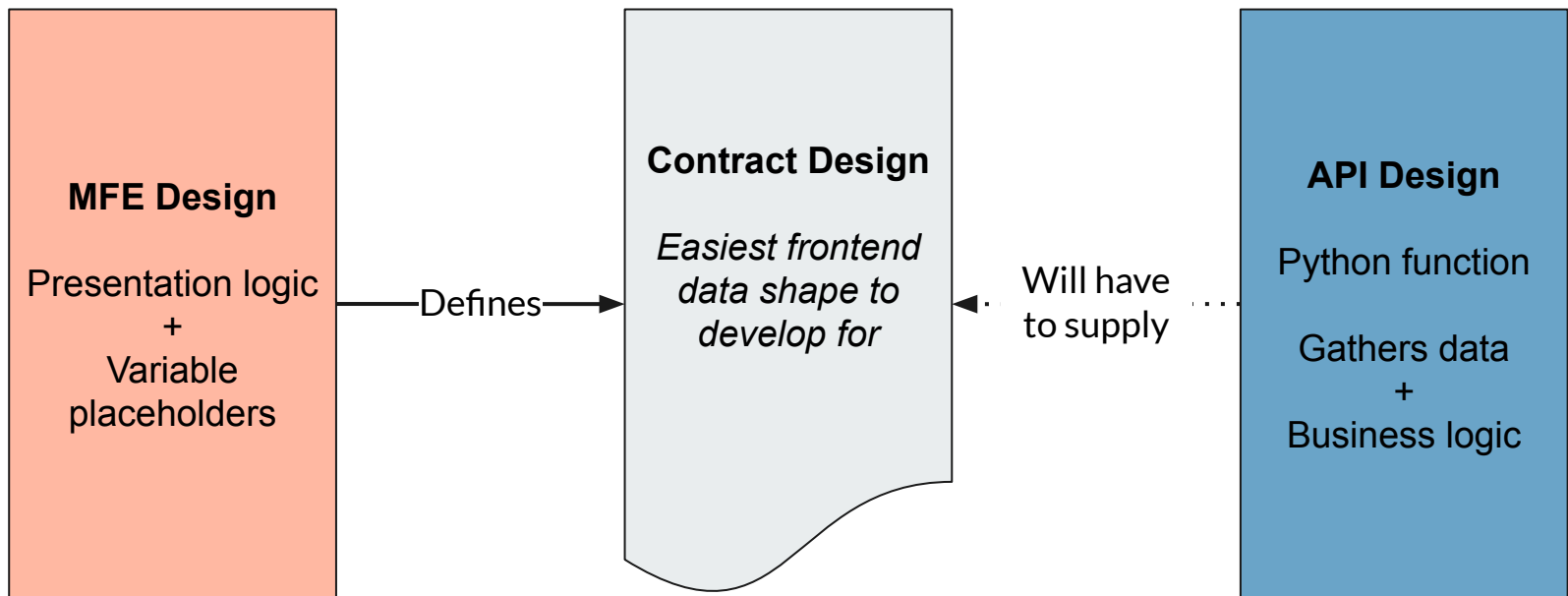


Our Process, in 4 steps:





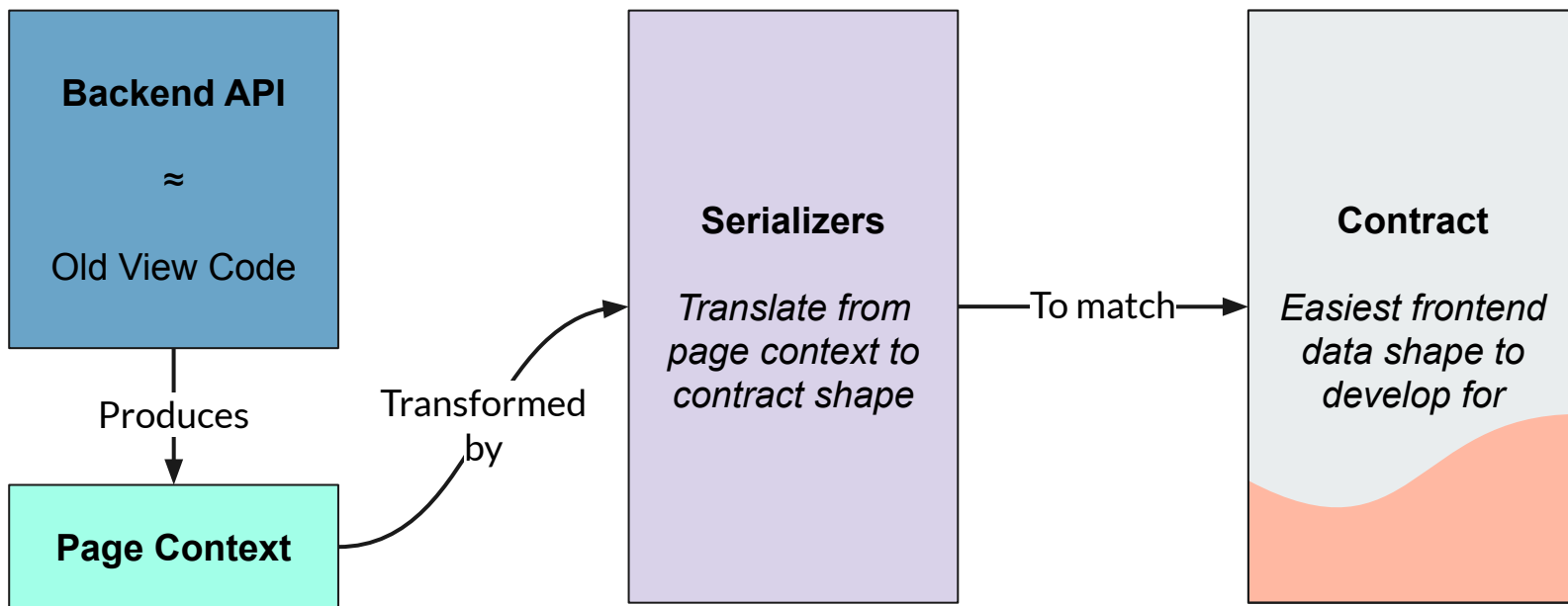
Step 1: Duplicate backend logic, clearly delineate logical boundaries



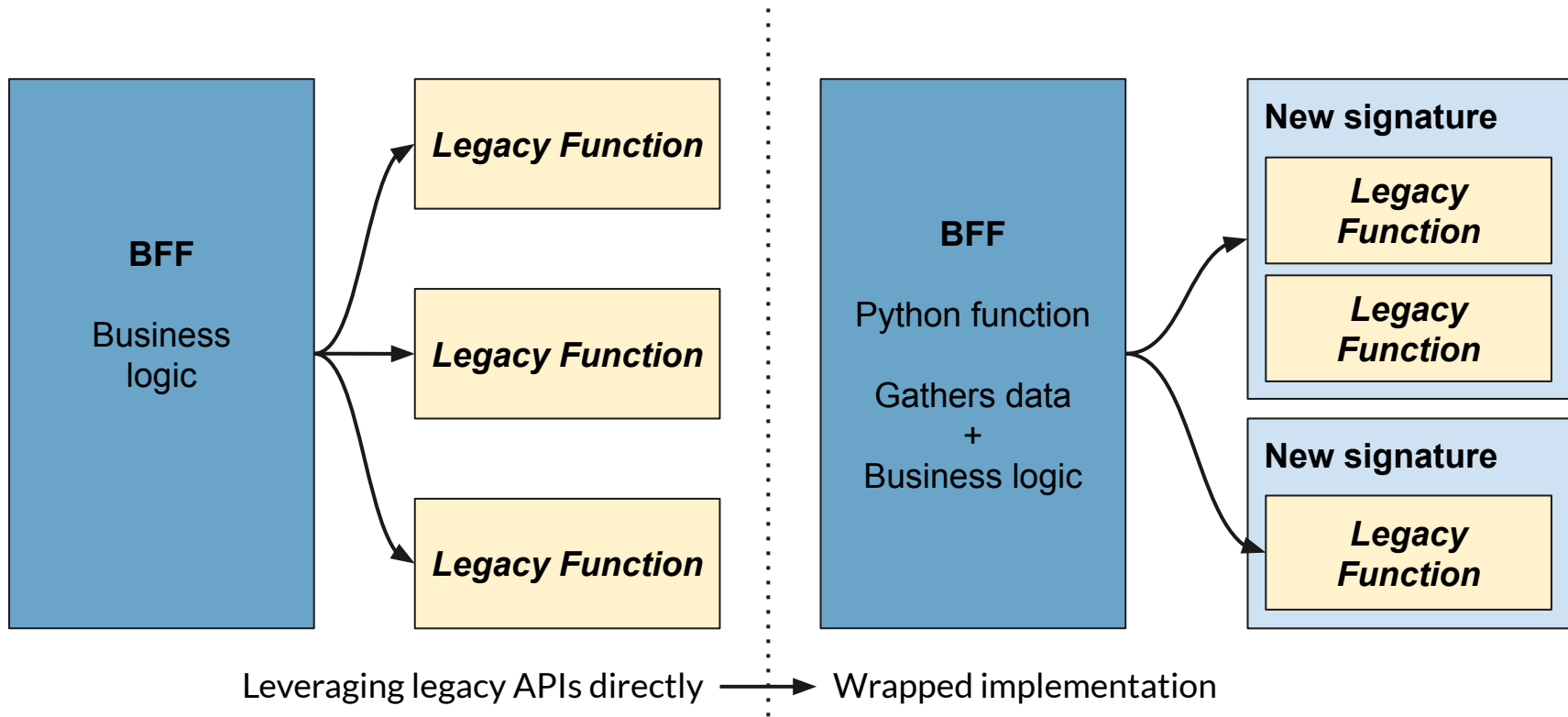
Backend-informed frontend-driven development



Step 2: Define frontend contracts, start building MFE



Step 3 : Translate data from backend to frontend



Step 4: Wrap calls to legacy APIs, selectively profile and refactor



Pattern TL;DR

1. Duplicate backend logic, clearly delineate logical boundaries
2. Define frontend contracts, start building MFE
3. Translate data from backend to frontend
4. Wrap calls to legacy functions, profile and selectively refactor

End Result:

Clean, new MFE frontend.

Retains much existing logic & nuance.

Allows for refactorability.

Quick development process.

My Courses

Refine



Defense Against the Dark Arts

Hogwarts • D.A.D.A 601 • Course starts January 24, 2024

Upgrade

Resume

i Grade required to pass the course: 50%



Potions

Hogwarts • POT 418 • Course ended December 31, 2022

Upgrade

View Course

i Grade required to pass the course: 50%



Herbology

Hogwarts • HERB 512 • Course ends December 31, 2998

Upgrade

Resume

i Grade required to pass the course: 50%

Hope this is useful!

Questions? Want to nerd out?

openedx 
2U 

@nsprenkle
nsprenkle@2u.com



Look for this guy!

Thank You

