

# Playful Features

**Kevin Bridges**

Applied Software Technologies

kevin@wiiBridges.com

@kevinast

<http://bit.ly/feature-u-pres>

slides, syllabus, articles, docs, and repo!



## Kevin Bridges

Applied Software Technologies

kevin@wiiBridges.com

@kevinast

- Married, Father, Grandfather
- 40 yrs in software (20 yrs consulting)
- JS (since 96, es6 since 2015)
- Retired
- Be nice to the old guy :-)



# Playful Features

<http://bit.ly/feature-u-pres>

slides, syllabus, articles, docs, and repo!



# <https://feature-u.js.org/>

SideBar:  
feature-u  
docs

The screenshot displays the official documentation for feature-u.js.org. The sidebar on the left contains the following sections:

- FEATURE-U (1.0.0)
  - Introduction
  - Benefits
  - Getting Started
- Basic Concepts
  - Usage
  - A Closer Look
  - Application Life Cycle Hooks
  - Cross Feature Communication
  - Feature Based Routes
  - Feature Enablement
  - Best Practices
- Core API
  - createFeature()
    - Built-In aspects ...
    - fassets aspect
    - appWillStart()
    - appDidStart()
    - expandWithFassets()
    - expandWithFassetsCB()
    - launchApp()
    - registerRootAppElem()
  - withFassets()
    - mapFassetsToPropsSt...
    - mapFassetsToProps()
  - Object Refs ...
    - Fassets
      - get()
      - hasFeature()
    - fassetValidations
    - Feature
    - Aspect
    - AspectContent
- Extending feature-u
- Distribution
- Why feature-u?

The main content area is titled "launchApp([aspects], features, registerRootAppElem) ⇒ Fassets". It includes the following text:

Launch an application by assembling the supplied features, driving the configuration of the frameworks in use (as orchestrated by the supplied set of pluggable Aspects).

For more information (with examples), please refer to [Launching Your Application](#).

**Please Note** this function uses named parameters.

Param	Type	Description
[aspects]	Array.<Aspect>	the set of pluggable Aspects that extend <b>feature-u</b> , integrating other frameworks to match your specific run-time stack.  When NO Aspects are supplied (an atypical case), only the very basic <b>feature-u</b> characteristics are in effect (like fassets and life-cycle hooks).
features	Array.<Feature>	the features that comprise this application.
registerRootAppElem	registerRootAppElemCB	the callback hook that registers the supplied root application element to the specific React framework used in the app.  Because this registration is accomplished by app-specific code, <b>feature-u</b> can operate in any of the react platforms, such as: <code>react web</code> , <code>react-native</code> , <code>expo</code> , etc.  Please refer to <a href="#">React Registration</a> for more details and complete examples.

**Returns:** `Fassets` - the Fassets object used in cross-feature-communication.

The next section is titled "withFassets(mapFassetsToProps, [component]) ⇒ HoC | HoF". It describes a Higher-order Component (HoC) that injects fasset props into a component. It mentions that a Higher-order Function (HoF) is created that encapsulates this "mapping knowledge".

Ultimately, this HoF must be invoked (passing `component`), which exposes the HoC (the "wrapped" Component).

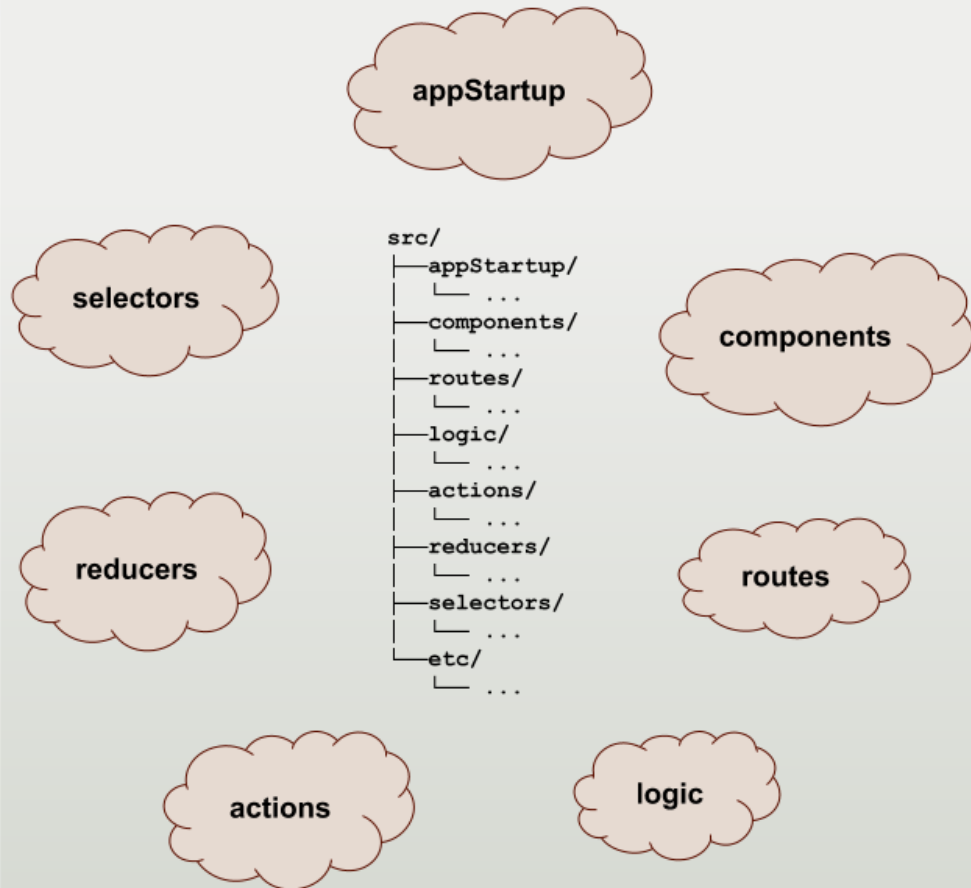
```
+ withFassetsHoF(component): HoC
```

There are two ways to use `withFassets()`:

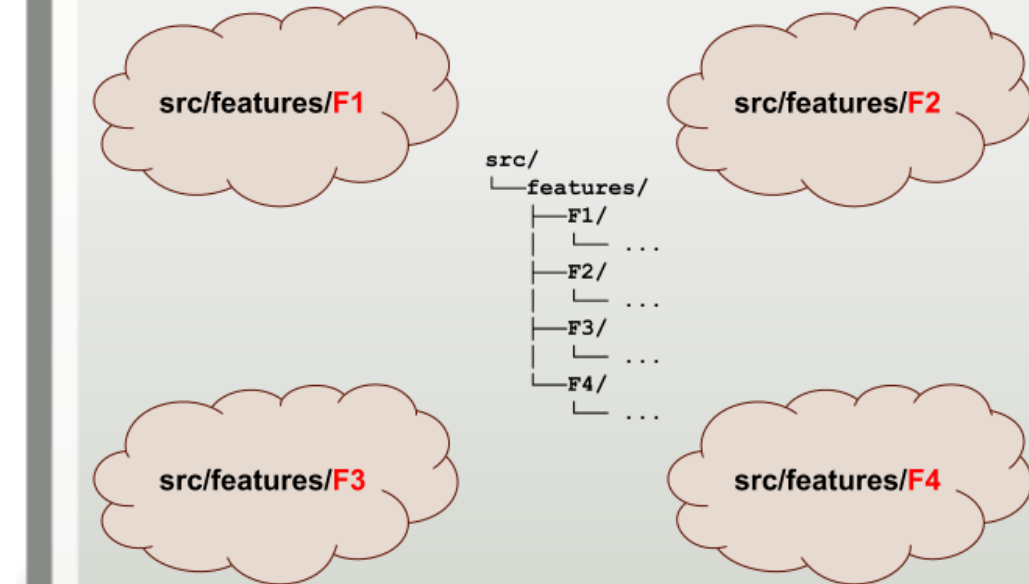
- By directly passing the `component` parameter, the HoC will be returned (internally invoking the HoF). This is the most common use case.
- By omitting the `component` parameter, the HoF will be returned. This is useful to facilitate "functional composition" (functional programming). In this case it is the client's responsibility to invoke the HoF (either directly or indirectly).

# Project Organization

## by Type

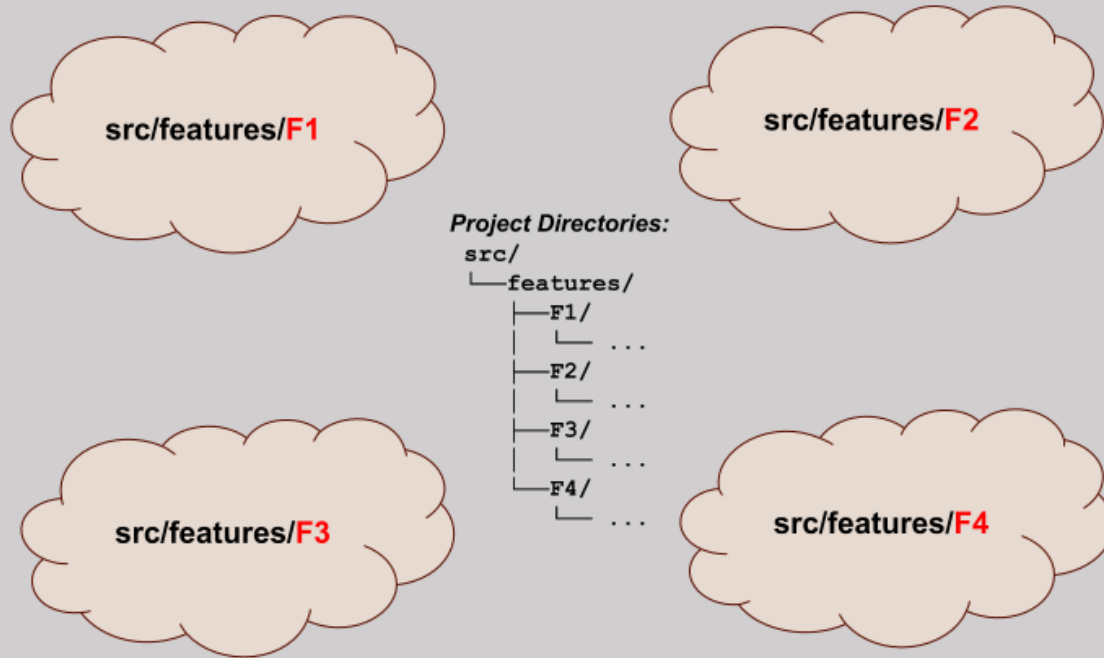


## by Feature





## Segregating Features



## Goals

- Requirements based
- Encapsulation
- Self Sufficient
- Plug-and-Play

## Hurdles

- Isolation vs. Collaboration
- Start-Up Initialization
- Framework Configuration
- UI Composition
- Feature Enablement

**In short, how do we achieve a running application from these isolated features?**

# Two Primary Tenets

needed to  
Achieve  
our  
Goal

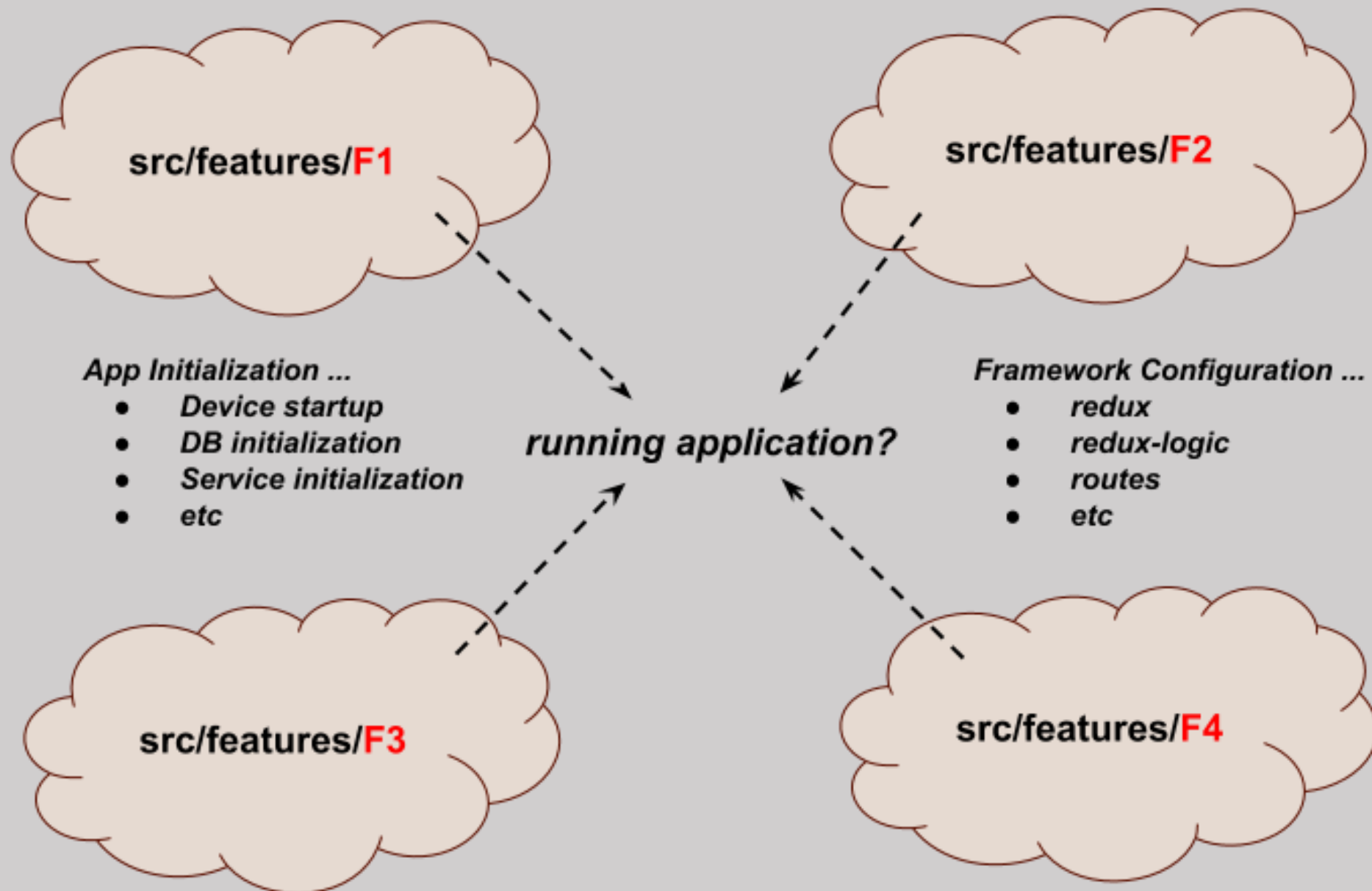


1. Feature Runtime Consolidation



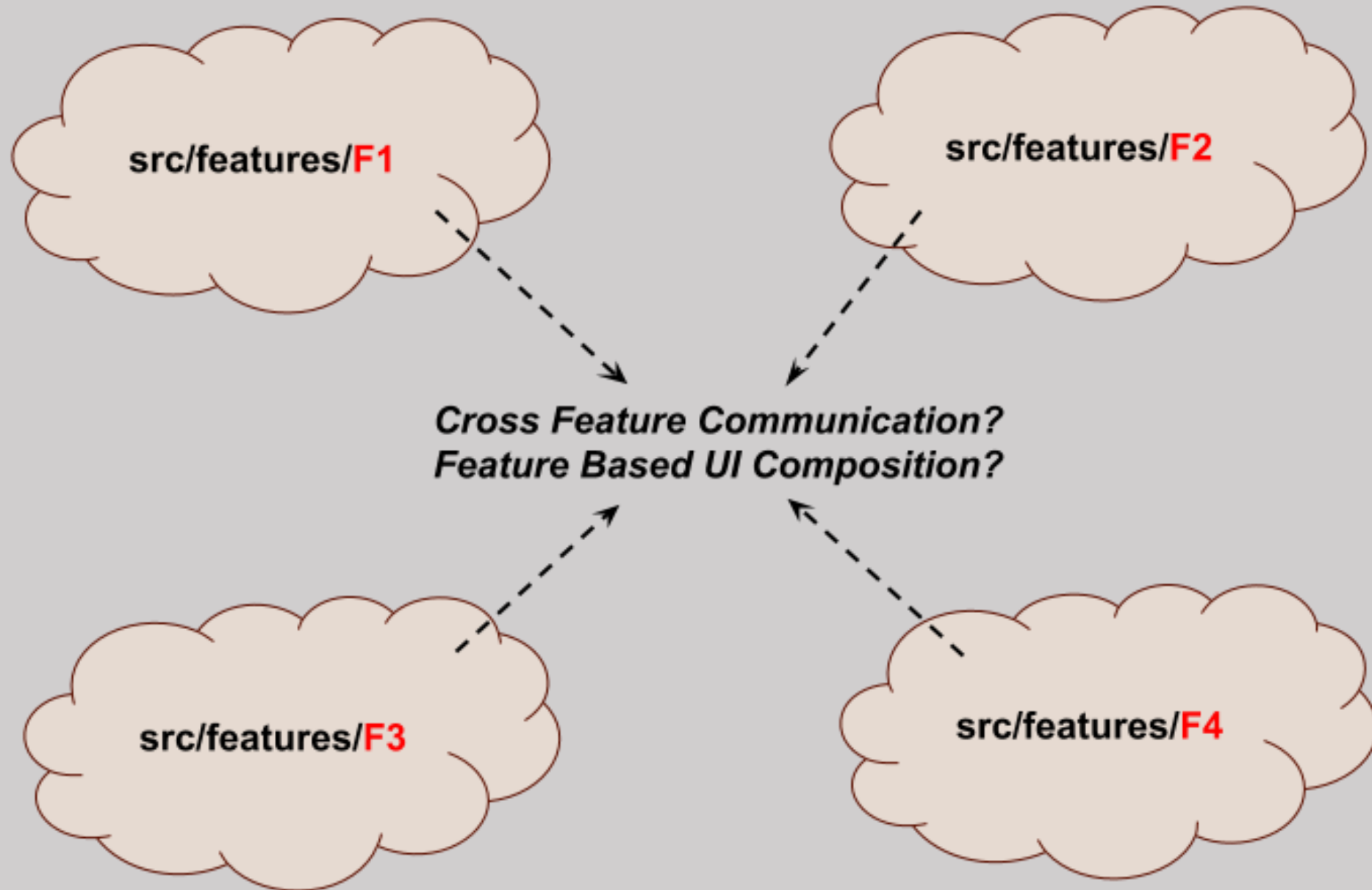
2. Feature Collaboration

# 1. Feature Runtime Consolidation

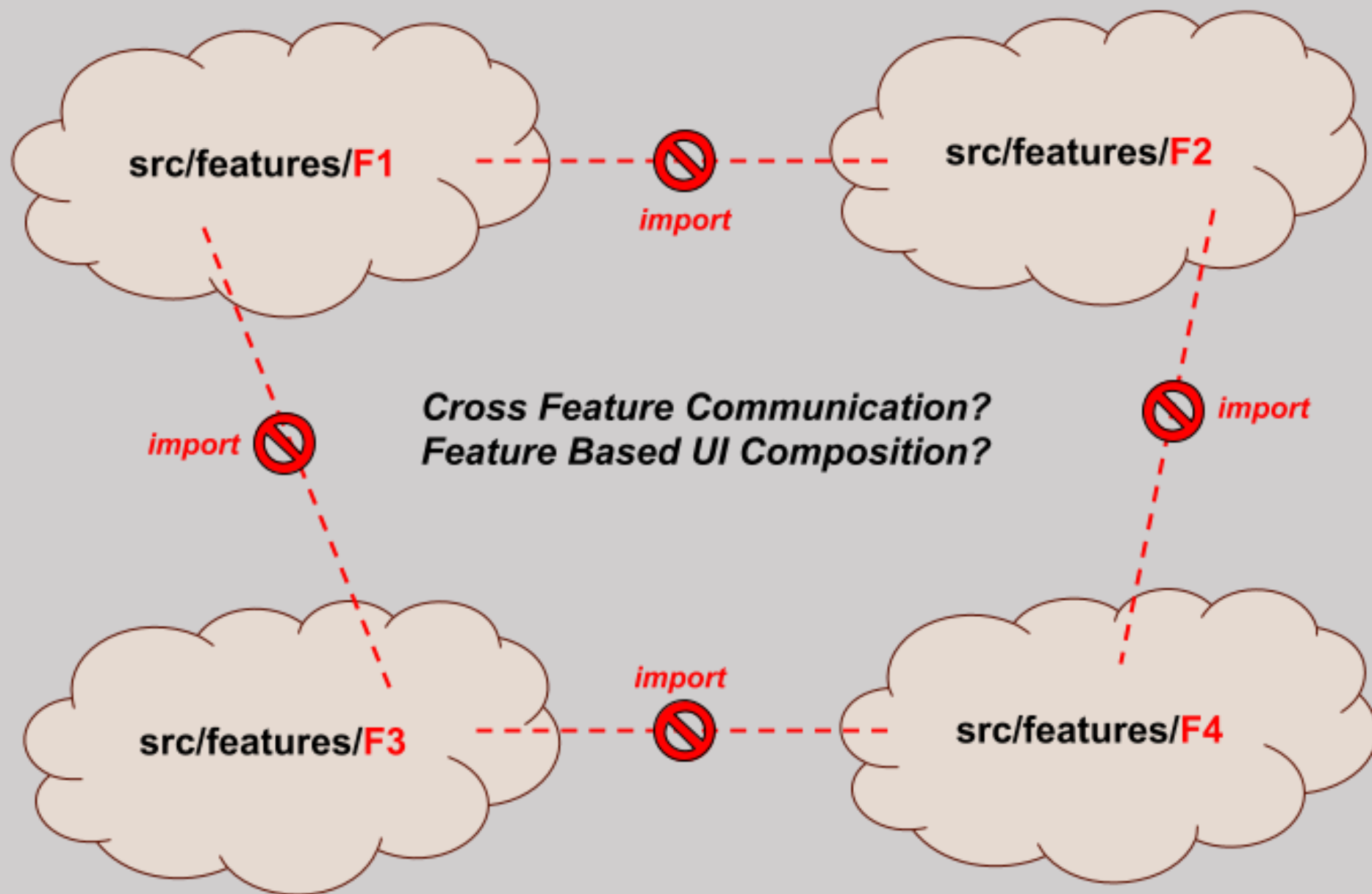




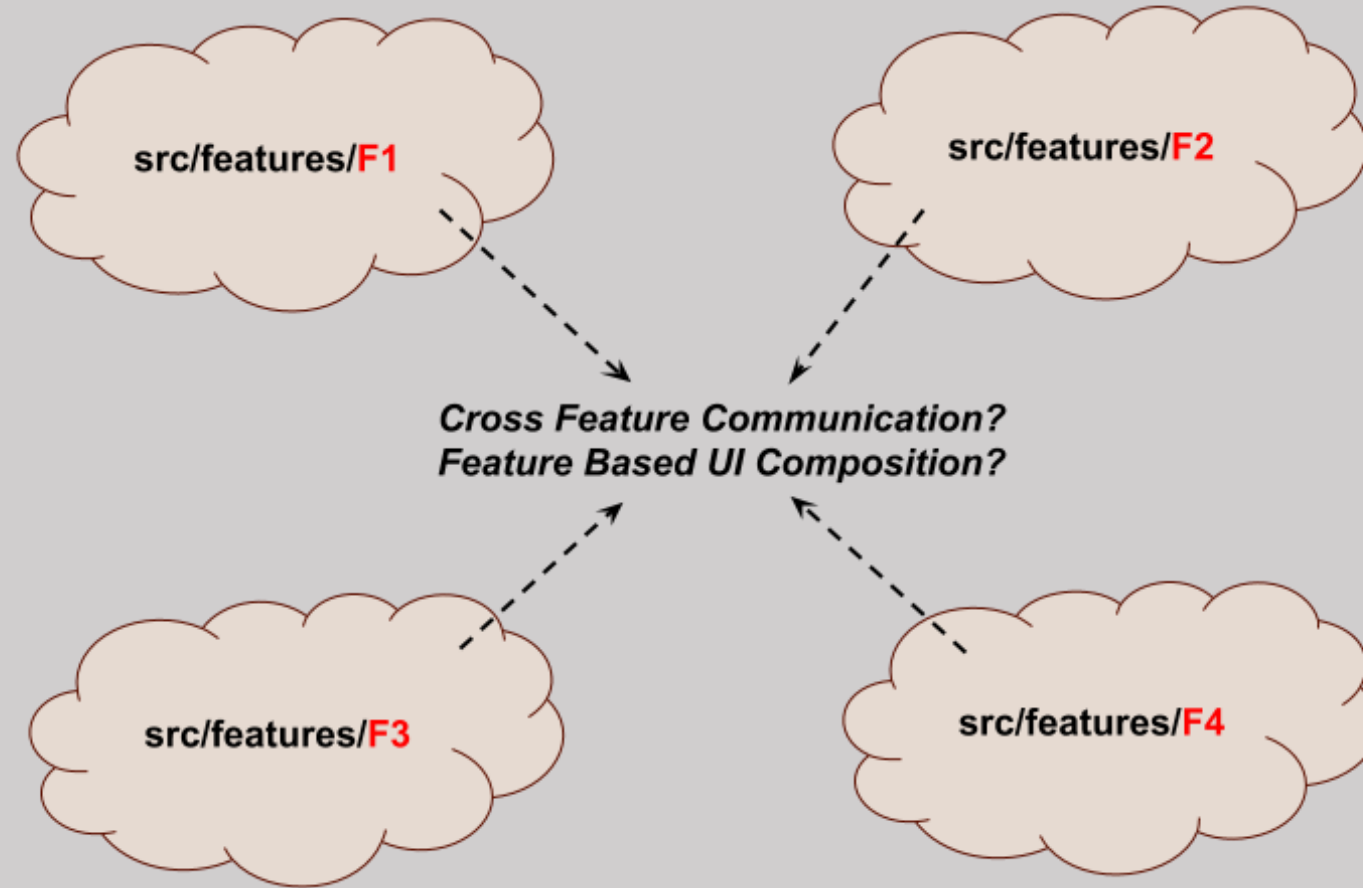
## 2. Feature Collaboration



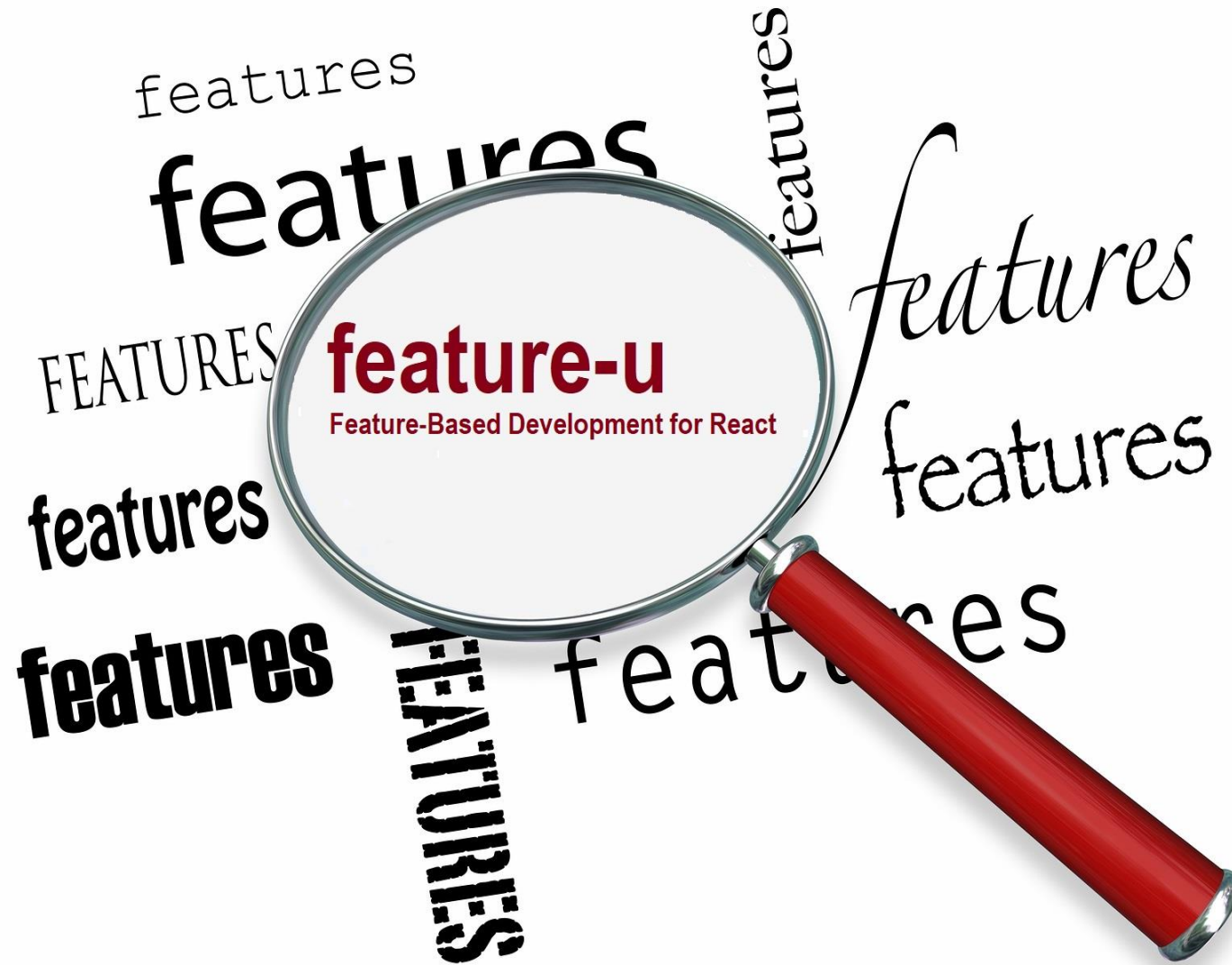
## ~~Cross Feature Imports~~



## 2. Feature Collaboration

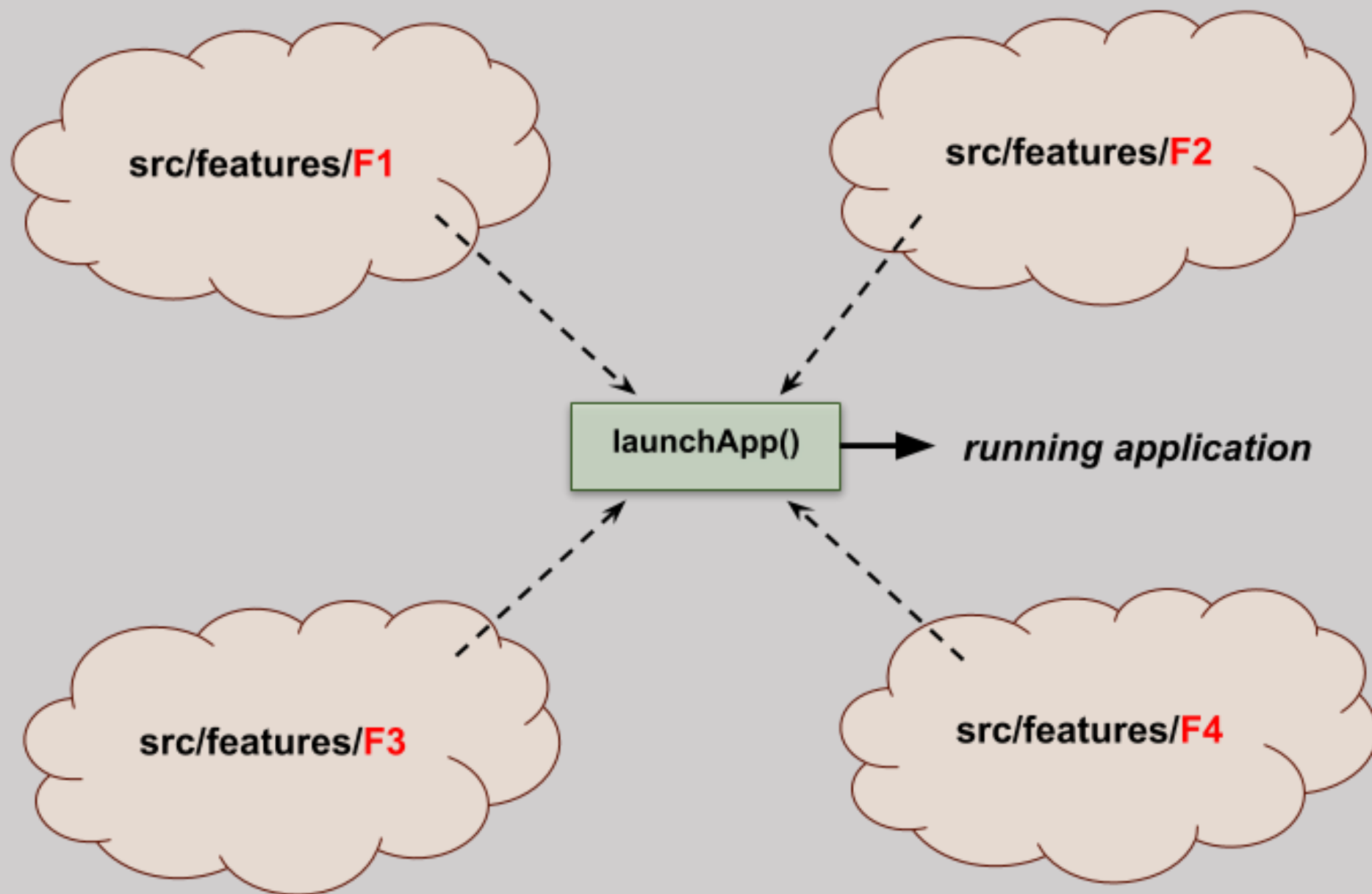


**How is Cross Feature Communication achieved  
in a way that doesn't break encapsulation?**

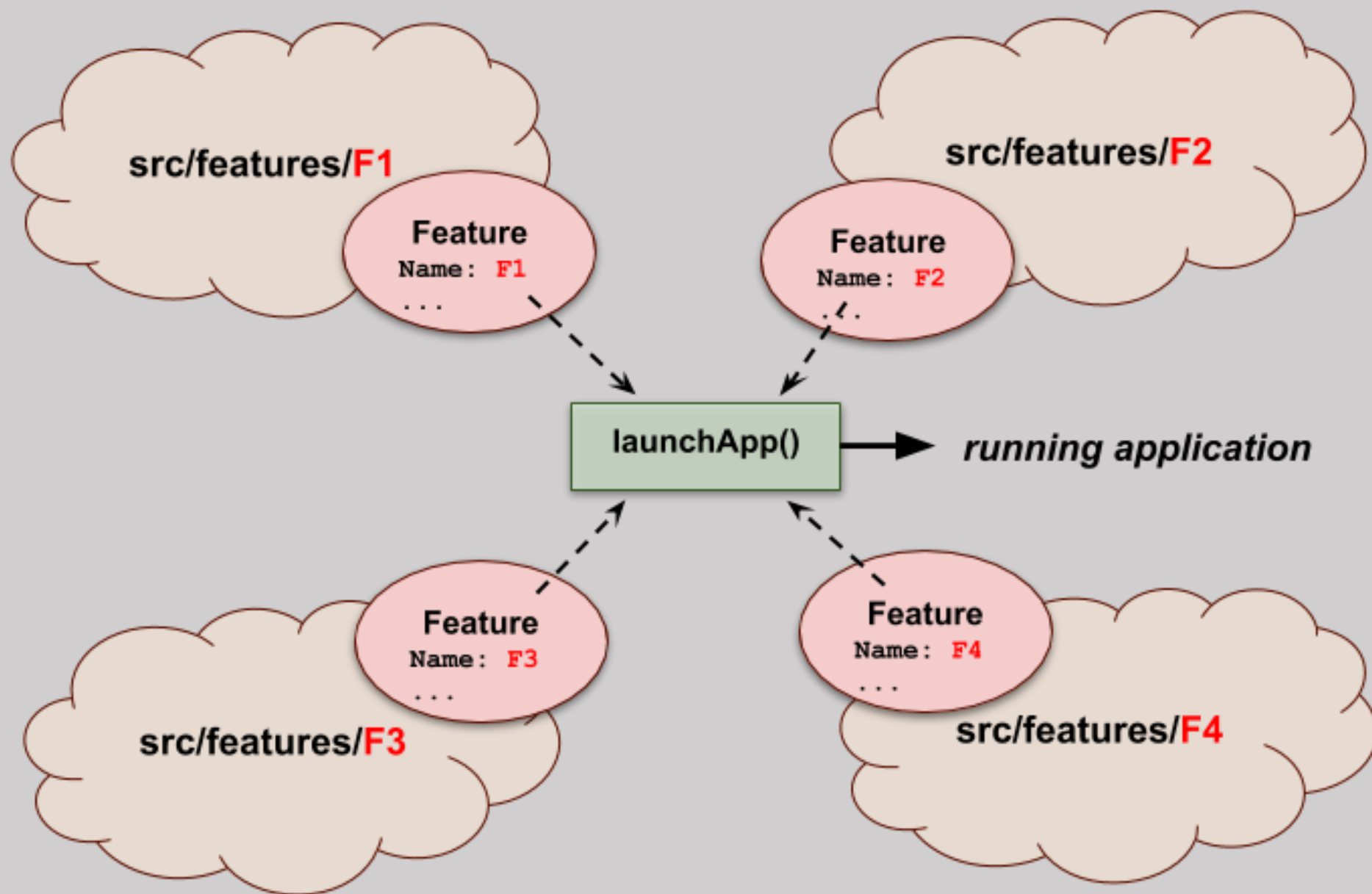


# The feature-u Solution

## Introducing: *launchApp()*



## Introducing: Feature Object





# Introducing: aspects

src/features/**F1**

## Internal Aspects <private>

- UI Components
- Routes
- State Management
  - ... actions, reducers, selectors
- Business Logic
- Startup Initialization Code
- etc

## Feature

```
name:      F1
enabled:   true
fassets:   {..public..}
reducer:   myReducers
logic:     myLogic
appWillStart() ...
```

*Aspect container*

built-in  
extended

How does  
**feature-u**  
accommodate:

# 1. Feature Runtime Consolidation

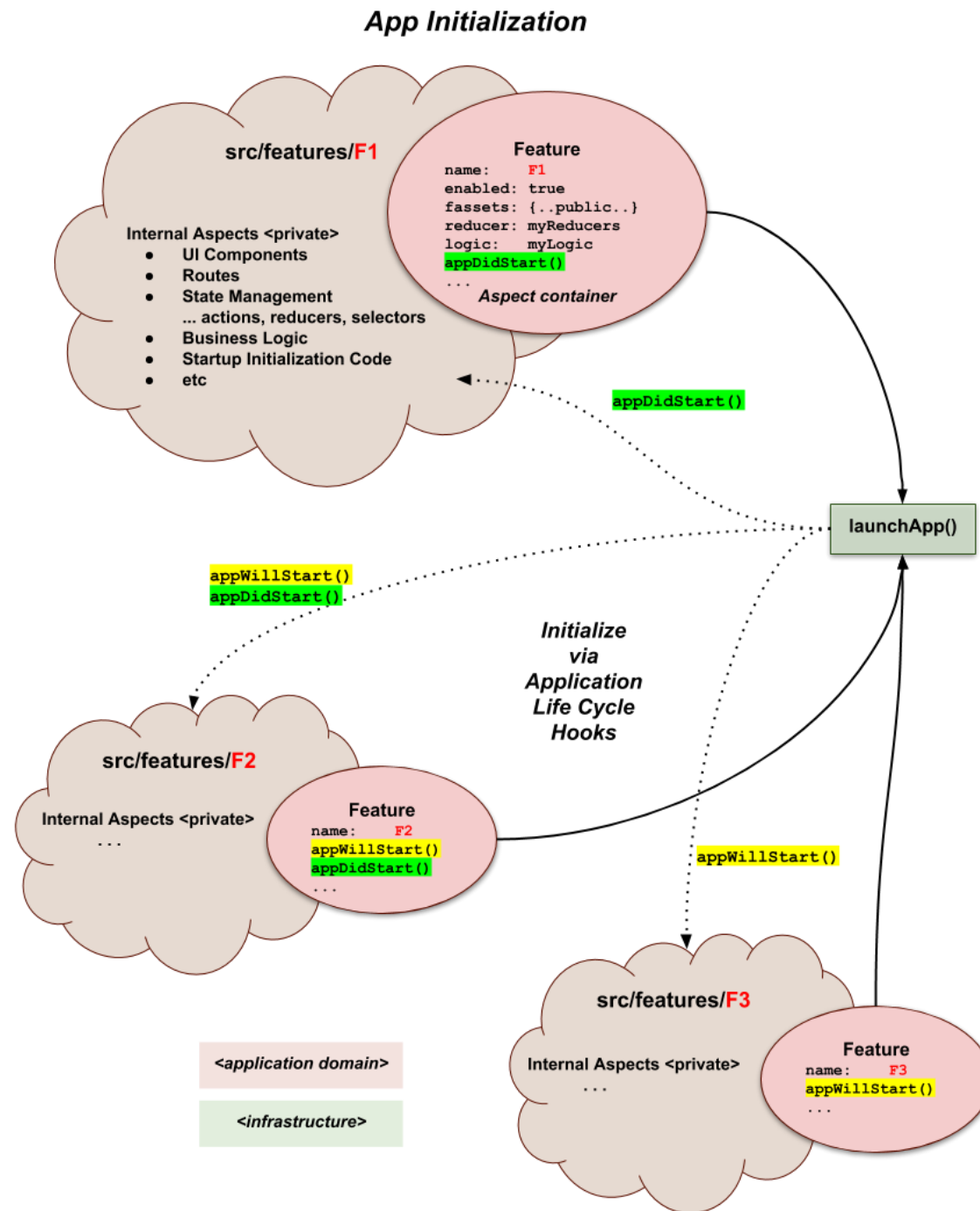


APP INITIALIZATION



FRAMEWORK  
CONFIGURATION

# Application Life Cycle Hooks



**appWillStart():**

*invoked one time at app startup time*

**appDidStart():**

*invoked one time immediately after app has started*

Extendable  
Aspect  
Plugins

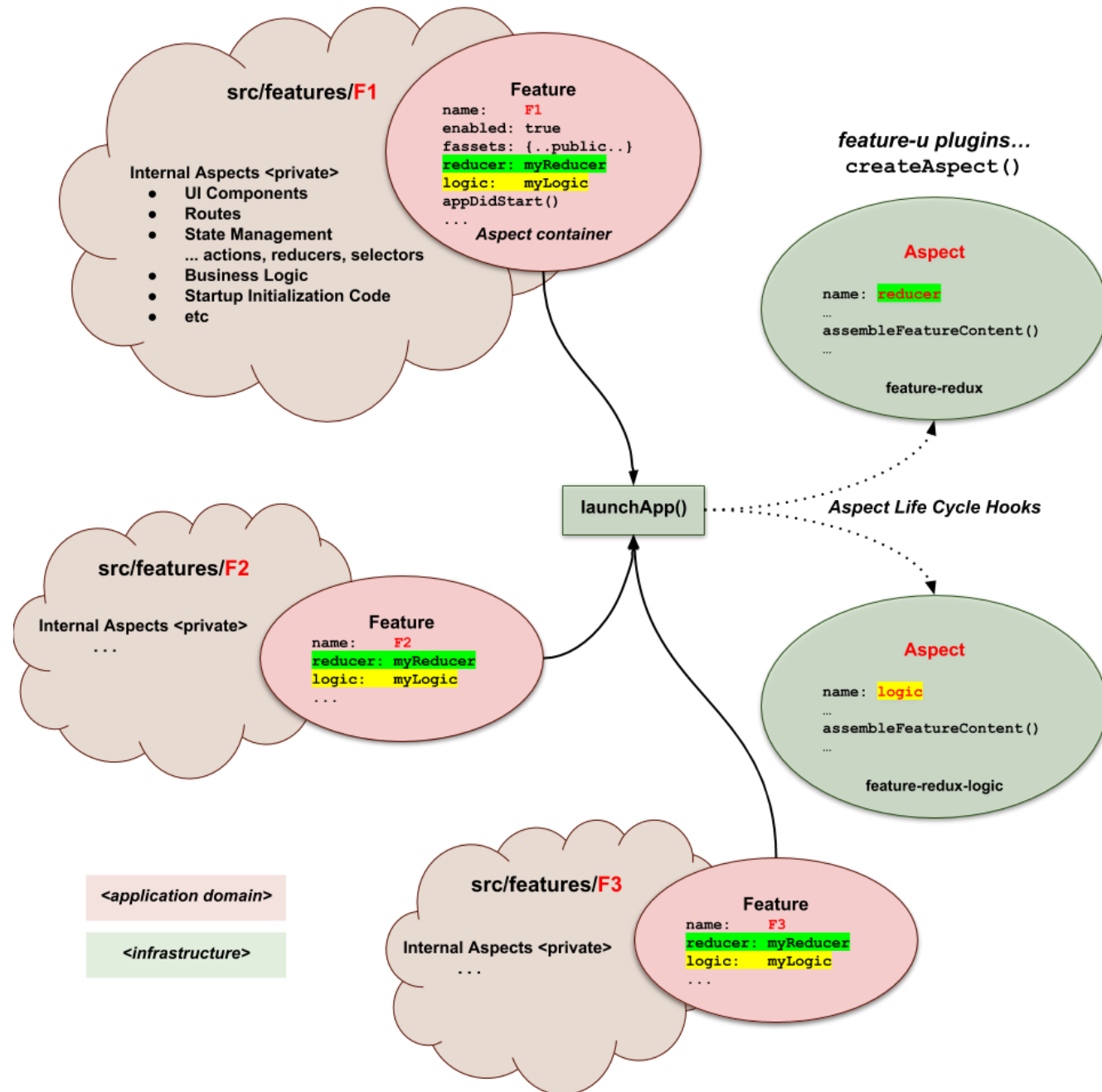


## Framework Configuration Goal

***feature-u is extendable!***

# Extendable Aspect Plugins

## Framework Configuration



1. Feature  
Runtime  
Consolidation





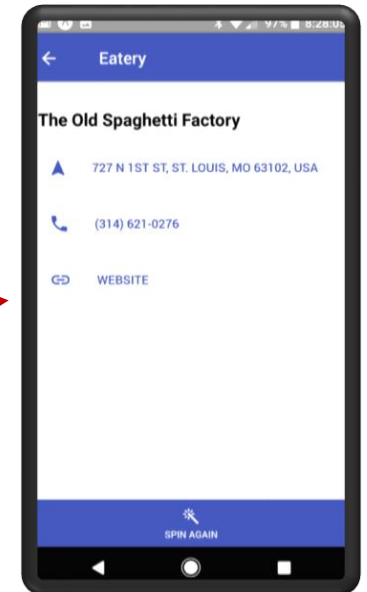
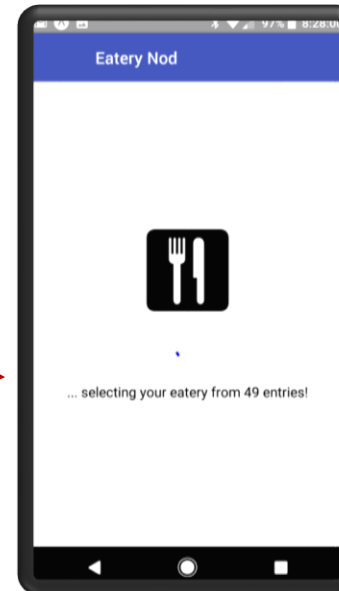
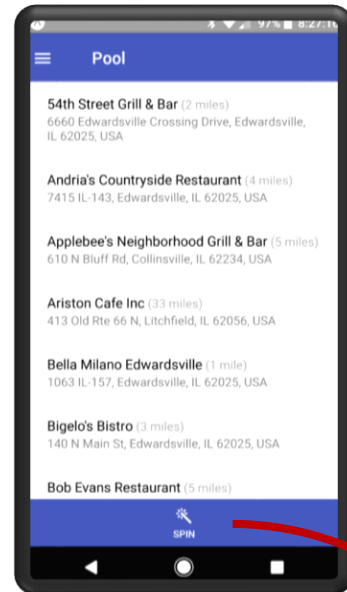
# *eatery-nod*



<https://github.com/KevinAst/eatery-nod>



## 1. Feature Runtime Consolidation



```

//
// ... now module - launches app via launchApp()
//
feature {
    index-3s    ... accumulate/promote all app feature objects
    //
    // ... the app's authorization feature
    //
    authorize-3s
    feature-3s
    //
    // ... expose aspects of interest to feature-u
    //
    index-3s
    feature-3s
    route-3s
    //
    // ... expose features-3s
    //
    state-3s
    //
    // ...
    //
    significance-3s
    significanceIdValues-3s
    //
    // ...
    //
    currentViewId-3s    ... new features
    //
    // ... feature to initialize the device
    //
    device-3s
    actions-3s
    app-3s
    appIdentifier-3s
    appIdentifierId-3s
    //
    // ... expose aspects of interest to feature-u
    //
    index-3s
    feature-3s
    route-3s
    //
    // ...
    //
    test-3s
    //
    // ...
    //
    platformFingerprint.android-3s
    platformFingerprint.lin-3s
    //
    // ... more features
    //
    dictionary-3s
    natural-3s
    filename-3s
    languages-3s
    username-3s
    //
    // ...
    //
    util-3s    ... common utilities used across all features
    //
    // ...
    //

```

**Our App is running from**

**isolated/independent**

**features ...**

- **App Initialization**
- **Framework Configuration**

[illegible]

```
export default launchApp({
  aspects: appAspects(),
  features,
  registerRootAppFile(rootAppFile) {
    Expo.registerRootComponent(() => rootAppFile);
  }
});
```

```
import React      from 'react';
import Platform    from './lib/platformSetup';
import Notify      from './../util/notify';

//
// An app-level life-cycle hook, initializing our feature by:
//   - performing platform-specific setup (iOS/Android)
//   - inject our notify utility into the DOM
//
export default function appLevelStart({assets, curAppAppId}) {
  // platform-specific setup (iOS/Android)
  platformSetup();

  // Initialize notify utility, by injecting it in our App root
  return [React.Children.toArray(curAppAppId), notify key="notify"]
}
```

```
import (createFeature) from 'feature-u';
import {initFirestore} from './init/initFirestore';

/**
 * The "Firestore" feature initializes the google firestore service,
 * and provides a placeholder for future API abstractions.
 */
export default createFeature({
  name: 'Firestore',

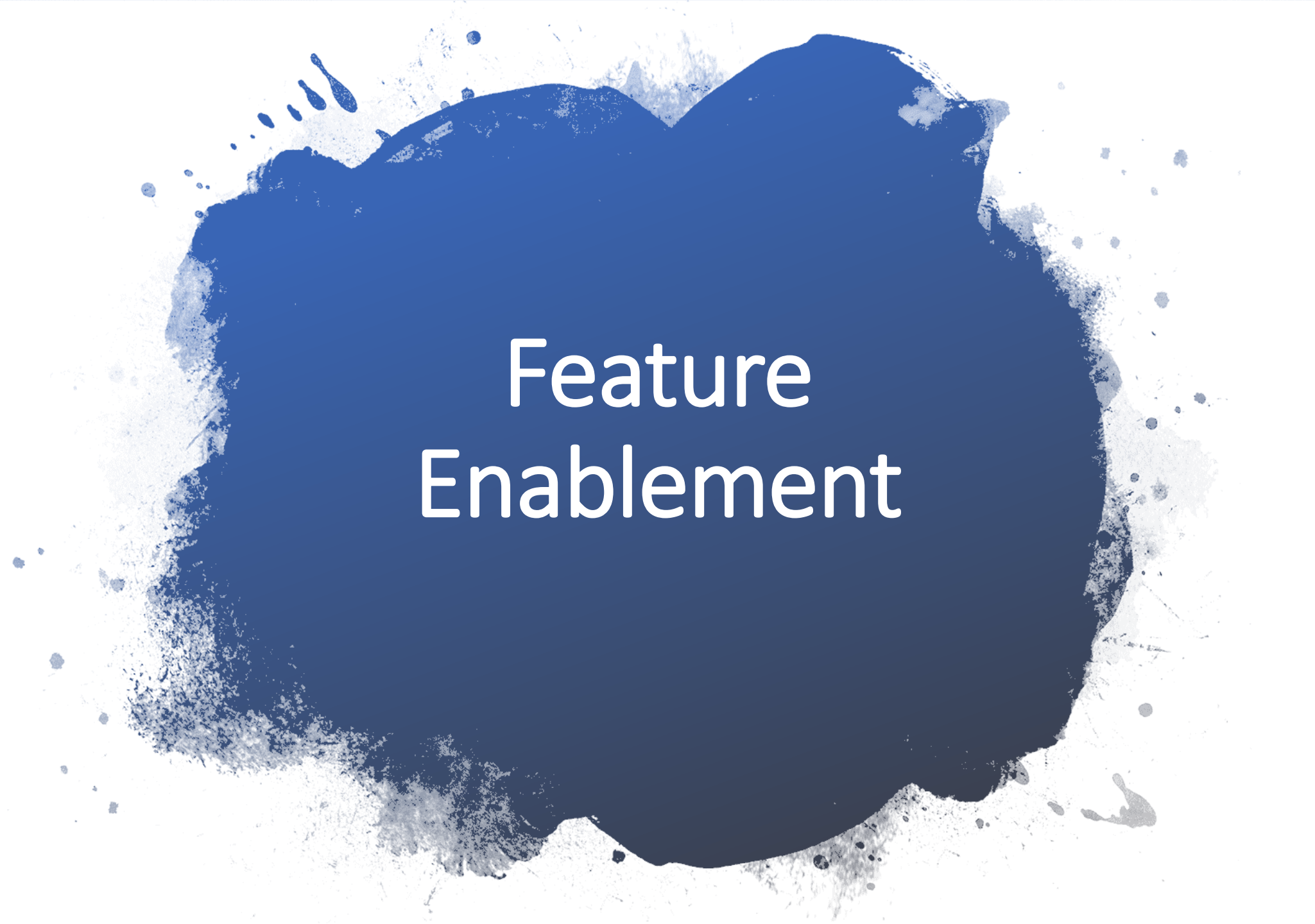
  appWillStart: ([assets, currentUser]) {
    initFirestore(); // Initialize Firestore
  },
});
```

```
import React      from 'react';
import {Drawer}   from 'native-base';
import Sidebar    from './components/Sidebar';
import classDrawer from './components/classDrawer';

//**
// Inject our Drawer/Sidebar component at the root of our app
//**
export default function App(props) {
  return (
    <Drawer ref={ref} > registerDrawer(ref)
      <content> <Sidebar/>
      <close> <classDrawer/>
    </Drawer>
  );
}
```

```
import actions from './actions';

/*
 * An app-level life-cycle hook that dispatches our bootstrap action
 * that gets the ball rolling!
 */
export default function appOnStart({assets, appState, dispatch}) {
  dispatch(actions.bootstrap());
}
```



# Feature Enablement

## Feature Enablement

# *Feature Enablement*

- by default all Features are active
- can be disabled via Feature.enabled built-in aspect

src/feature/sandbox/feature.js

```
export default createFeature({  
  name: 'sandbox',  
  enabled: inDevelopmentMode(),  
  ... snip snip  
});
```

**Made possible because feature-u is  
in control of App Startup**



How does  
feature-u  
accommodate:

## 2. Feature Collaboration



Cross Feature  
Communication

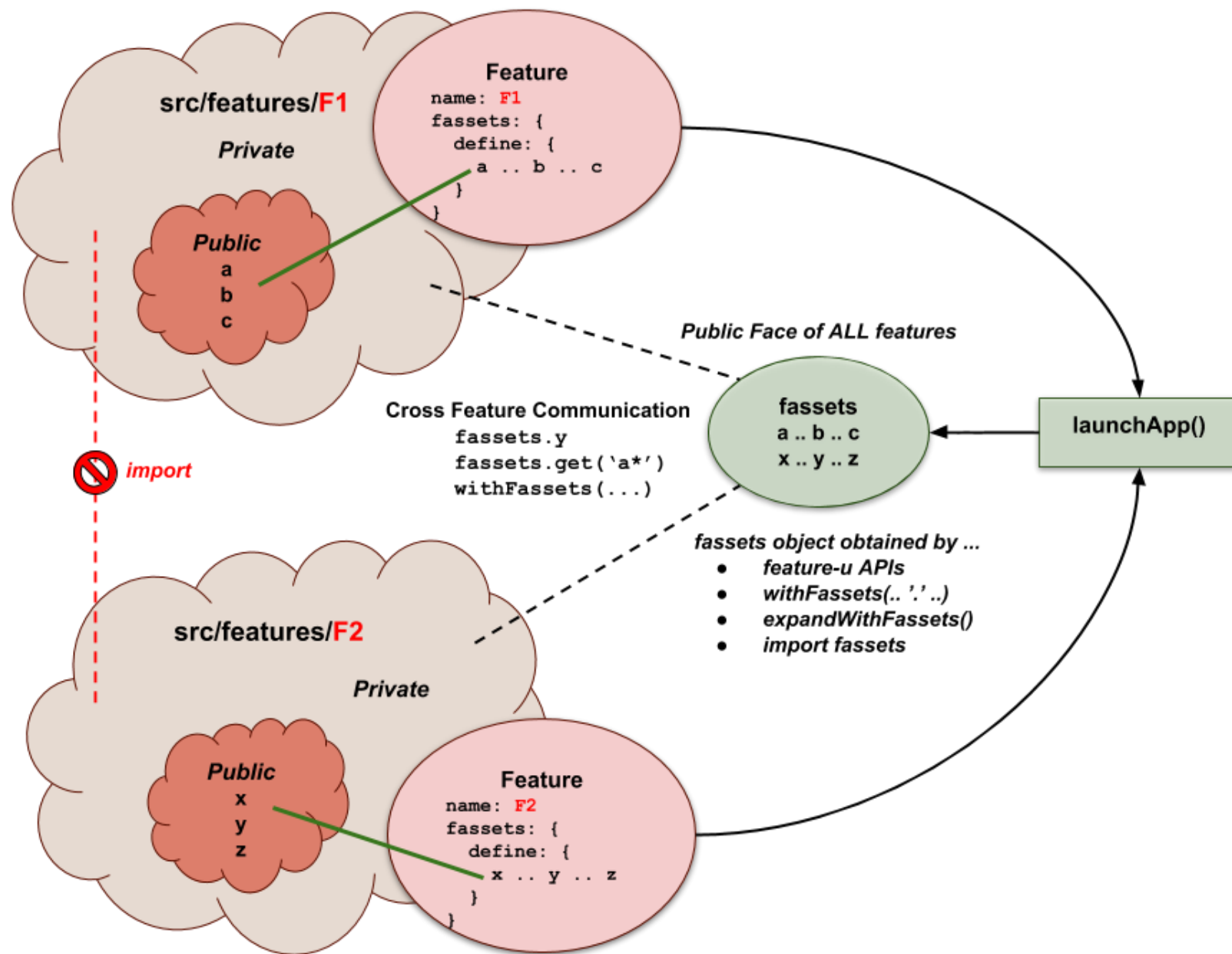


Feature Based  
UI Composition

# Cross Feature Communication

fassets

## Cross Feature Communication





# Cross Feature Communication

## fassets

*code snippet ...*

defining fassets

```
export default createFeature({  
  
  name:      'featureA',  
  
  fassets: {  
    define: {  
      'actions.openView': actions.view.open,      // openView(viewName): Action  
      'sel.currentView':  selector.currentView,   // currentView(appState): viewName  
      'sel.isDeviceReady': selector.isDeviceReady, // isDeviceReady(appState): boolean  
    },  
  },  
  
  ...  
});
```

using fassets

```
if (fassets.sel.isDeviceReady(appState)) {  
  ...  
}
```

**NOTE:** This uses a push philosophy

# Feature Based UI Composition withFassets()


## code snippet ...

### defining logo

```
export default createFeature({  
  
  name: 'common',  
  
  fassets: {  
    define: {  
      'company.logo': () => , // a react component  
    },  
  },  
  
  ...  
});
```

### injecting fasset component properties

```
function MyComponent({Logo}) {  
  return (  
    <div>  
      <Logo/>  
    </div>  
    ... snip snip  
  );  
}  
  
export default withFassets({  
  component: MyComponent,  
  mapFassetsToProps: {  
    Logo: 'company.logo',  
  }  
});
```



Feature Based  
UI Composition

Resource  
Contracts

# UI Composition can be a "contract"

- Supported by additional **fasset** directives
  - **fassets.use**: specify a series of injection needs
  - **fassets.defineUse**: supply this content
- **NOTE: This uses a pull philosophy**
- **Wildcards (\*)** provide additional dynamics
  - allowing content to be **injected autonomously**

# Feature Based UI Composition

## Resource Contracts

src/features/main/feature.js

```
createFeature({  
  name: 'main',  
  
  fassets: {  
    use: [  
      'MainPage.*.link',  
    ],  
  },  
});
```

*code snippet ...*

src/features/main/comp/MainPage.js

```
function MainPage({mainLinks}) {  
  return (  
    <div> { /* links section */}  
    {mainLinks.map( (MainLink, indx) => <MainLink key={indx}/>)}  
    </div>  
  );  
}  
  
export default withFassets({  
  component: MainPage,  
  mapFassetsToProps: {  
    mainLinks: 'MainPage.*.link',  
  },  
});
```

**We have switched to a pull philosophy  
with autonomous injection!!**

src/features/cart/feature.js

```
createFeature({  
  name: 'cart',  
  
  fassets: {  
    defineUse: {  
      'MainPage.cart.link': () => <Link to="/cart">Cart</Link>,  
    },  
  },  
});
```

src/features/search/feature.js

```
createFeature({  
  name: 'search',  
  
  fassets: {  
    defineUse: {  
      'MainPage.search.link': () => <Link to="/search">Search</Link>,  
    },  
  },  
});
```

UI Composition

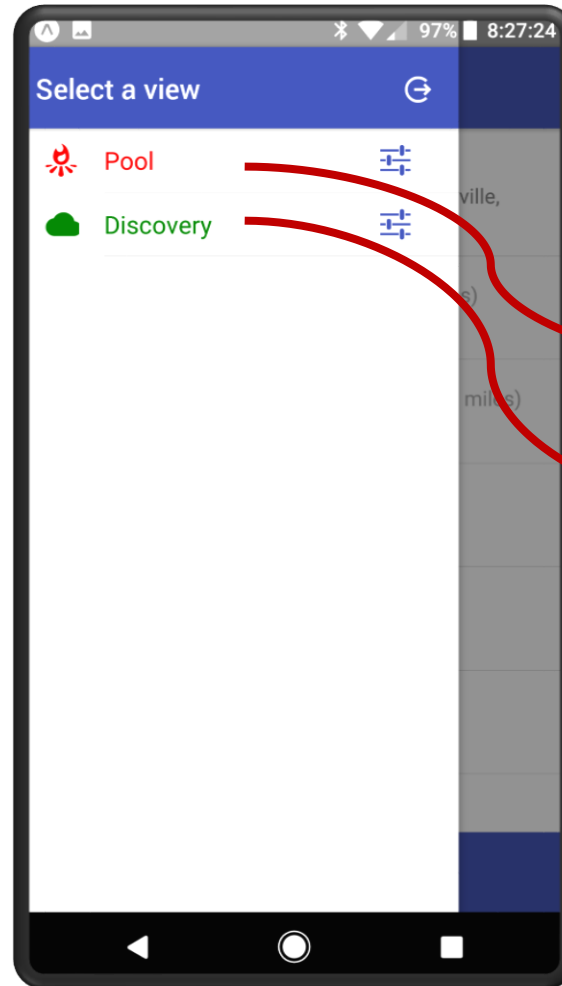
Resource  
Contracts



# *eatery-nod*



<https://github.com/KevinAst/eatery-nod>



*leftNav* feature

*eatery* screens

*discovery* screens

# A/B Testing

change a feature to a new version

or swap out to a mock service

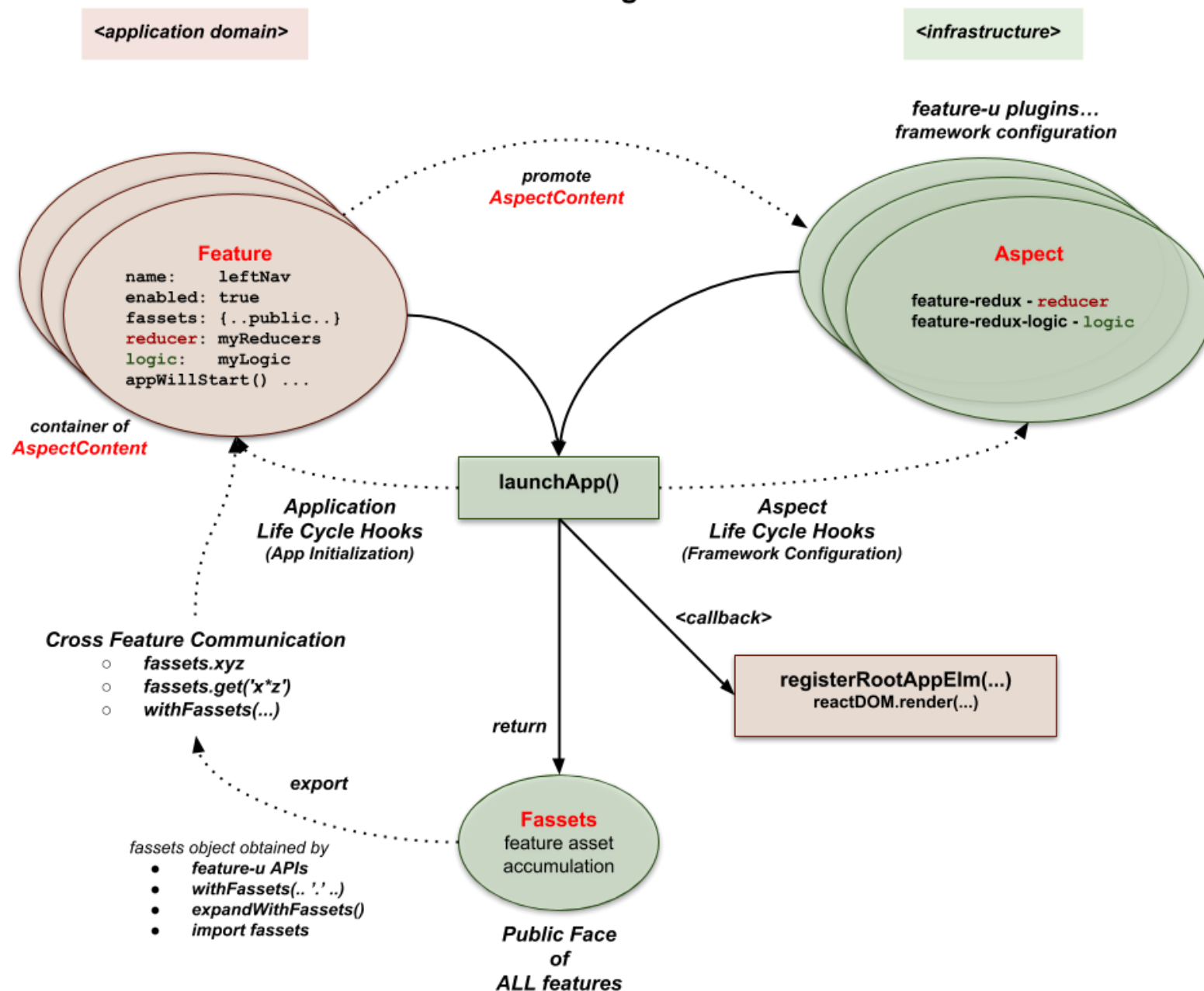
or turn on a new feature in it's entirety





# In Summary

## feature-u Context Diagram



A final word  
about  
Feature Based  
Development

Features should mirror requirements

Features are a Higher Level Abstraction

- ***because** they contain other programming paradigms*
- *features are more about a “logical packaging”  
so they **plug-and-play***

What does this mean in terms of **feature-u**?

**Question:** "*How does **feature-u** impact my design constructs?*"

**Answer:** It doesn't!

**feature-u** is NON Intrusive!

- you employ same constructs and styles
- you use same frameworks in the same way
- *only diff:* your scope is smaller (*i.e. a feature*)

**feature-u** *frees you up to focus your attention  
on the "business end" of your features!*

