Micro Frontends

Communication Patterns

Satyam Pandey | Sravya Chodisetti

Table of contents

Speakers

Satyam Pandey

Engineering Manager

Global Merchant Lending, PayPal Credit

Sravya Chodisetti

Senior Software Engineer

Global Merchant Lending, PayPal Credit

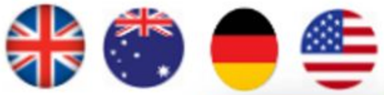
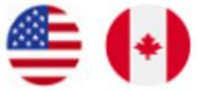
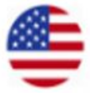
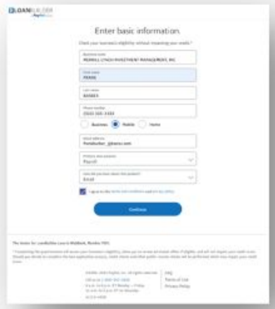
PayPal

Global Merchant Lending in PayPal

Democratize Credit to our 30M+ Merchant partners!

Offer Fast, Fair, Flexible Credit solutions!

Grow by enabling growth to our Merchant partners!

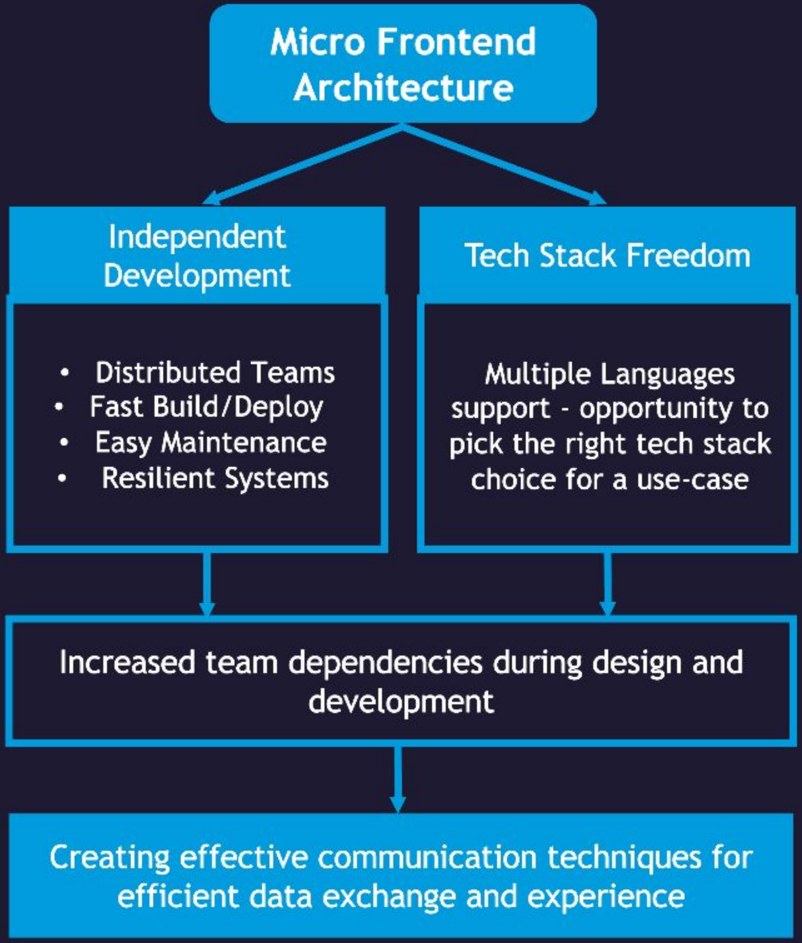
Product	**PayPal** Working Capital	**PayPal** Business Loan	LOANBUILDER
Key Features	• Sales Based Repayments • No Credit Check	• Traditional Term Loans • Use Credit Check	• Paycheck Protection Loans • Use Credit Check
Target Audience	• Small, Micro Sellers with seasonal / unpredictable cashflows	• Mature merchants with predictable cash flows	• Support Government Initiatives during COVID19
Countries Served			 

PayPal

Why is communication important in Micro Frontend Architecture?

Overview

PayPal



Micro Frontend Architecture

Independent Development

- Distributed Teams
- Fast Build/Deploy
- Easy Maintenance
- Resilient Systems

Tech Stack Freedom

Multiple Languages support - opportunity to pick the right tech stack choice for a use-case

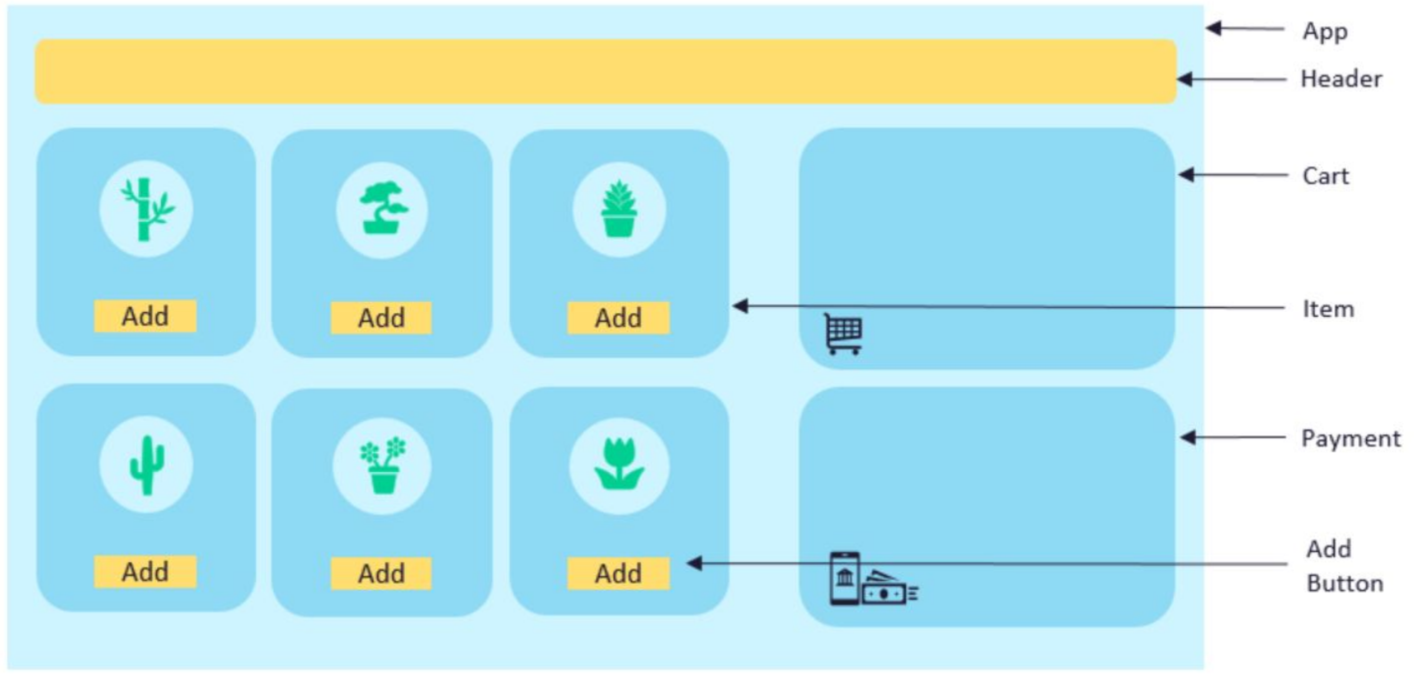
Increased team dependencies during design and development

Creating effective communication techniques for efficient data exchange and experience

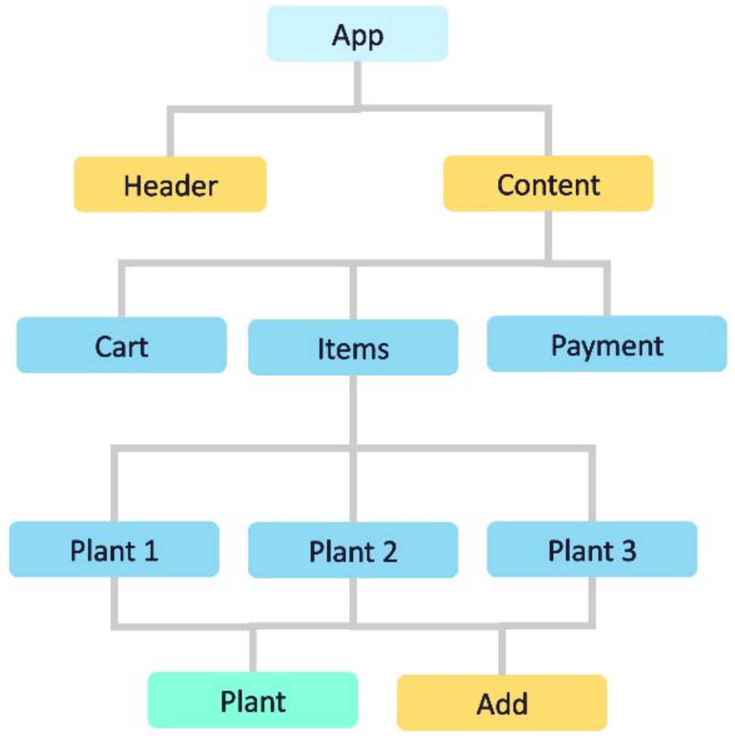
PayPal

Introduction

Website



Component Structure



Introduction

1 Parent to Fragment

- Element Attributes
- Connected Callback
- Attribute Changed Callback

2 Fragment to Parent

- Custom Events
- Event Listeners

3 Fragment to Fragment

- DOM Manipulation
- Attributes and Callbacks
- Event Bus
- Broadcast Channel API

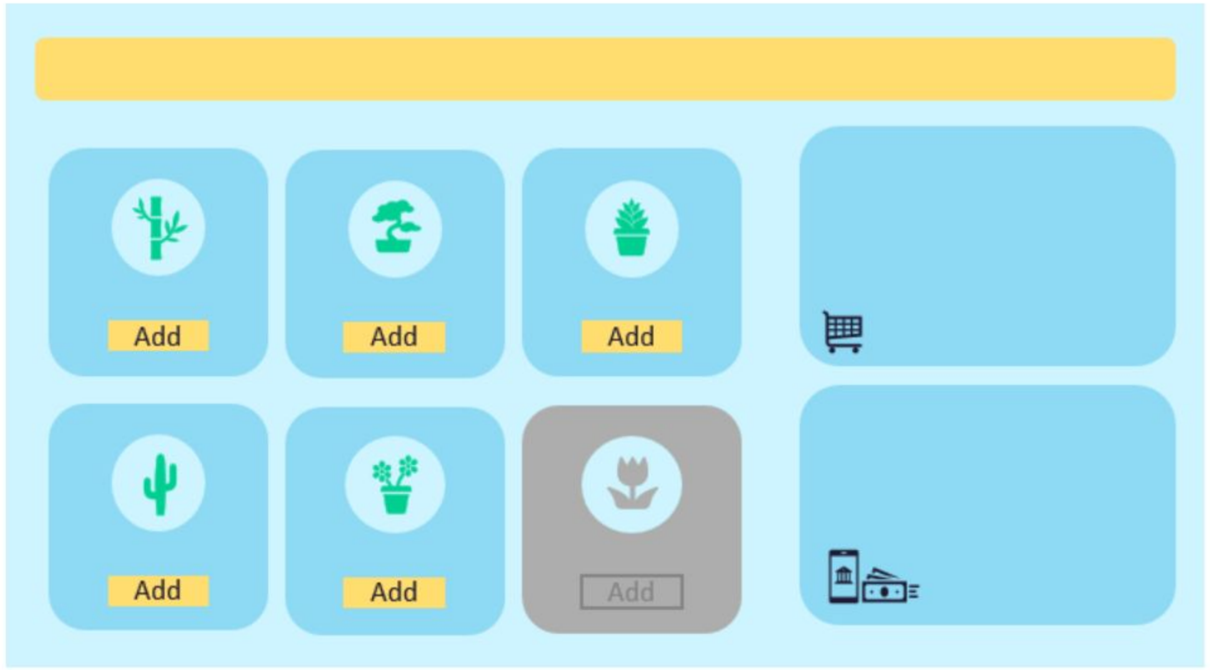
4 Global Communication

- URL Params
- Global Context and State
- State Management Libraries (Redux)

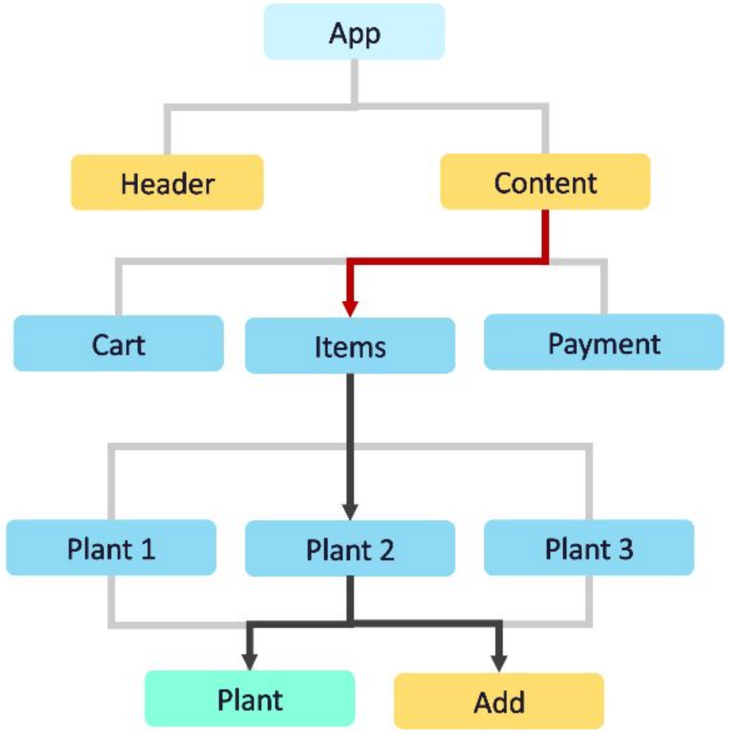
Communication Patterns

Parent to Fragment - Scenario

Website



Component Structure



PayPal

Communication Patterns

Parent to Fragment – Attribute Implementation

1 Create a custom attribute on the element

- Create a custom attribute called 'itemData' on the item component

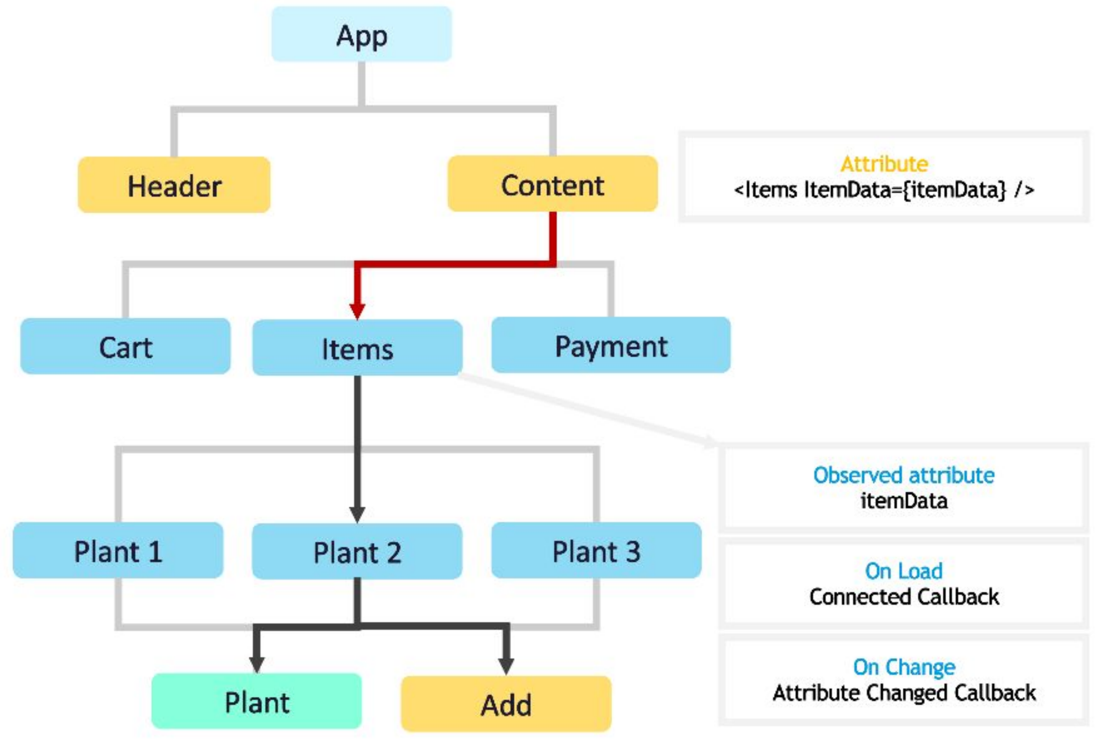
2 Pass data as attribute-value to the Fragment

- Pass the plant data to the plant component (instock -true/false) to render plant out-of-stock

3 Create lifecycle methods on observed attribute

- Create connectedCallback or attributeChangedCallback hooks on 'inStock'
- Create handlers for the callbacks
- Render the plant item as enabled or disabled based on the value of 'inStock' attribute in the Fragment

Component Structure

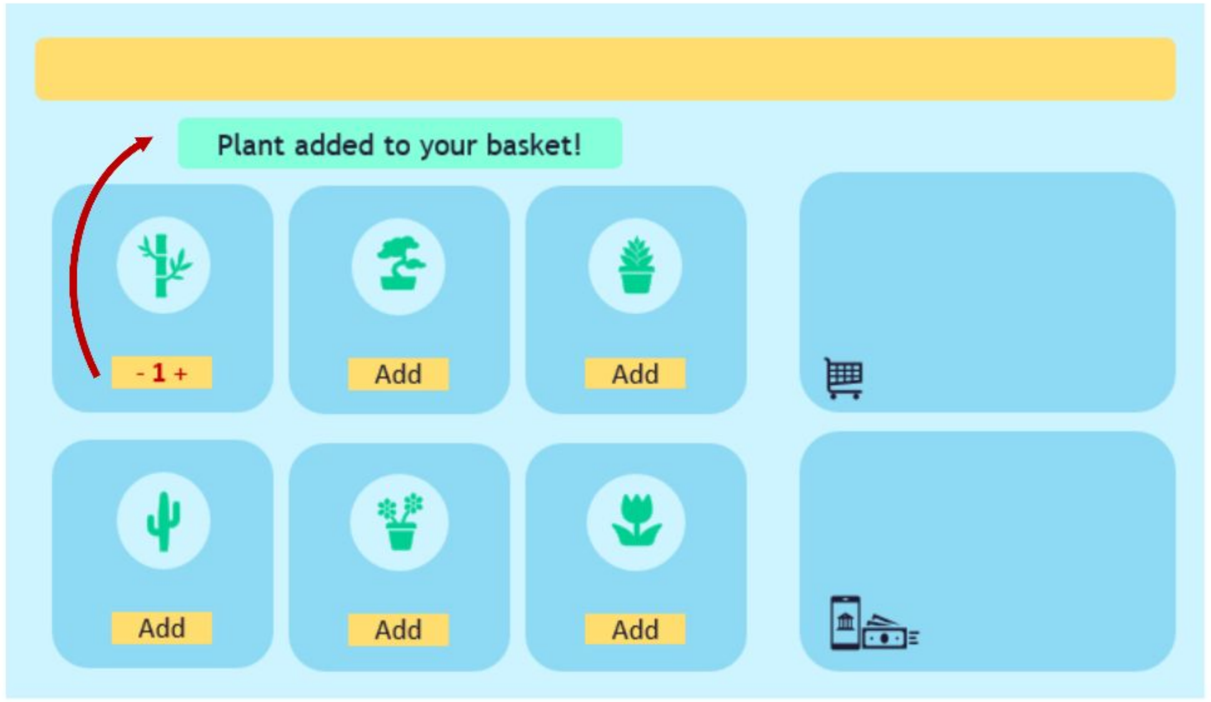


PayPal

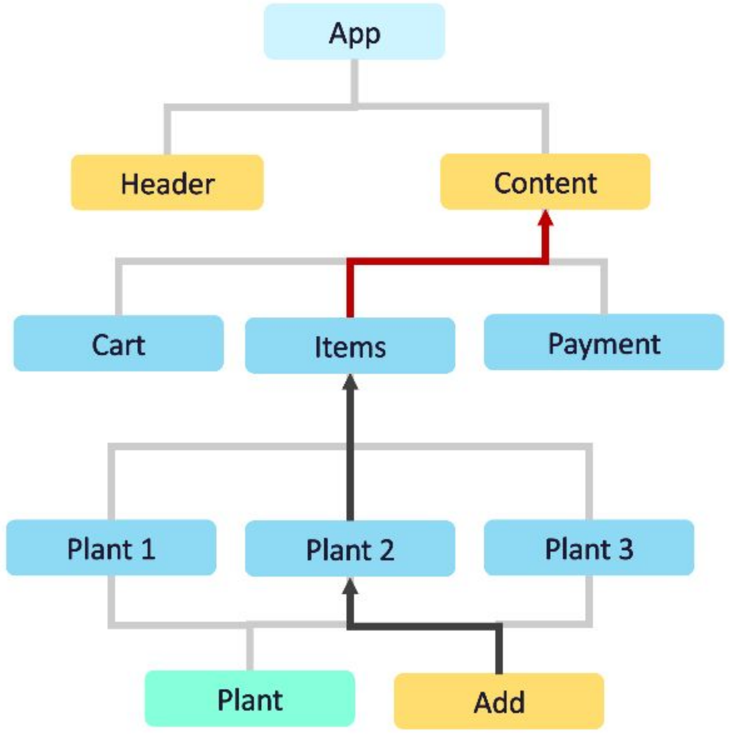
Communication Patterns

Fragment to Parent - Scenario

Website



Component Structure



Communication Patterns

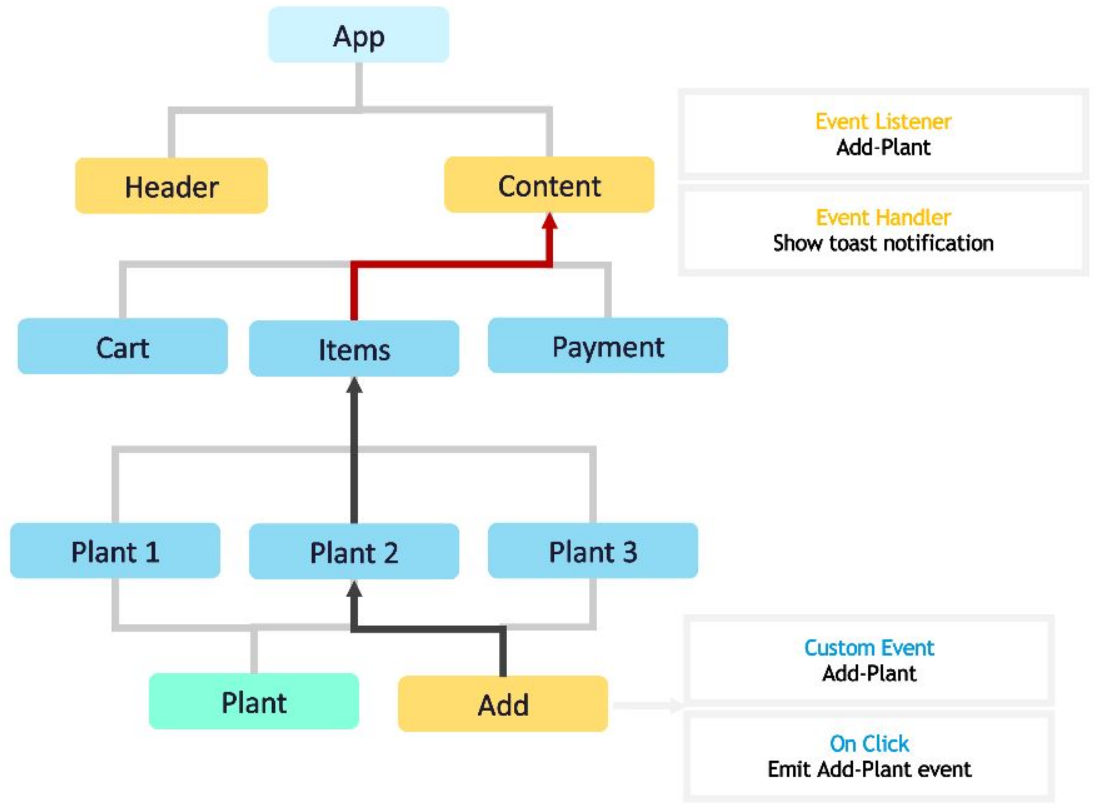
Fragment to Parent – Custom Event Implementation

1. **Create a custom event on the element**
   - Create a custom event called 'add-plant' on the 'Add' Button

2. **Emit the event on action**
   - Emit the 'add-plant' event on clicking the button

3. **Create event listener and a handler on the parent**
   - Add an event listener on the 'content' element
   - Listen for the 'add-plant' event
   - Execute handler for the next steps
     - send a toast notification for successful update

Component Structure

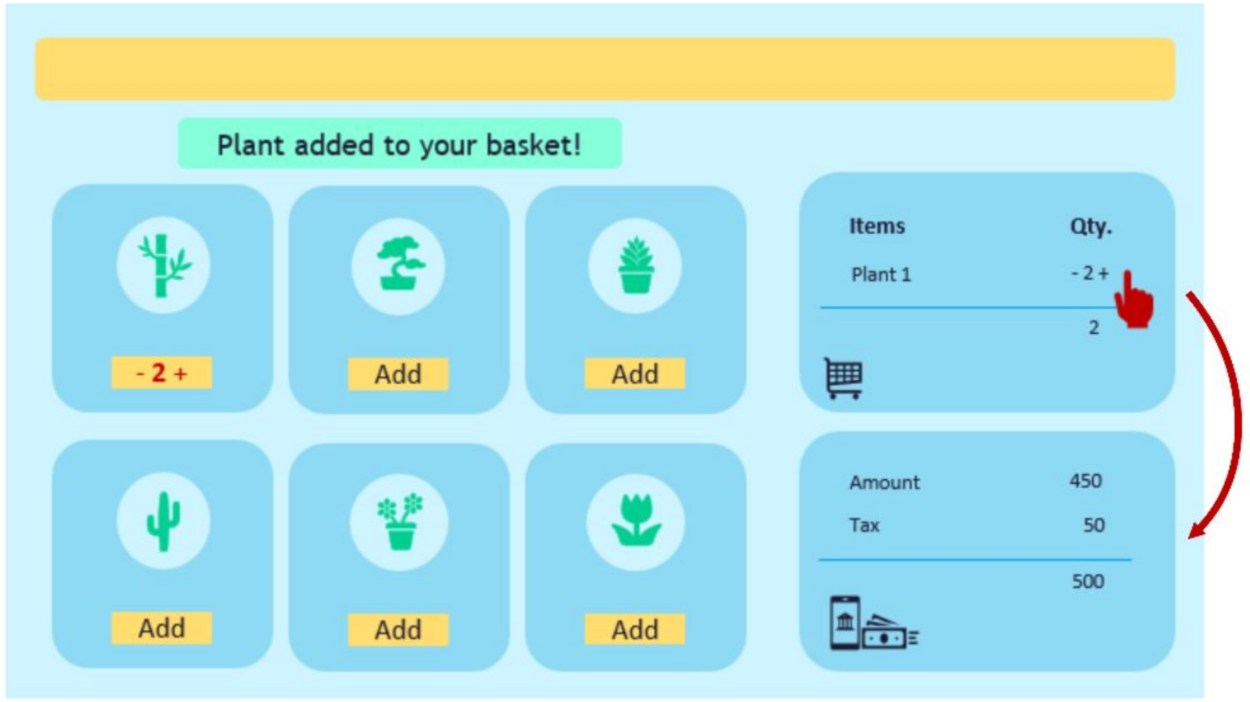


PayPal

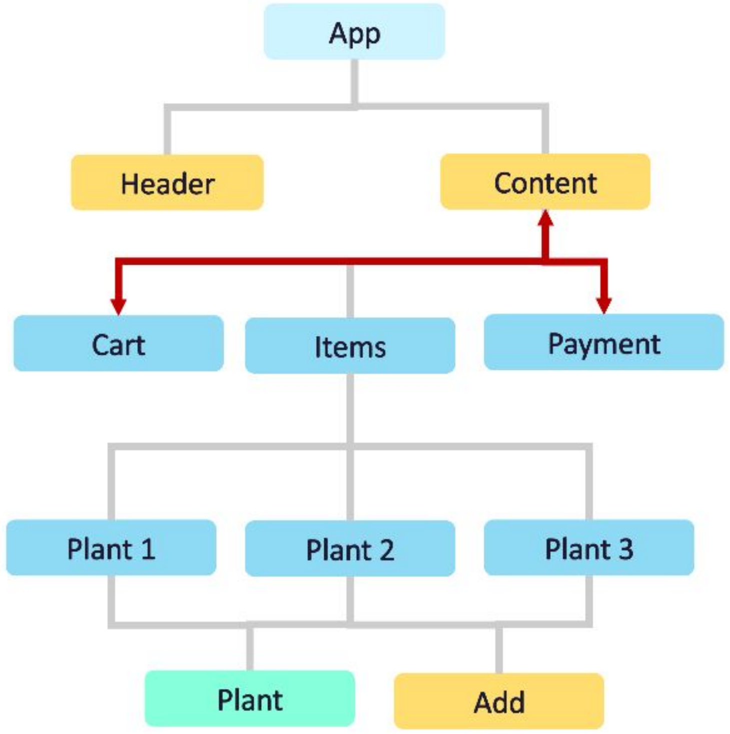
Communication Patterns

Fragment to Fragment - Scenario

Website



Component Structure

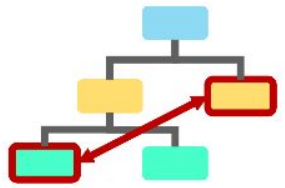


Communication Patterns

Fragment to Fragment – Implementation – 4 ways

1 Direct DOM reference

- Using DOM navigation properties
  - Get elements by X
  - Query selectors
- Easy to implement
- Introduces tight-coupling
  - Needs DOM context to navigate and identify
  - Defies the concept of MFA

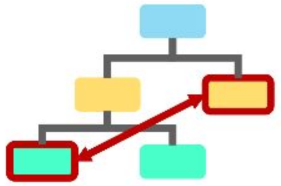


PayPal

Communication Patterns

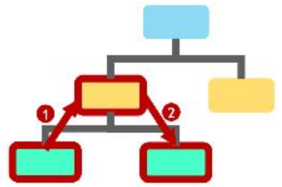
Fragment to Fragment – Implementation – 4 ways

1 Direct DOM reference



2 Routing via Parents

- Combine the previous communication patterns –
  - Trigger an event from Source Fragment to Parent
  - Pass the attribute from Parent to Destination Fragment
- Easy to implement
- Increased no. of communications between fragments across DOM tree

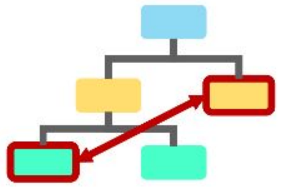


PayPal

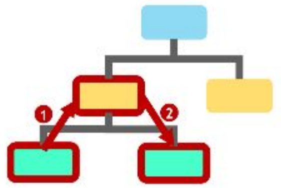
Communication Patterns

Fragment to Fragment – Implementation – 4 ways

1 Direct DOM reference

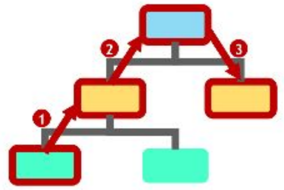


2 Routing via Parents



3 Event Bus via Custom Events

- Dispatch custom events and let the events bubble
- Dispatch custom events directly to the Window
- Event bubbling allows modification at parents in the tree
    - Stop propagation / Prevent default
    - Intercept and modify payload
- Heavily dependent on the DOM structure

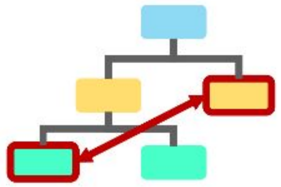


PayPal

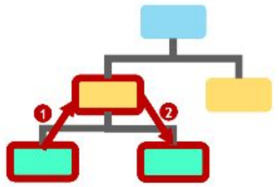
Communication Patterns

Fragment to Fragment – Implementation – 4 ways

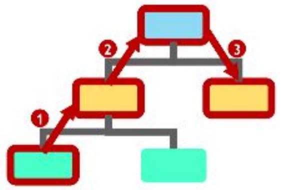
1 Direct DOM reference



2 Routing via Parents

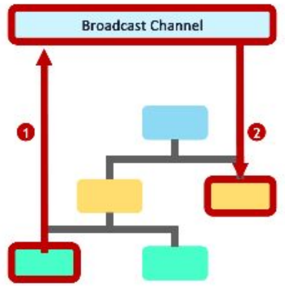


3 Event Bus via Custom Events



4 Publish / Subscribe

- Publish and Subscribe model via Broadcast APIs
  - Create and connect to a channel
  - Send messages to the channel
  - Receive messages and perform necessary actions
- Communicate across windows/tabs/iframes
- Channels are open to everyone to publish messages – pro or con depending on the use-case

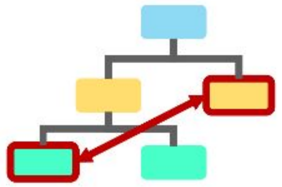


PayPal

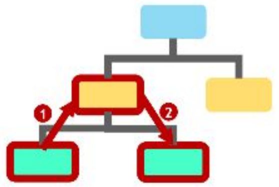
Communication Patterns

Fragment to Fragment – Implementation – 4 ways

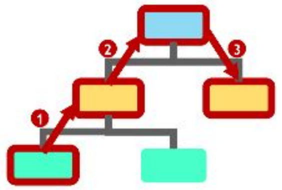
1 Direct DOM reference



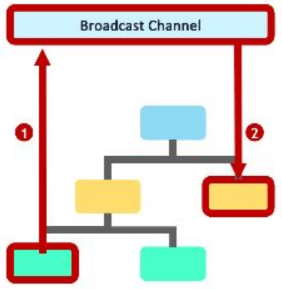
2 Routing via Parents



3 Event Bus via Custom Events



4 Publish / Subscribe

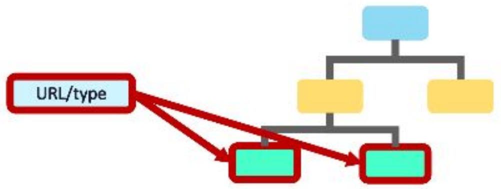


Communication Patterns

Global Communication – Scenarios and Solutions

1 Product Type

- Scenario
  - You sell multiple products on your website (plants, pots, soil, seeds, etc.)
  - How do you pass the product type information to all fragments in the application?
- Solution
  - URL params
    - www.mfa.com/***plants***
    - www.mfa.com/***pots***
    - www.mfa.com/***seeds***
  - Each Fragment reads the product type from the query/path params in the URL
  - Each Fragment maintains a local state and handles the rendering based on the params

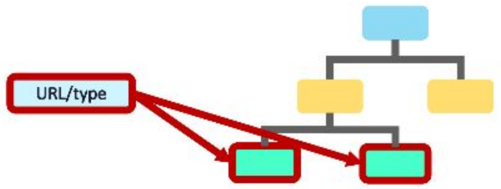


Communication Patterns

Global Communication – Scenarios and Solutions

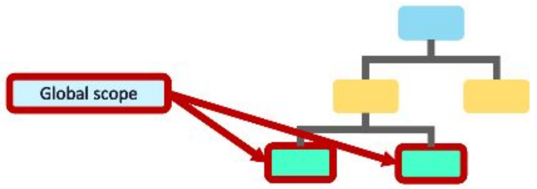
1 Product Type

- Scenario
  - You sell multiple products on your website (plants, pots, soil, seeds, etc.)
  - How do you pass the product type information to all fragments in the application?
- Solution
  - URL params
    - www.mfa.com/***plants***
    - www.mfa.com/***pots***
    - www.mfa.com/***seeds***
  - Each Fragment reads the product type from the query/path params in the URL
  - Each Fragment maintains a local state and handles the rendering based on the params



2 User Details

- Scenario
  - You have multiple user categories who use your website (Platinum, gold, silver, etc.) and the features and User experience varies for each category.
  - How do you pass the user details (category, country, language, etc.) to all the fragments in the application?
- Solution
  - Global Scope and Context
    - JS global scope and context APIs
  - State Management Libraries
    - Create Global state for shared data and local state for each fragment to avoid tight-coupling



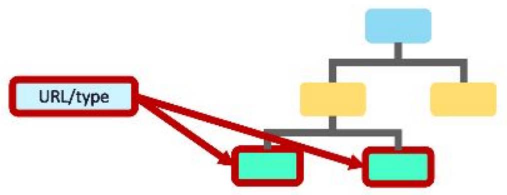
PayPal

Communication Patterns

Global Communication – Scenarios and Solutions

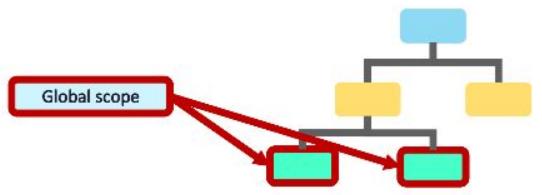
1 Product Type

- Scenario
  - You sell multiple products on your website (plants, pots, soil, seeds, etc.)
  - How do you pass the product type information to all fragments in the application?
- Solution
  - URL params
    - www.mfa.com/***plants***
    - www.mfa.com/***pots***
    - www.mfa.com/***seeds***
  - Each Fragment reads the product type from the query/path params in the URL
  - Each Fragment maintains a local state and handles the rendering based on the params



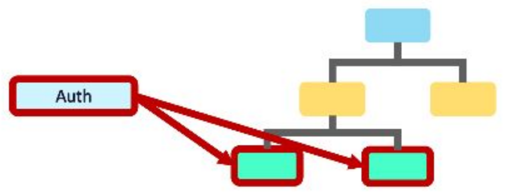
2 User Details

- Scenario
  - You have multiple user categories who use your website (Platinum, gold, silver, etc.) and the features and User experience varies for each category.
  - How do you pass the user details (category, country, language, etc.) to all the fragments in the application?
- Solution
  - Global Scope and Context
    - JS global scope and context APIs
  - State Management Libraries
    - Create Global state for shared data and local state for each fragment to avoid tight-coupling



3 Auth

- Scenario
  - You want to pass Auth information/status to the cart and payment fragments securely
- Solution
  - OAuth ad JWT
    - Use Oauth standards to pass user login information
    - Pass JWT tokens to the fragments from the login component to process with the next applicable steps
  - Headers and Cookies
    - In case of a SSR architecture, leverage HTTP headers and cookies to pass secure information



Best Practices

Payload size in Custom Events

- When creating and dispatching custom events, keep the payload size small and pass only 'need-to-know' information.
- Additional data can be fetched exclusively by the destination fragment via API calls
- This prevents tight-coupling of fragments and creates freedom to build self-contained fragments

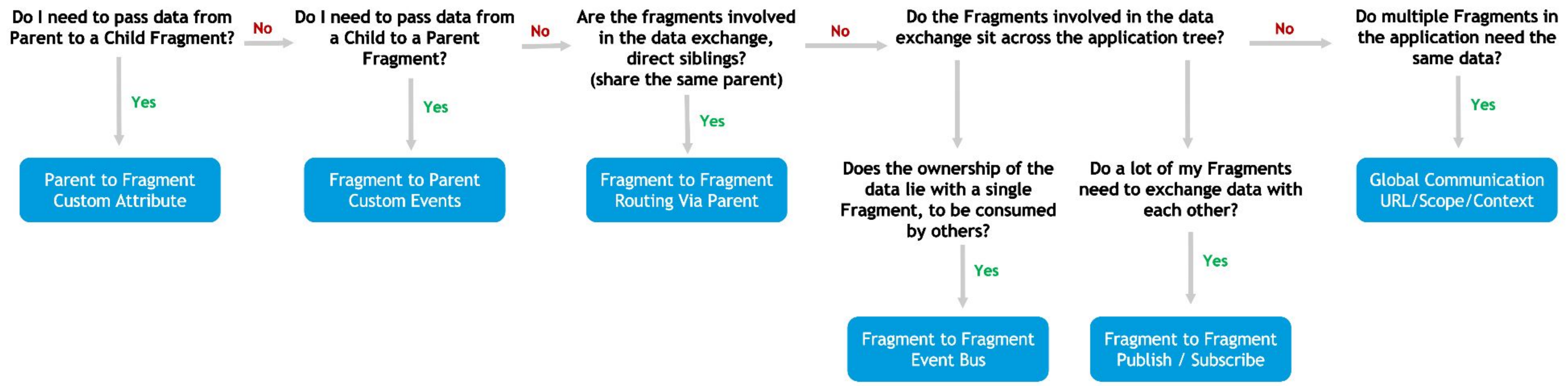
Event Bus via Custom Events

- Custom events do not bubble by default and need to be explicitly enabled
- Enable bubbling if the use-case needs a parent to intercept the event to act on it.
- Disable bubbling if the events needs to be dispatched to the immediate parent
- Avoid re-rendering of components while working with event bubbling using memo hooks in react and attribute callback handlers in JS

Broadcast Channel API

- Evaluate the need of broadcast channel based on the data published to the channel
- Avoid publishing sensitive information to the channel as it is open to subscribe to any and all fragments
- If multiple fragments need the same data with a high frequency of data updates, Publish/Subscribe is a great approach

Best Practices


Do I need to pass data from Parent to a Child Fragment?
No
Yes
Parent to Fragment Custom Attribute
Do I need to pass data from a Child to a Parent Fragment?
No
Yes
Fragment to Parent Custom Events
Are the fragments involved in the data exchange, direct siblings? (share the same parent)
No
Yes
Fragment to Fragment Routing Via Parent
Do the Fragments involved in the data exchange sit across the application tree?
No
Does the ownership of the data lie with a single Fragment, to be consumed by others?
Yes
Fragment to Fragment Event Bus
Do a lot of my Fragments need to exchange data with each other?
Yes
Fragment to Fragment Publish / Subscribe
Do multiple Fragments in the application need the same data?
Yes
Global Communication URL/Scope/Context

Questions?