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

Unleash the power

of

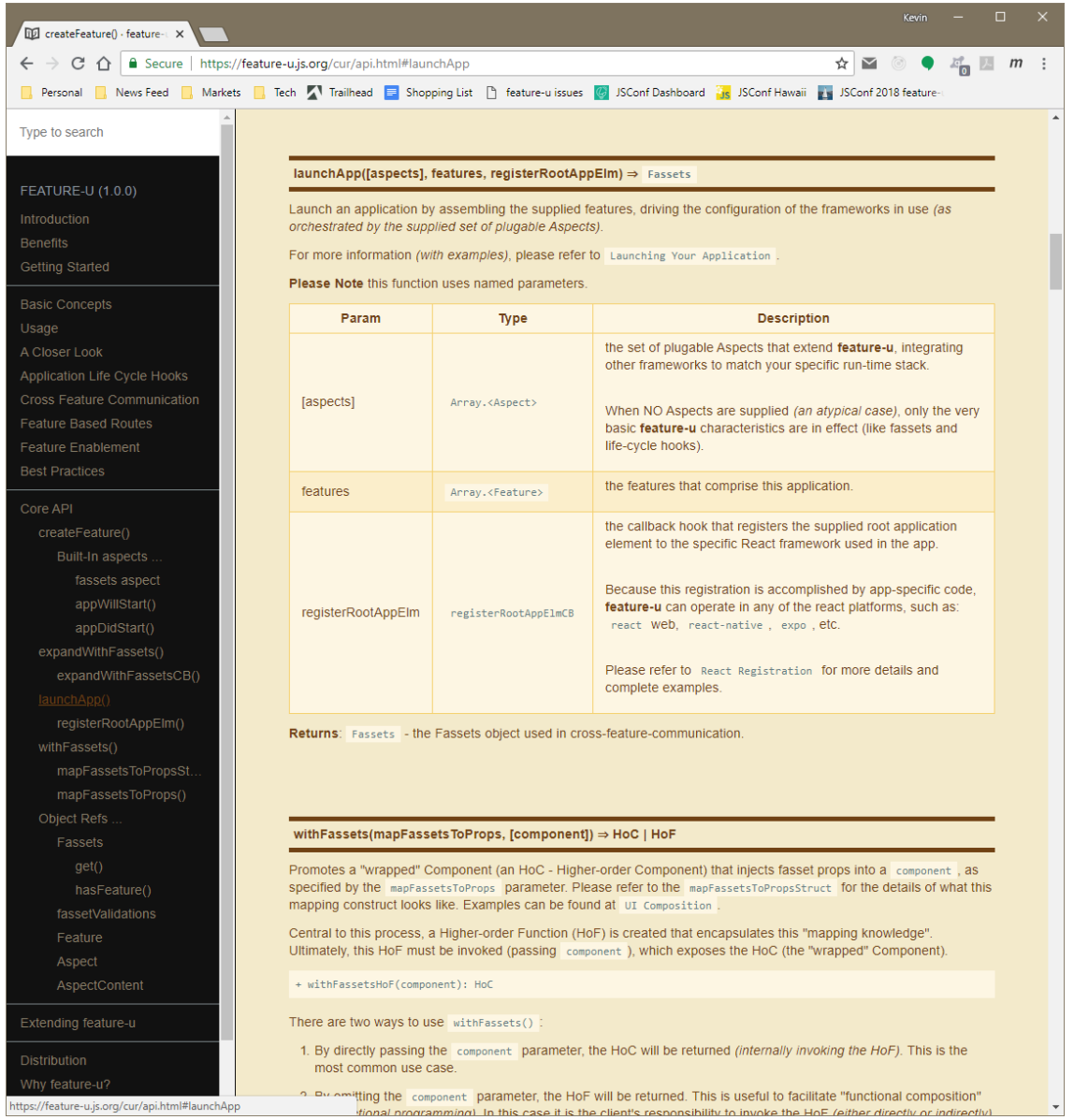
feature-based development

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

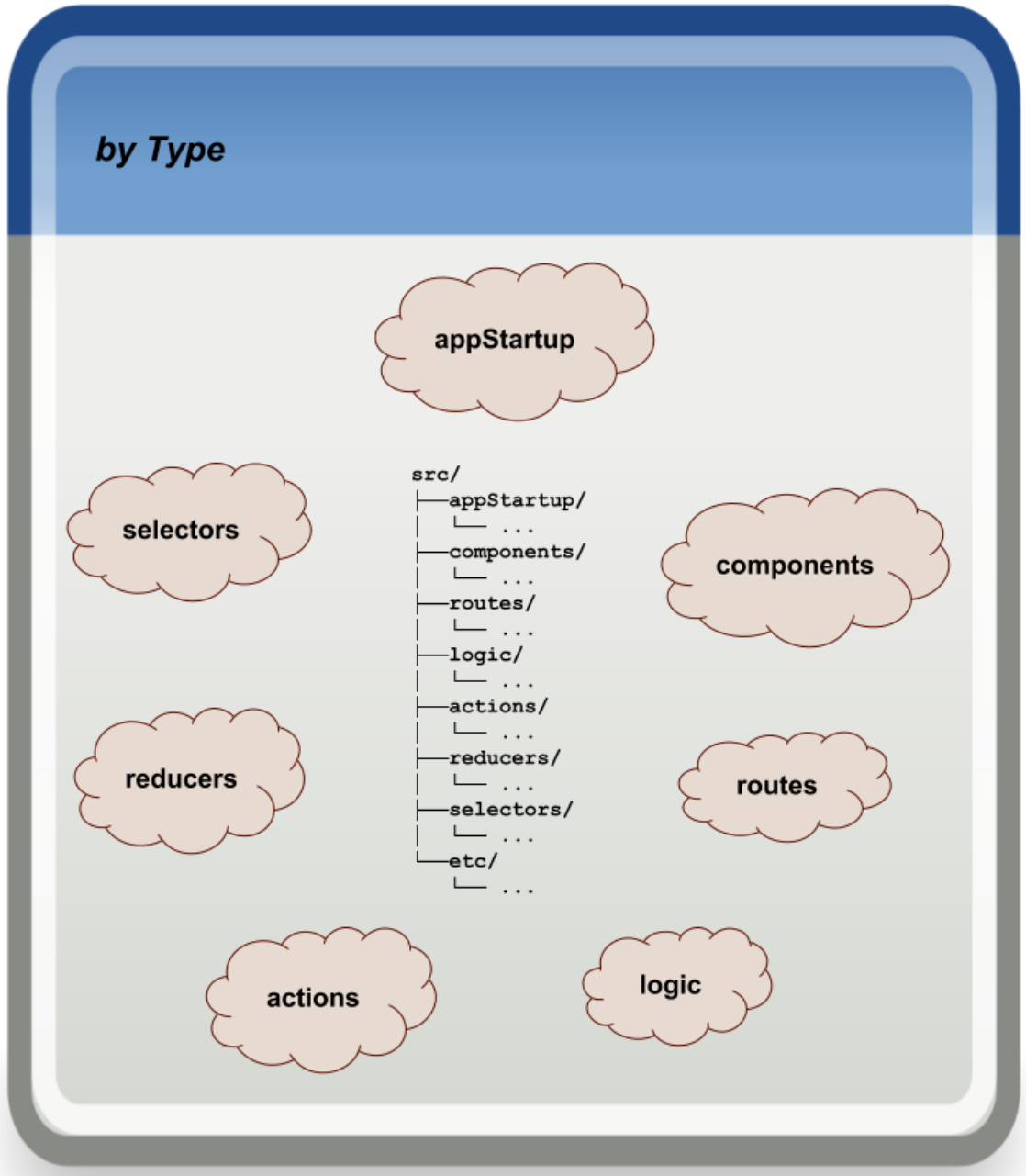
slides, syllabus, articles, docs, and repo!

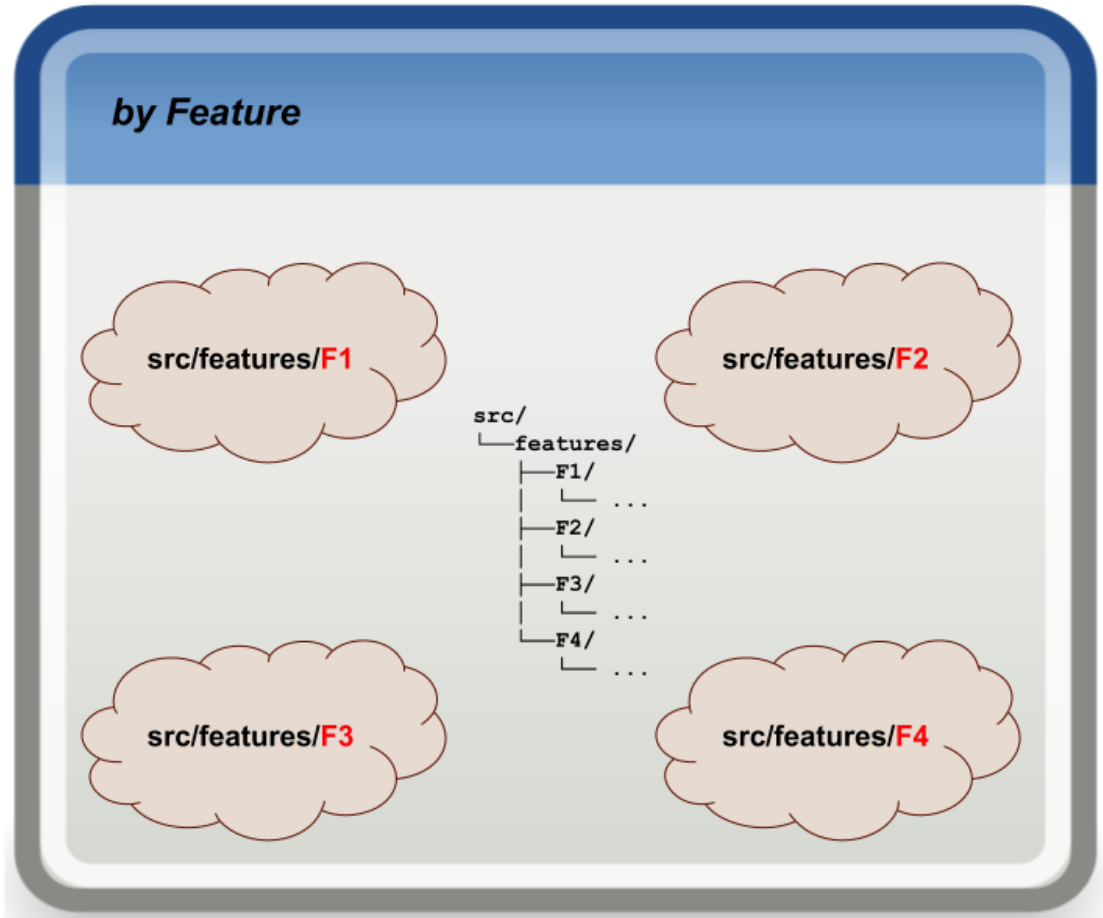
https://feature-u.js.org/

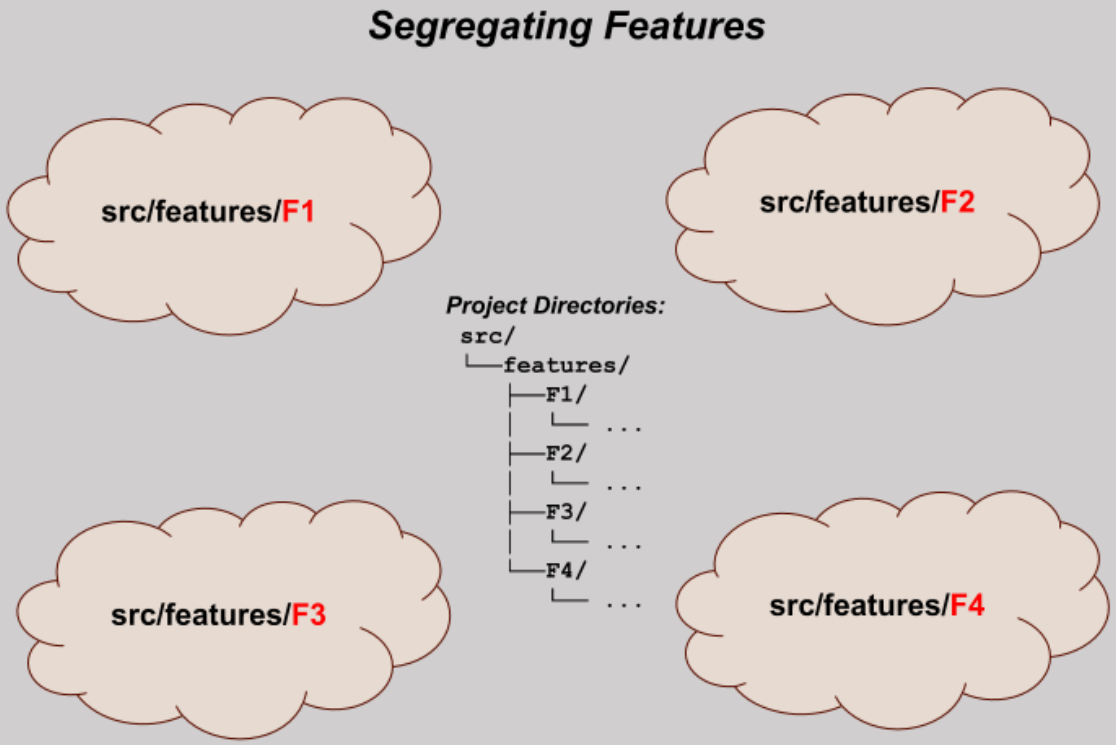
SideBar: feature-u docs



Project Organization







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?

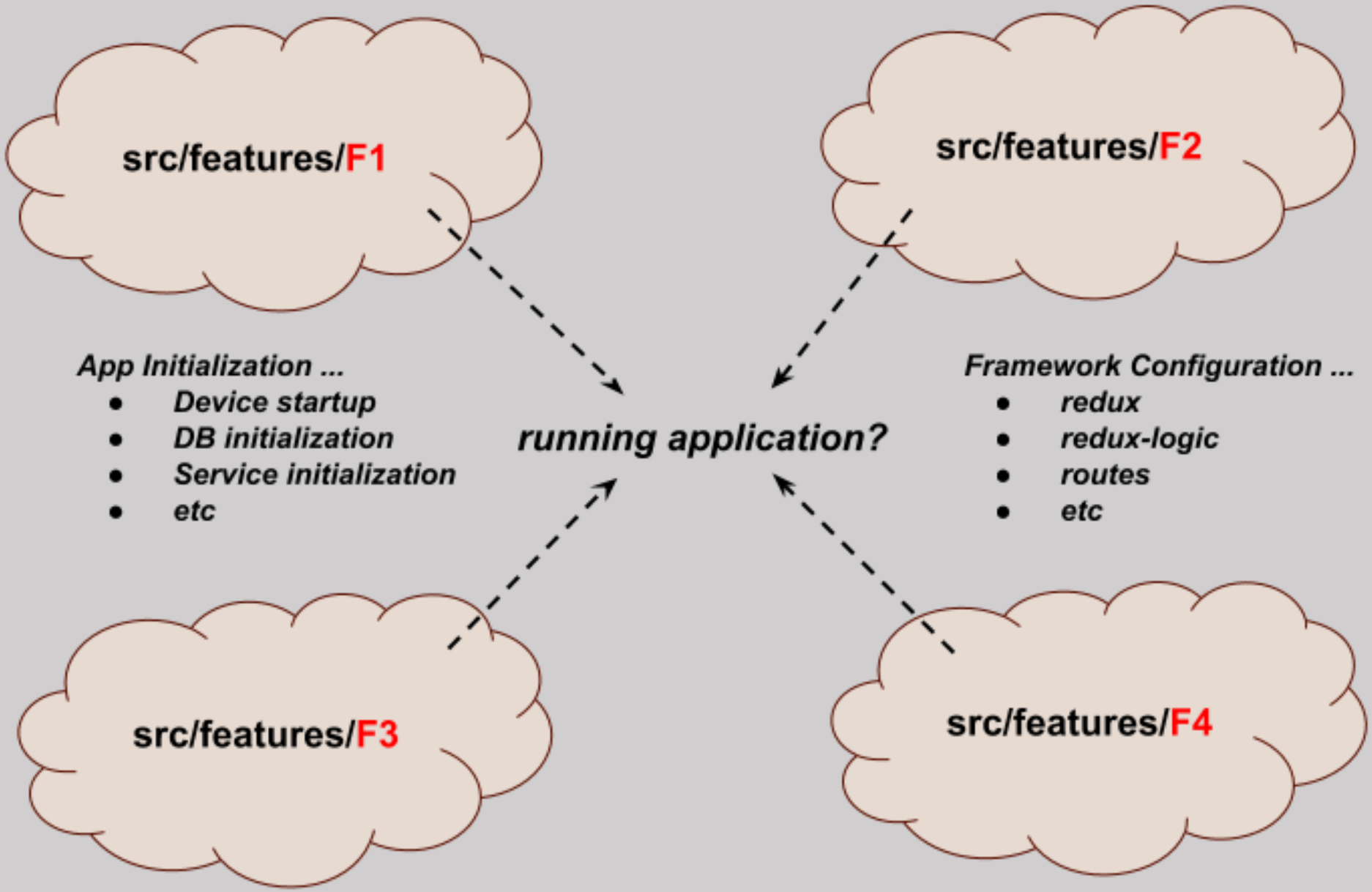
Two Primary Tenets

needed to Achieve our Goal

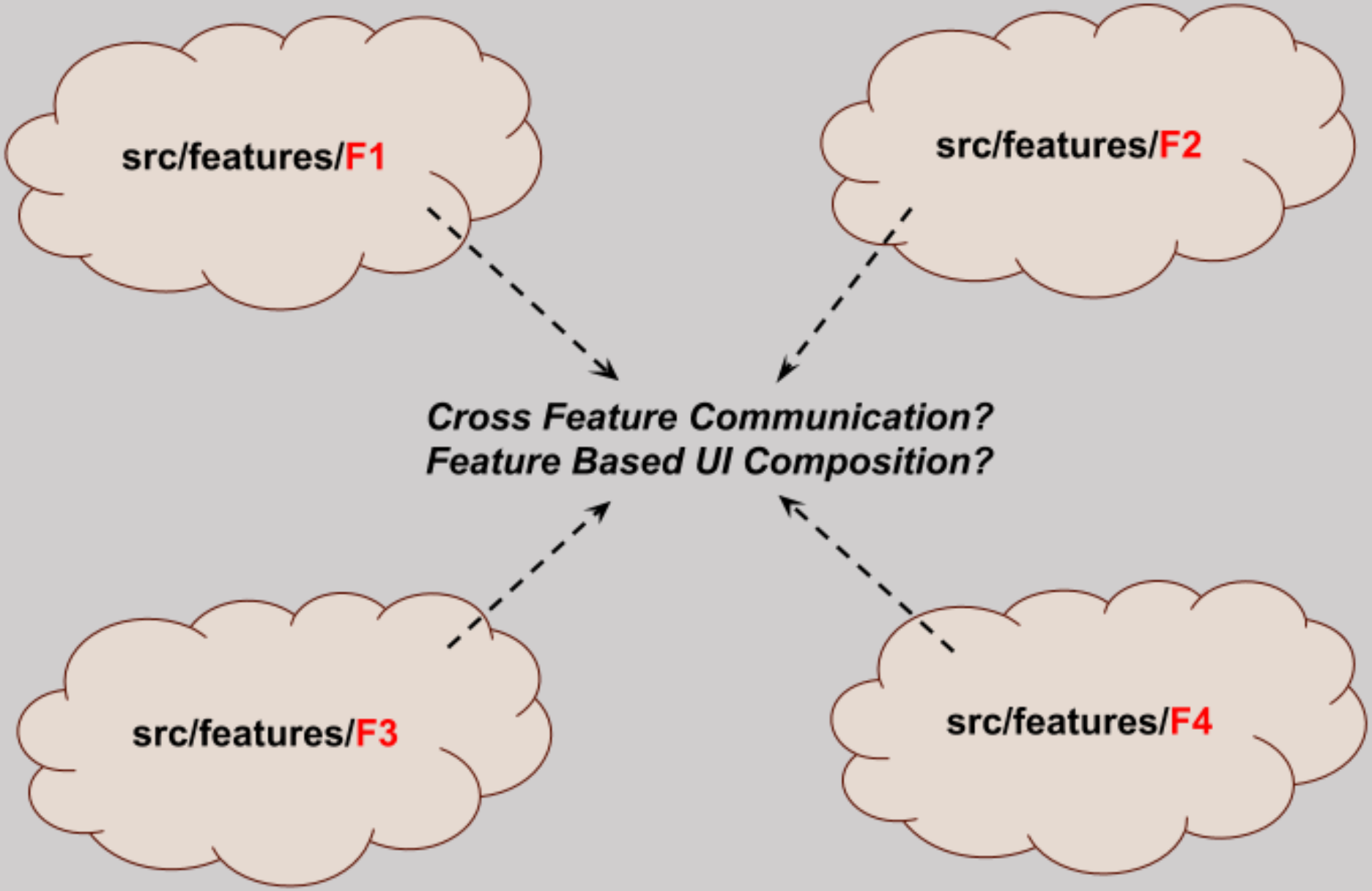
1. Feature Runtime Consolidation

2. Feature Collaboration

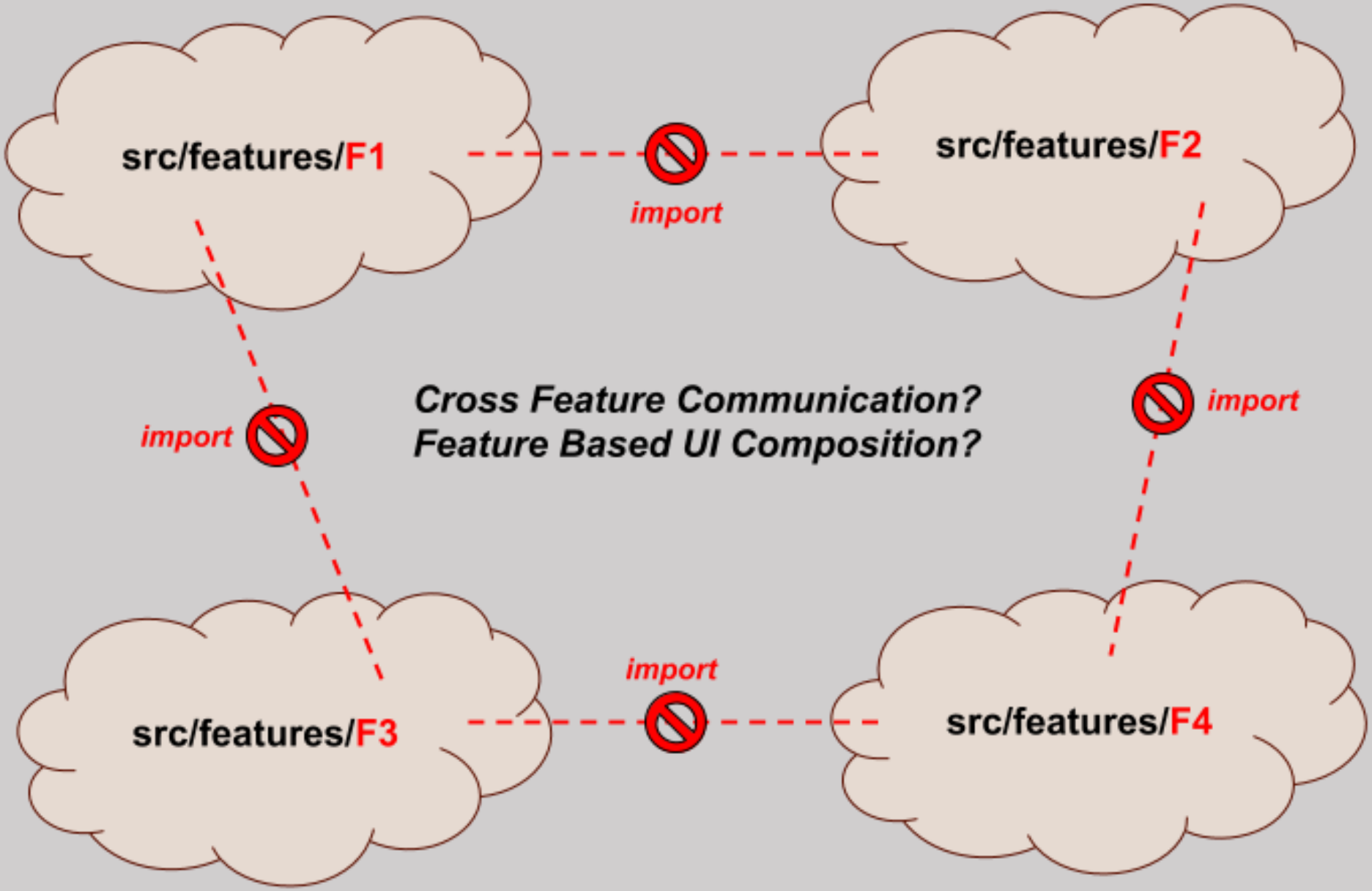
1. *Feature Runtime Consolidation*



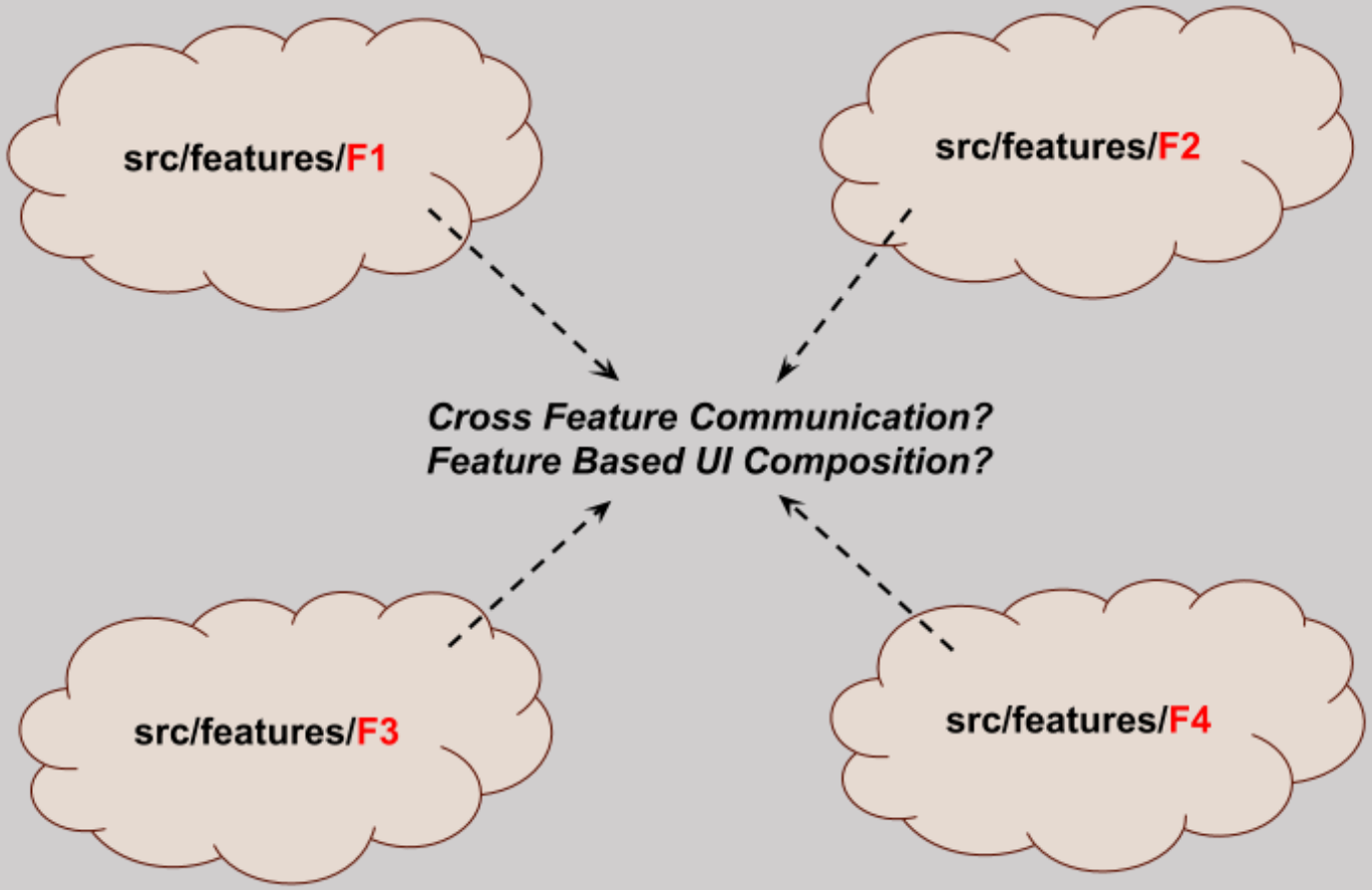
2. *Feature Collaboration*



~~Cross Feature Imports~~



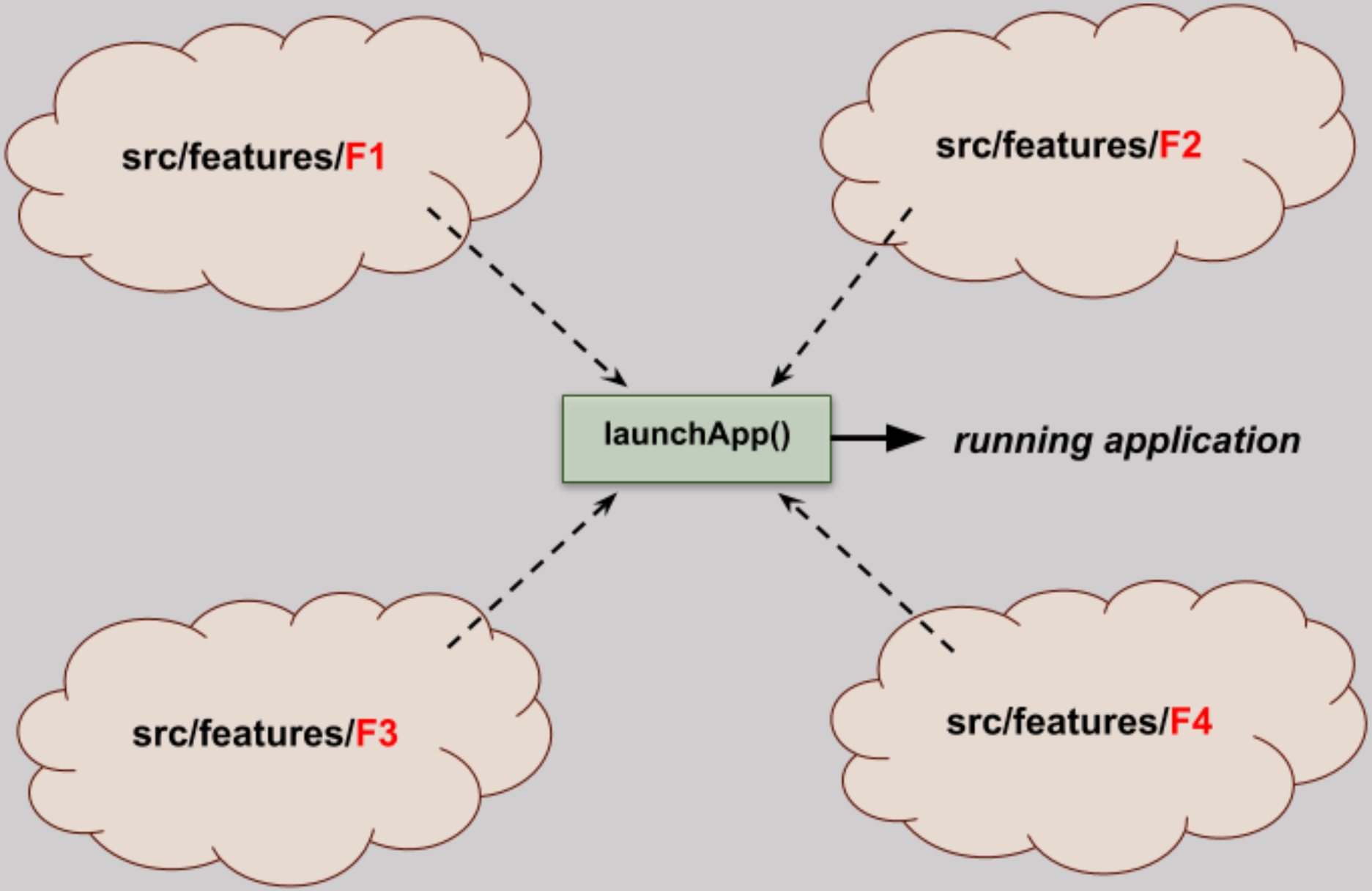
2. *Feature Collaboration*



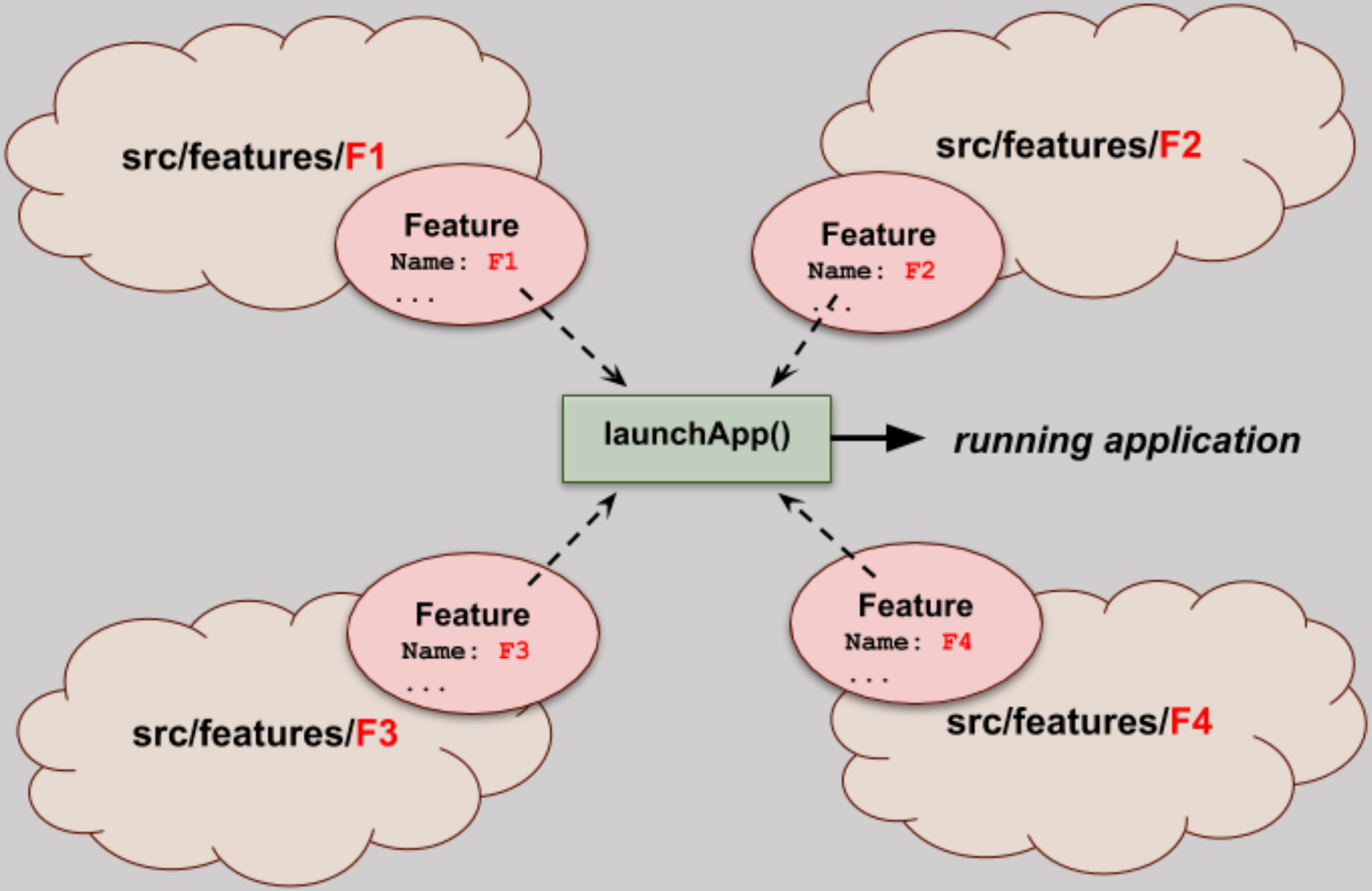
How is Cross Feature Communication achieved in a way that doesn't break encapsulation?

The feature-u Solution

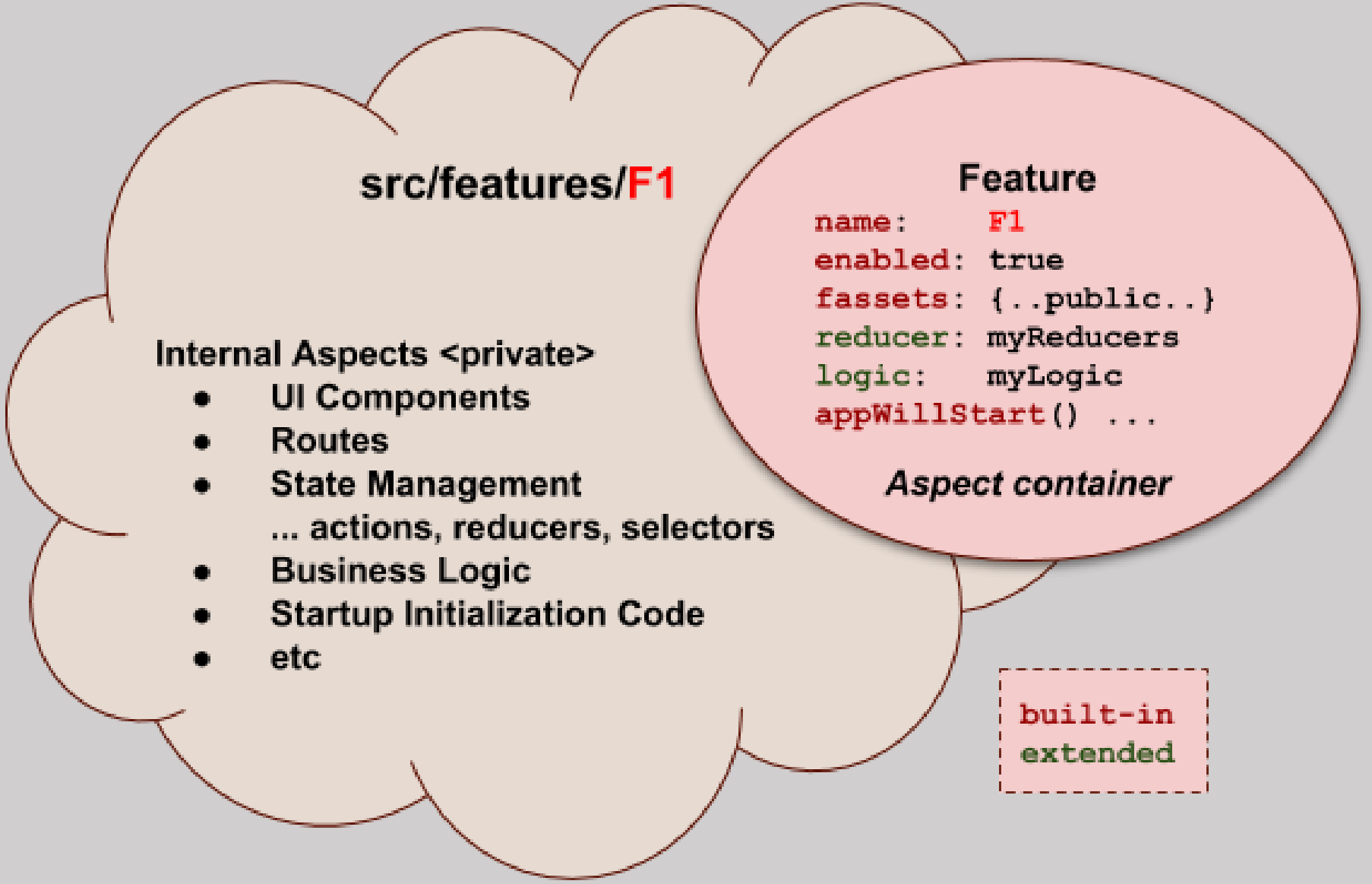
Introducing: launchApp()



Introducing: Feature Object



Introducing: aspects



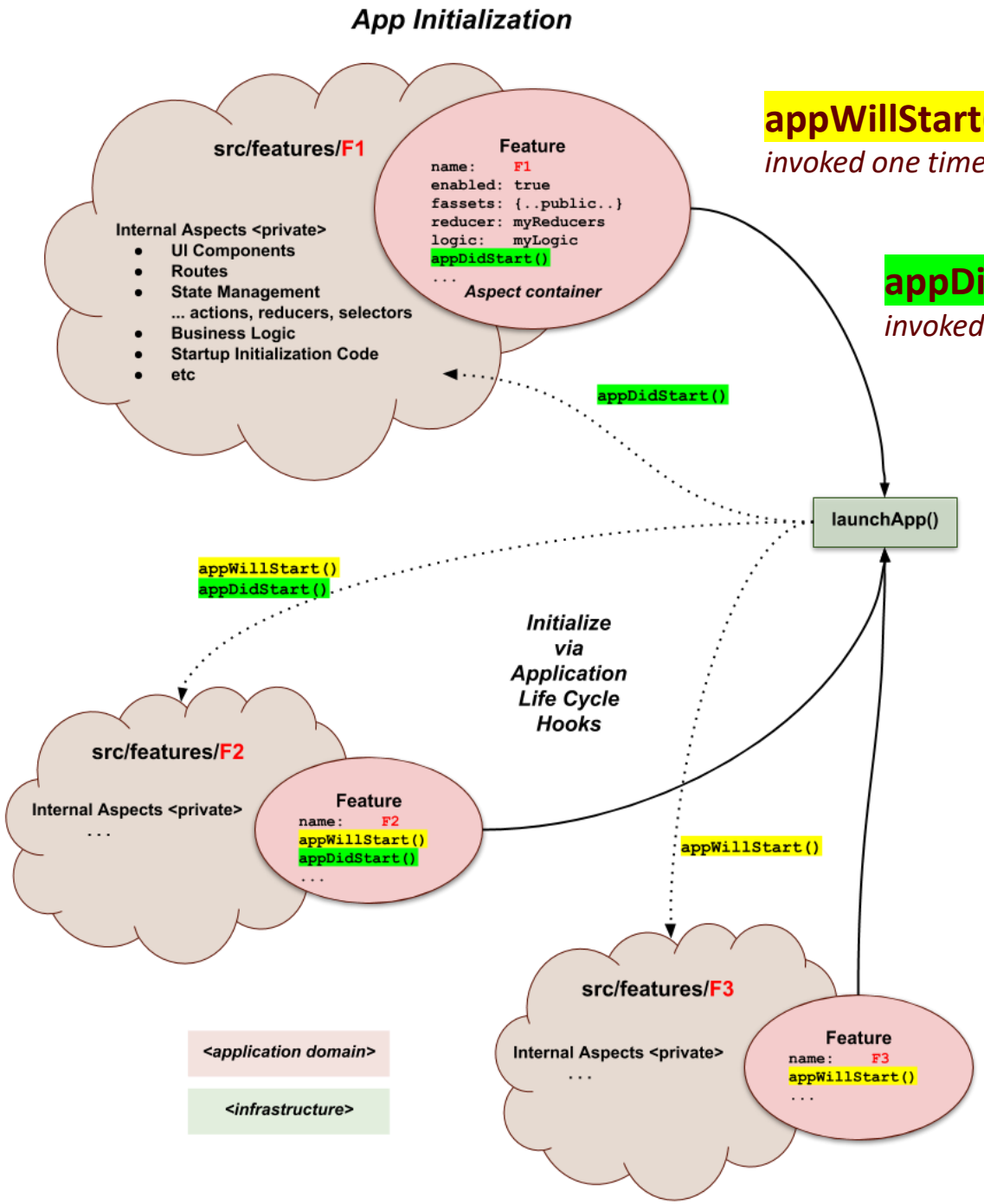
How does **feature-u** accommodate:

1. Feature Runtime Consolidation

APP INITIALIZATION

FRAMEWORK CONFIGURATION

Application Life Cycle Hooks



appWillStart({fassets, curRootAppElm}): rootAppElm | falsy

invoked one time at app startup time

appDidStart ({fassets, [appState, dispatch]}): void

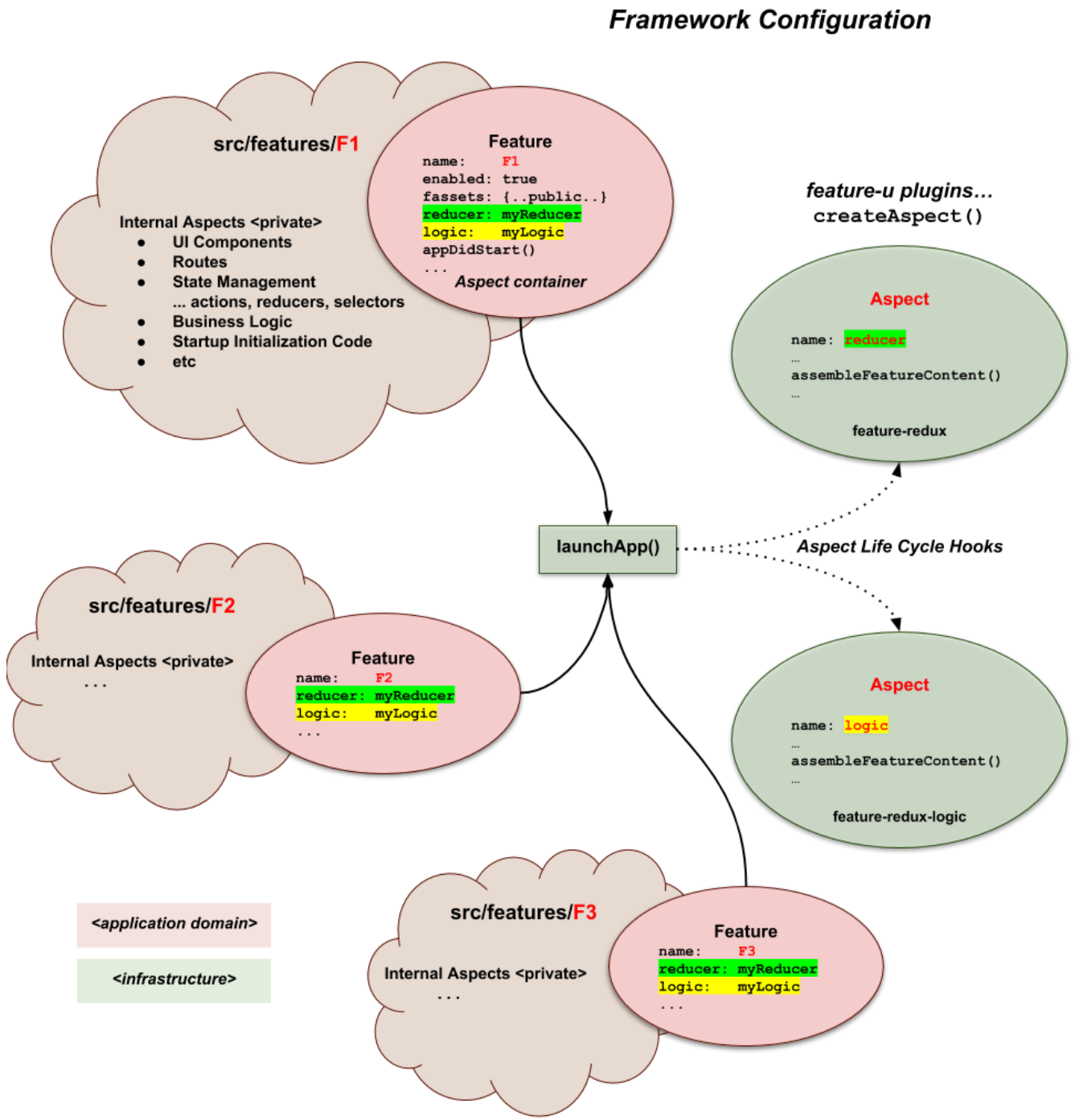
invoked one time immediately after app starts

Extendable Aspect Plugins

Framework Configuration Goal

feature-u is extendable!

Extendable Aspect Plugins

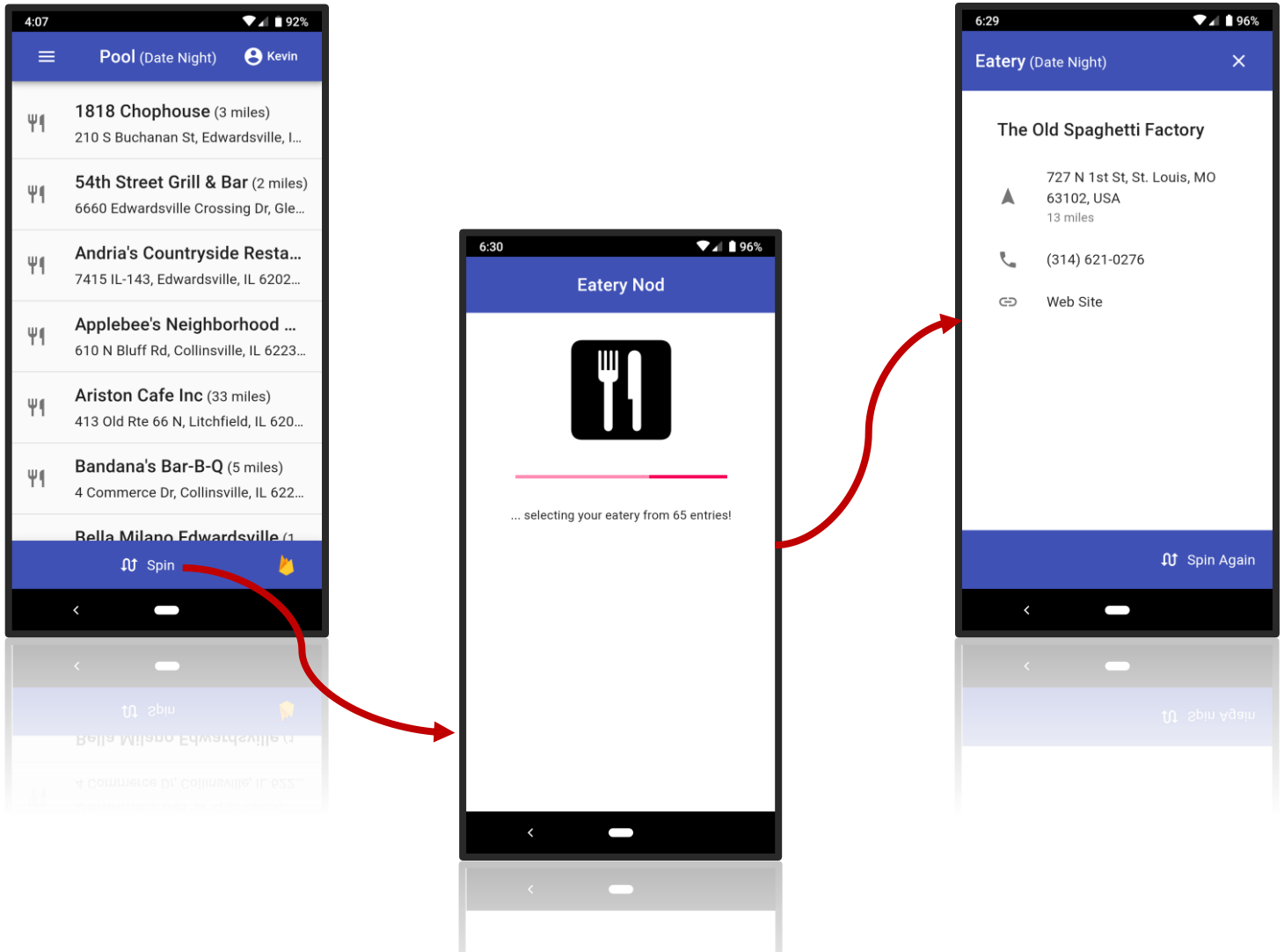


1. Feature Runtime Consolidation

eatery-nod-w

https://github.com/KevinAst/eatery-nod-w

1. Feature Runtime Consolidation



1. Feature Runtime Consolidation

directory structure

src/app.js

src/app.js

Our App is running from

isolated/independent

features …

- App Initialization
- Framework Configuration

src/feature/device/appWillStart.js

src/feature/firebase/feature.js

redux auto configured by feature-redux Aspect Plugin

src/feature/leftNav/appWillStart.js

src/feature/device/appDidStart.js

Feature Enablement

Feature Enablement

Feature Enablement

- **by default all Features are active**
- **can be disable 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:

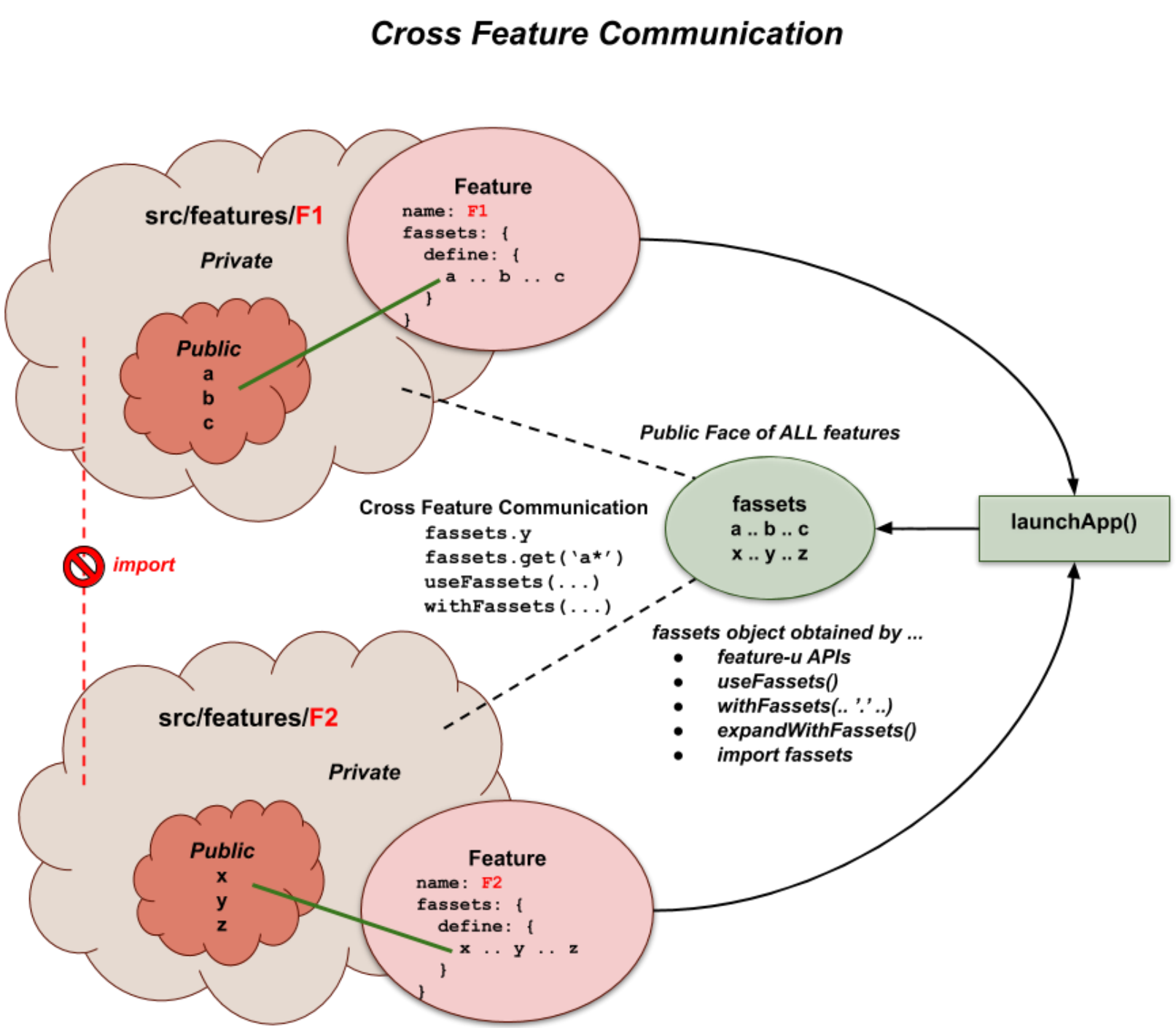
2. Feature Collaboration

Cross Feature Communication

Feature Based UI Composition

Cross Feature Communication

fassets



Cross Feature Communication

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': () => <img src="logo.png"/>, // a react component
    },
  },
  ...
});
```

HoC
Higher-order Components

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

useFassets()

code snippet ...

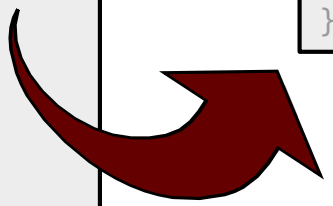
defining logo

```
export default createFeature({

  name: 'common',

  fassets: {
    define: {
     'company.logo': () => <img src="logo.png"/>, // a react component
    },
  },
  ...
});
```

feature-u V2 supports React Hooks

injecting fasset component properties

```
function MyComponent({Logo}) {
  return (
    <div>
      <Logo/>
    </div>
    ... snip snip
  );
}

export default withFassets({
  component: MyComponent,
  mapFassetsToProps: {
    Logo: 'company.logo',
  }
});
```



useFassets()

```
export default function MyComponent() {
 const Logo = useFassets('company.logo');
 return (
    <div>
      <Logo/>
    </div>
    ... snip snip
  );
}
```

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

code snippet ...

src/features/main/feature.js

```
createFeature({
  name: 'main',

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

src/features/main/comp/MainPage.js

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

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

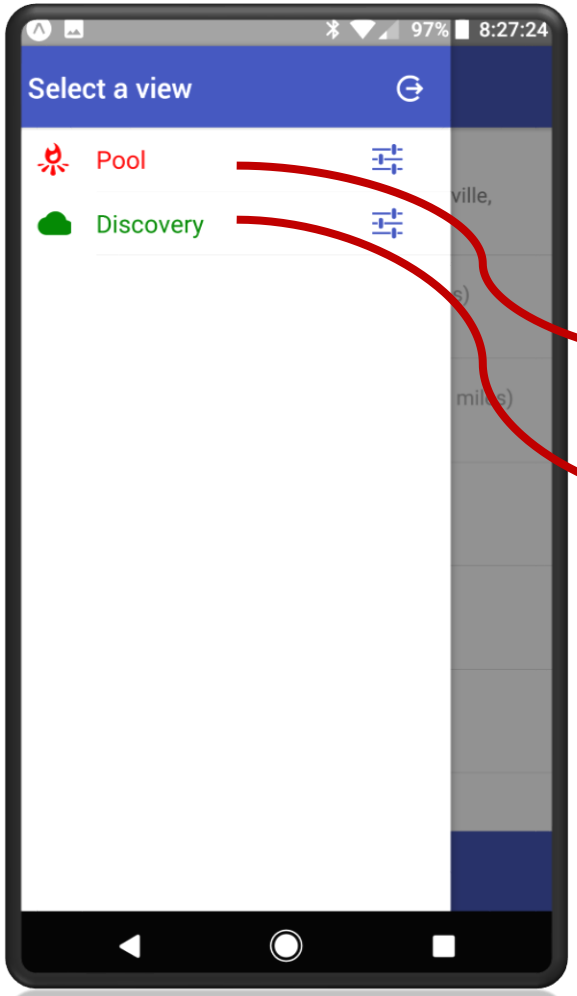
```
createFeature({
  name: 'search',

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

eatery-nod-w

https://github.com/KevinAst/eatery-nod-w

UI Composition

Resource Contracts



LeftNav **menu**
in *baseUI* feature

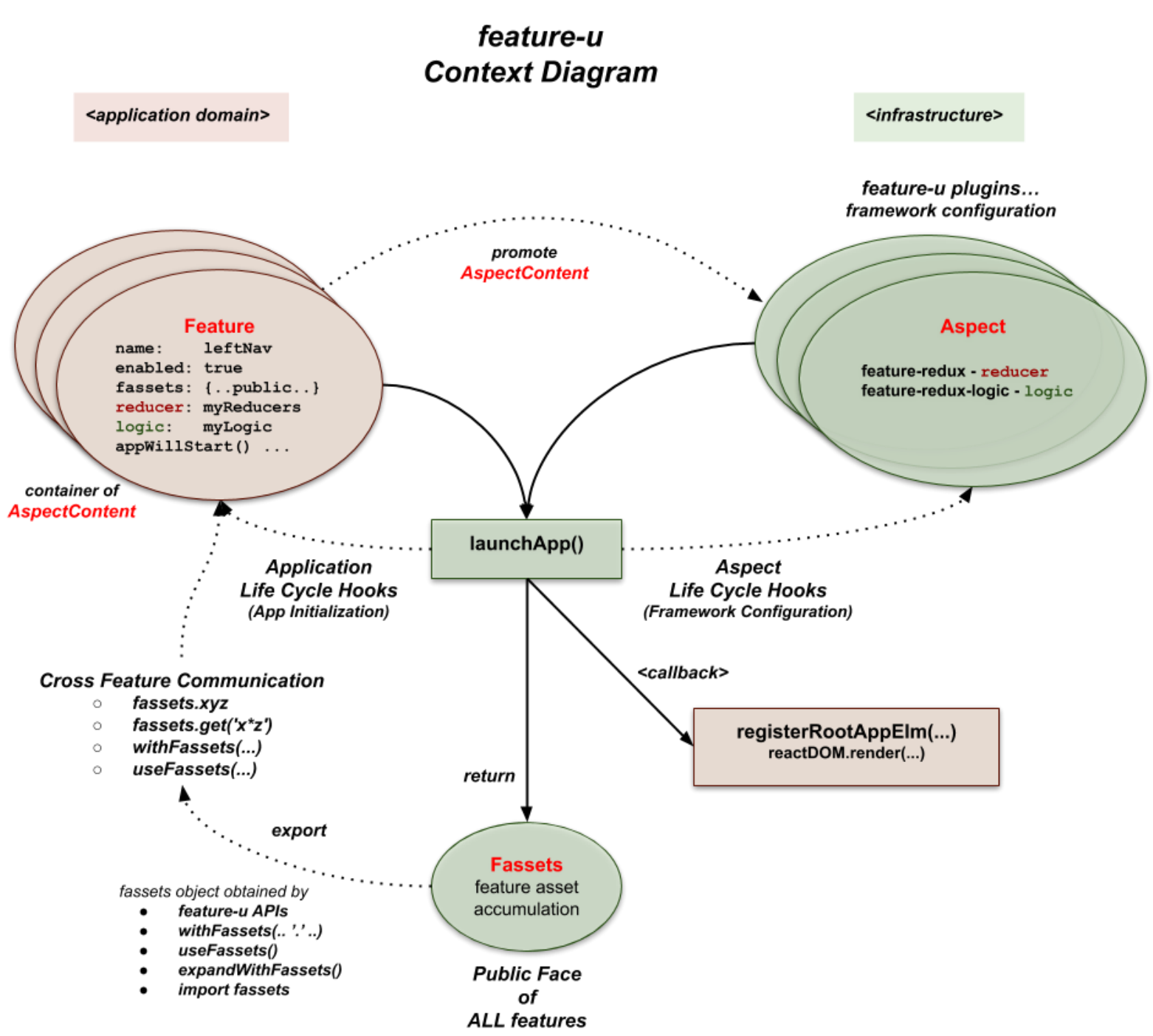
eatery **screens**

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
<application domain>
<infrastructure>
feature-u plugins...
framework configuration
promote
AspectContent
Feature
name: leftNav
enabled: true
fassets: {..public..}
reducer: myReducers
logic: myLogic
appWillStart() ...
Aspect
feature-redux - reducer
feature-redux-logic - logic
container of
AspectContent
launchApp()
Application
Life Cycle Hooks
(App Initialization)
Aspect
Life Cycle Hooks
(Framework Configuration)
Cross Feature Communication
fassets.xyz
fassets.get('x*z')
withFassets(...)
useFassets(...)
<callback>
registerRootAppElm(...)
reactDOM.render(...)
return
export
Fassets
feature asset
accumulation
fassets object obtained by
feature-u APIs
withFassets(.. '.' ..)
useFassets()
expandWithFassets()
import fassets
Public Face
of
ALL features

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***

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***

A final word about Feature Based Development

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*)

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***

A final word about Feature Based Development

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

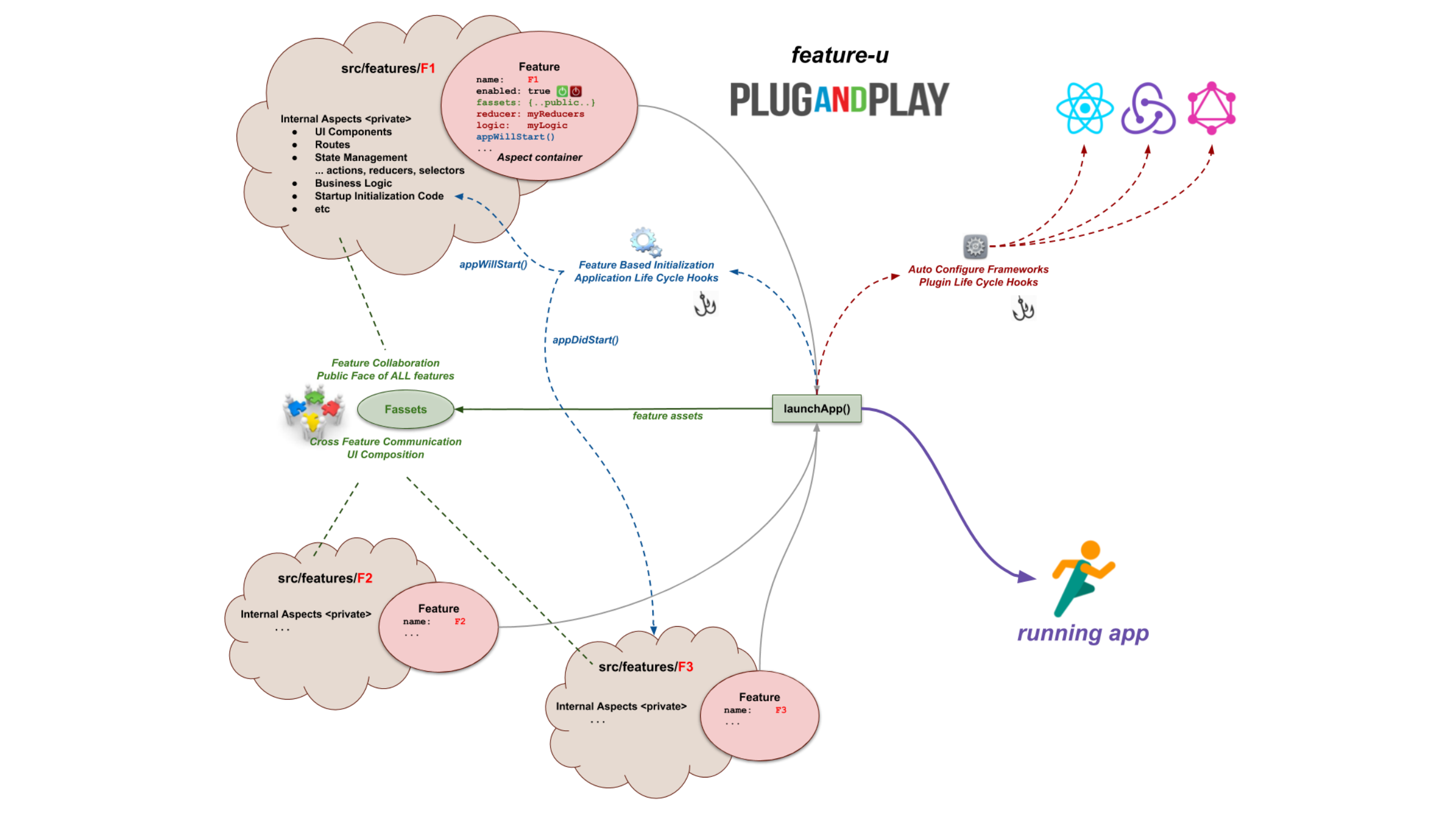
Question: "*How does* ***feature-u*** *impact my design constructs?*"

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*!


feature-u
PLUGANDPLAY
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
appWillStart()
Feature Based Initialization
Application Life Cycle Hooks
appDidStart()
Auto Configure Frameworks
Plugin Life Cycle Hooks
Feature Collaboration
Public Face of ALL features
Fassets
feature assets
launchApp()
Cross Feature Communication
UI Composition
running app
src/features/F2
Internal Aspects <private>
...
Feature
name: F2
...
src/features/F3
Internal Aspects <private>
...
Feature
name: F3
...