Unleash the power of feature-based JS development

Kevin Bridges

Applied Software Technologies

kevin@wiiBridges.com

@kevinast

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

slides, syllabus, articles, docs, and repo!

Kevin Bridges

Applied Software Technologies

kevin@wiiBridges.com

@kevinast

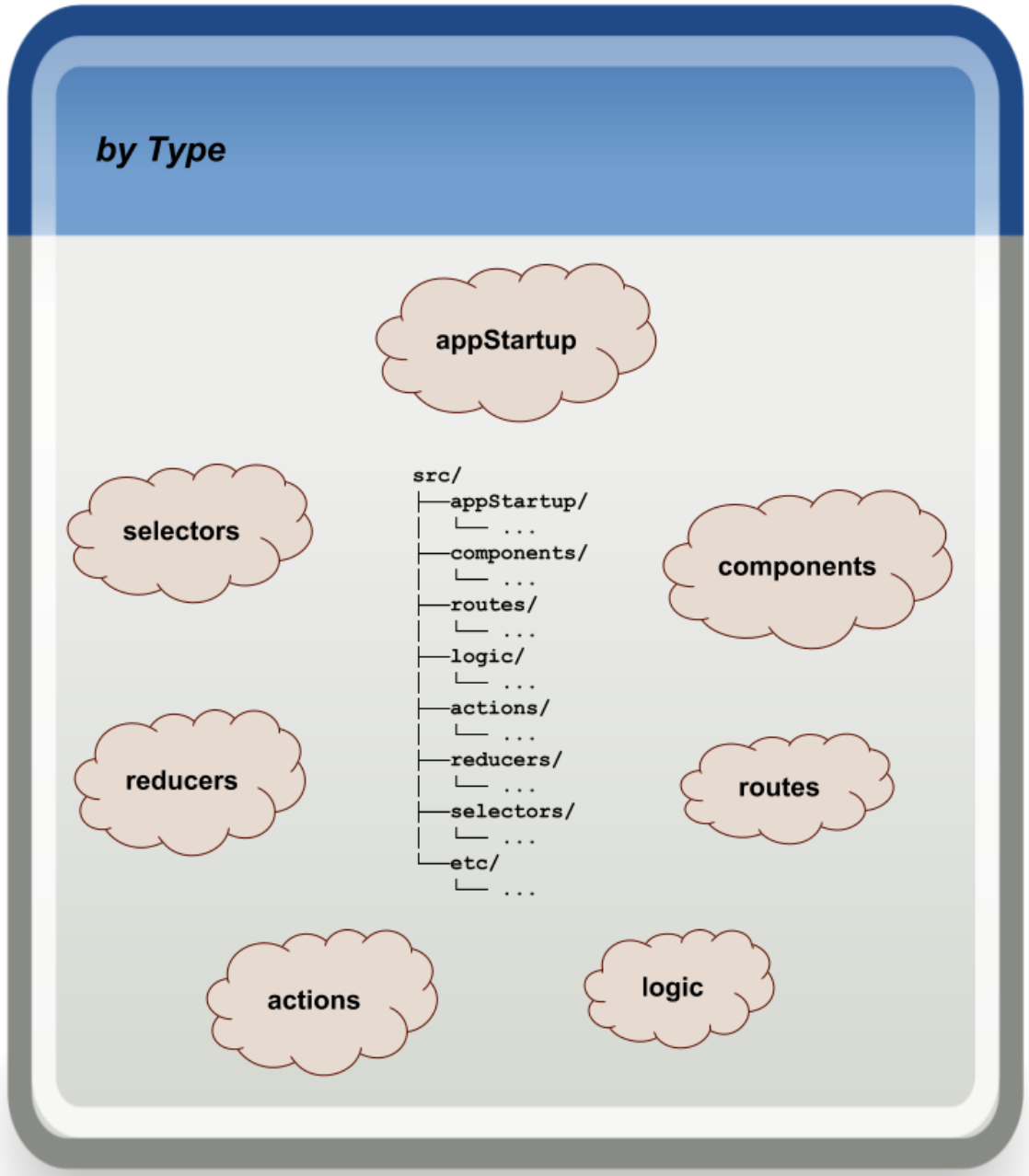
- Married, Father, Grandfather
- 40 yrs in software (20 yrs consulting)
- JS (since 96, es6 since 2015)
- Retired
- Be nice to the old guy :-)

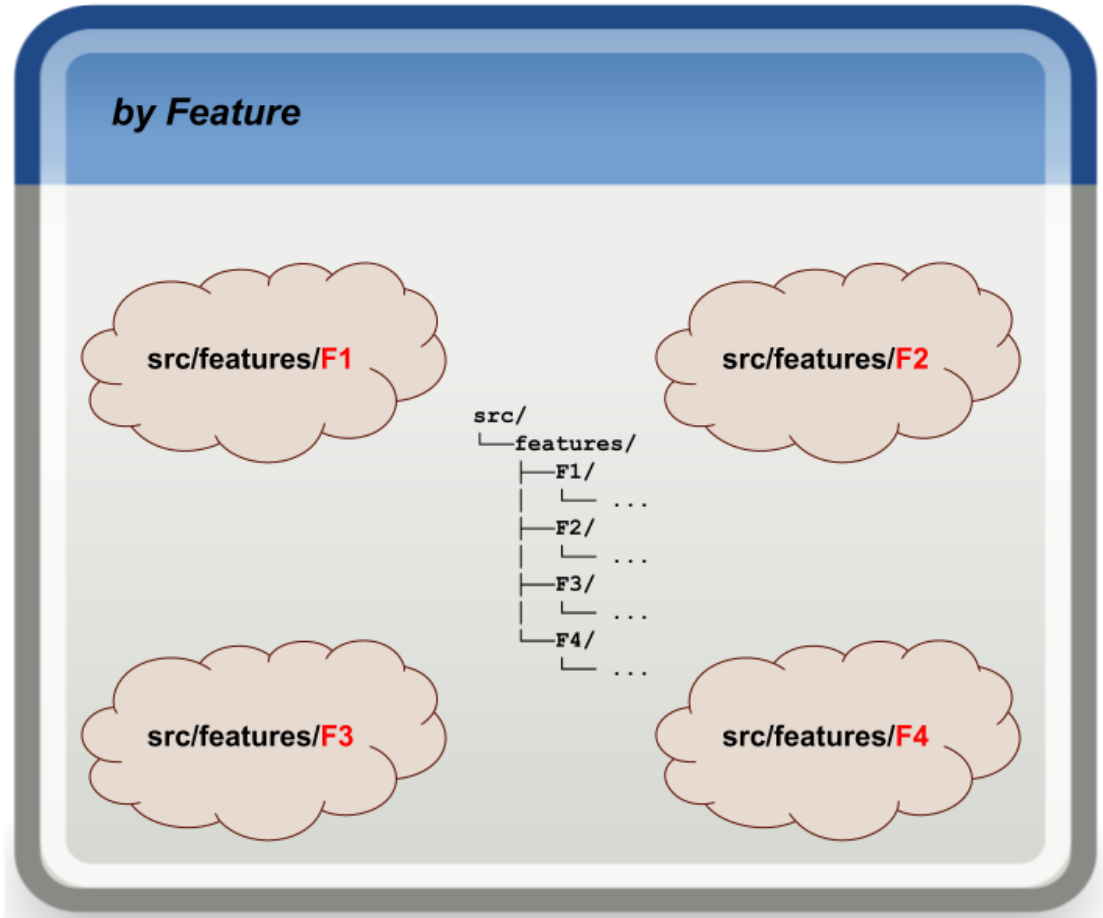
Unleash the power of feature-based JS development

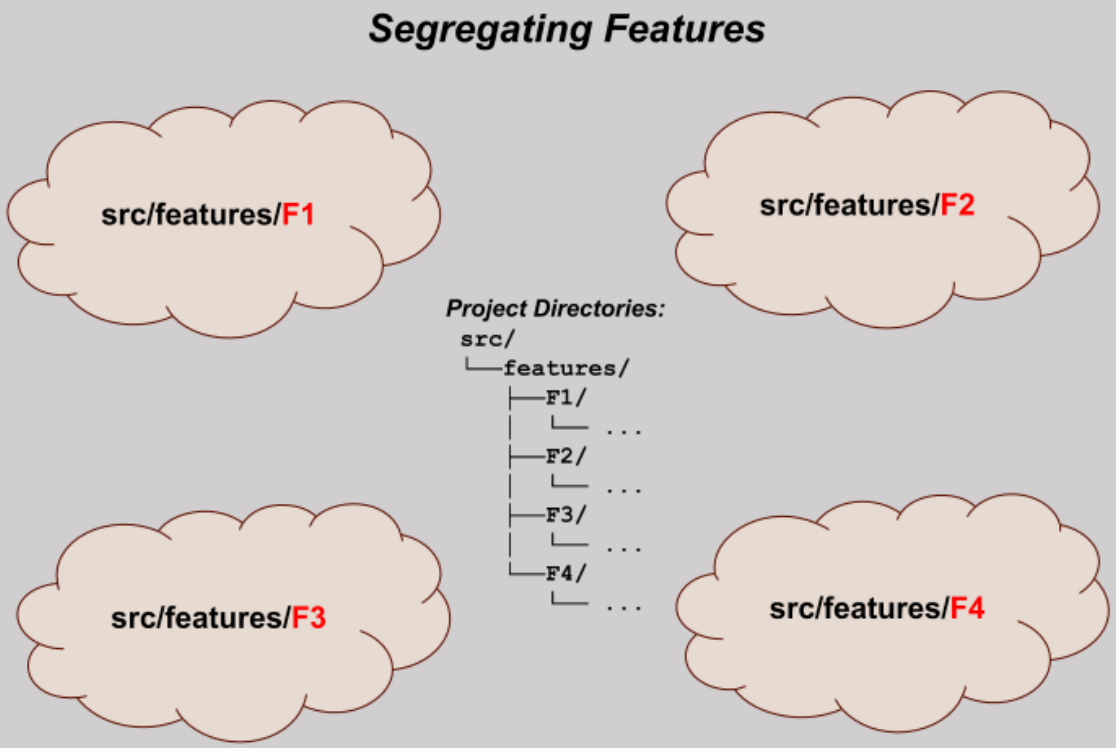
http://bit.ly/feature-u-slides

slides, syllabus, articles, docs, and repo!

Project Organization







Goals

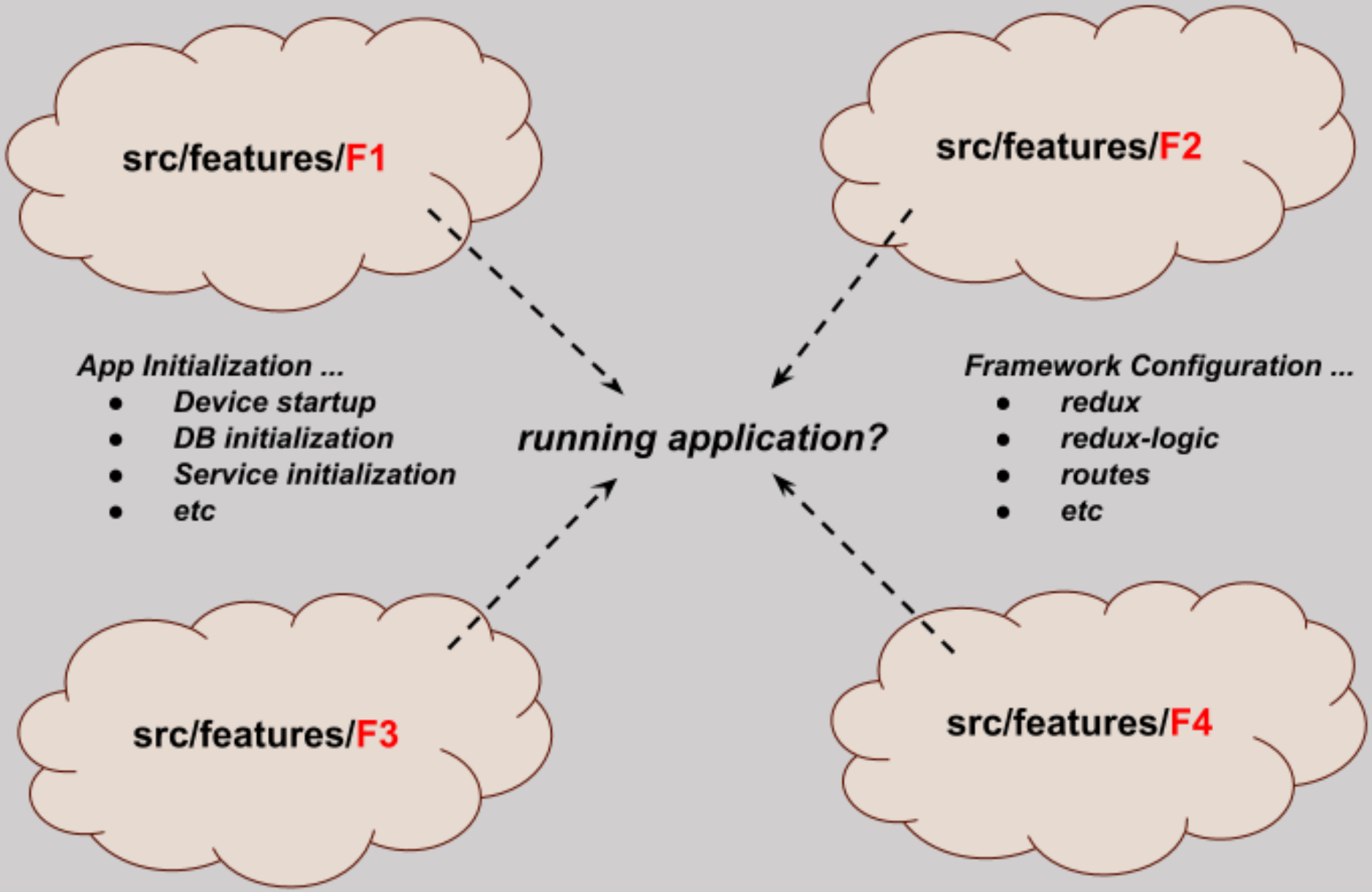
- Encapsulation
- Self Sufficient
- Plug-and-Play

Hurdles

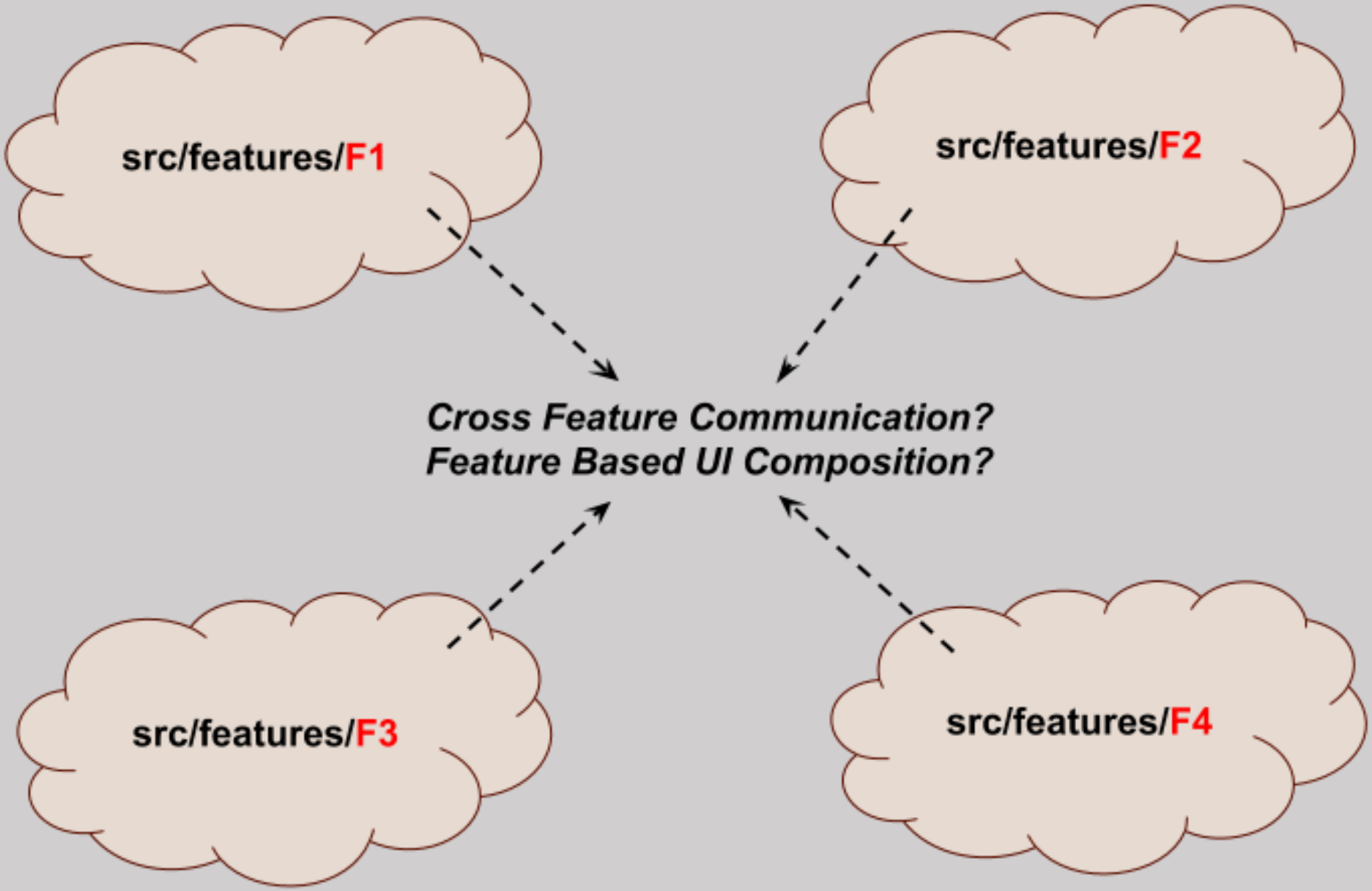
- Isolation vs. Collaboration
- Start-Up Initialization
- Framework Configuration
- UI Composition
- Feature Enablement

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

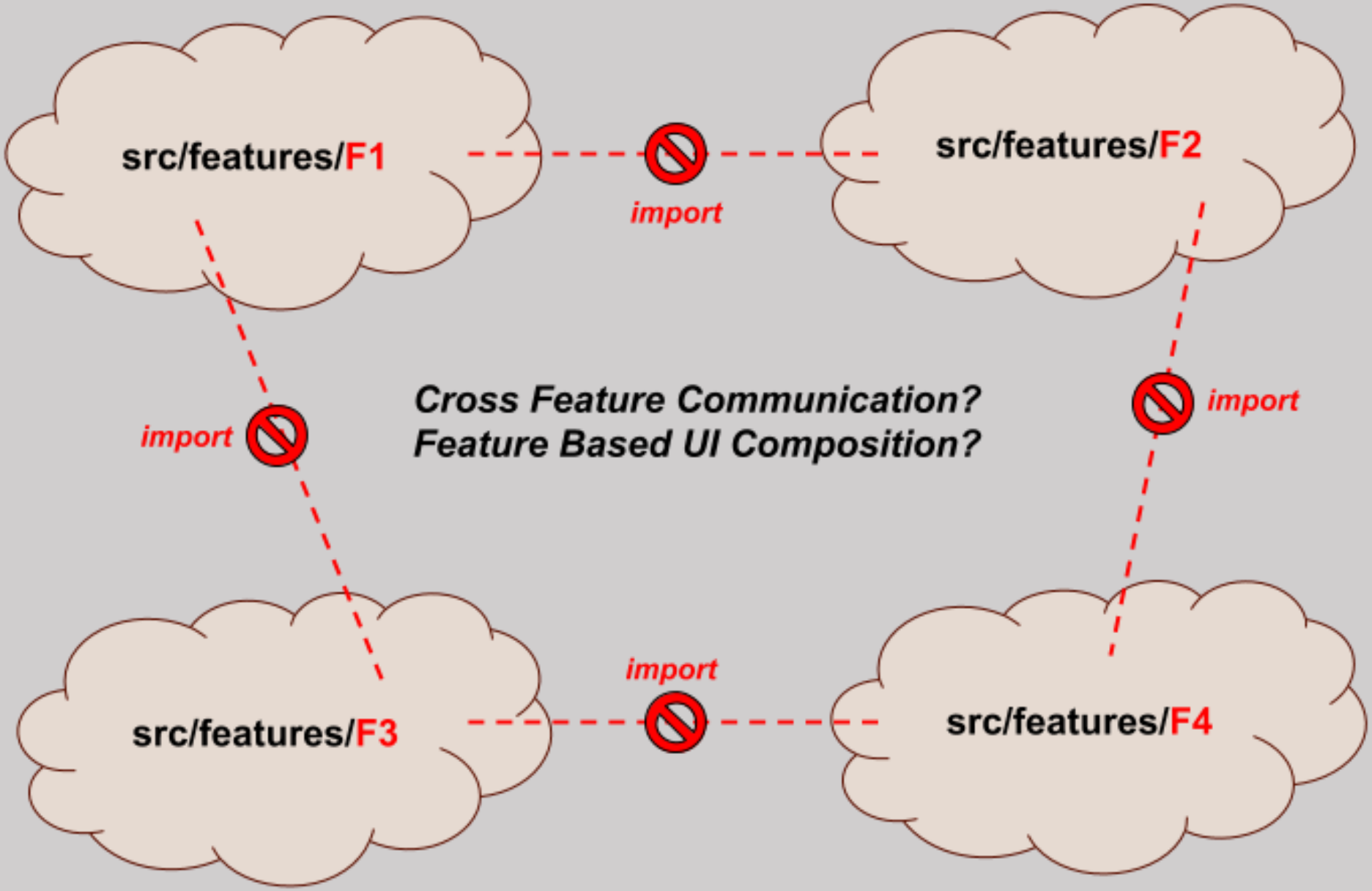
Feature Runtime Consolidation

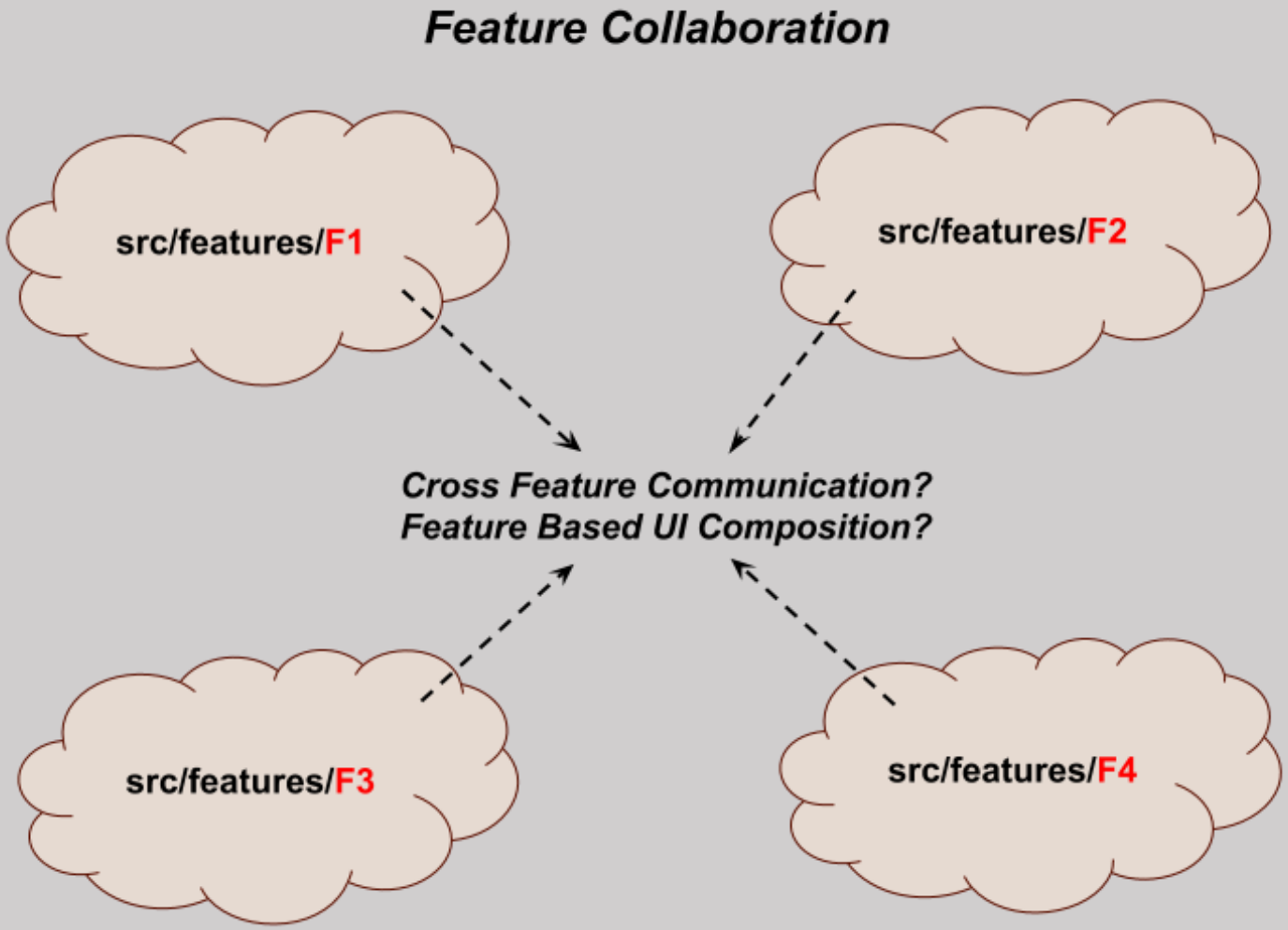


Feature Collaboration



~~Cross Feature Imports~~



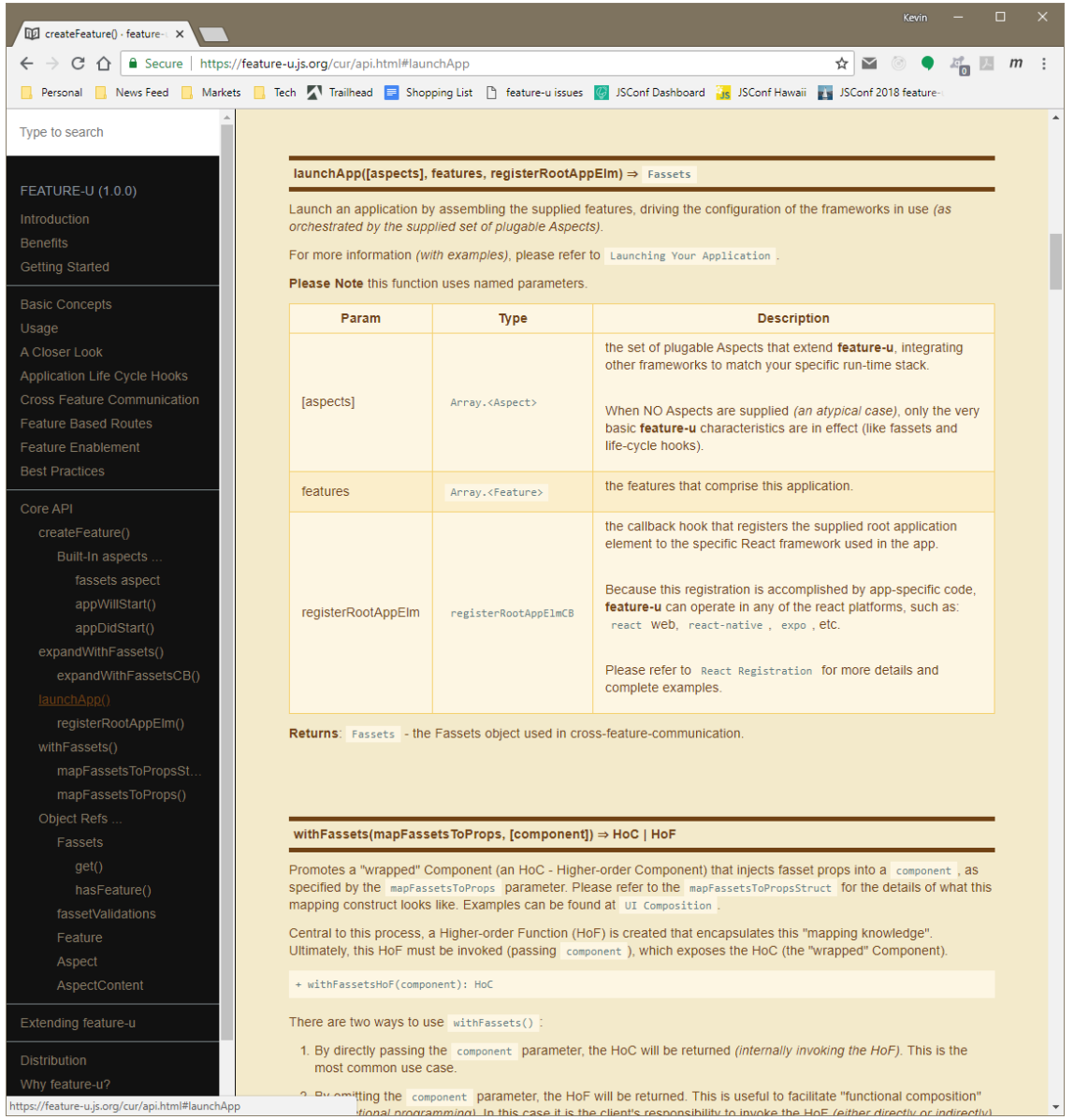


How is Cross Feature Communication achieved in a way that doesn't break encapsulation?

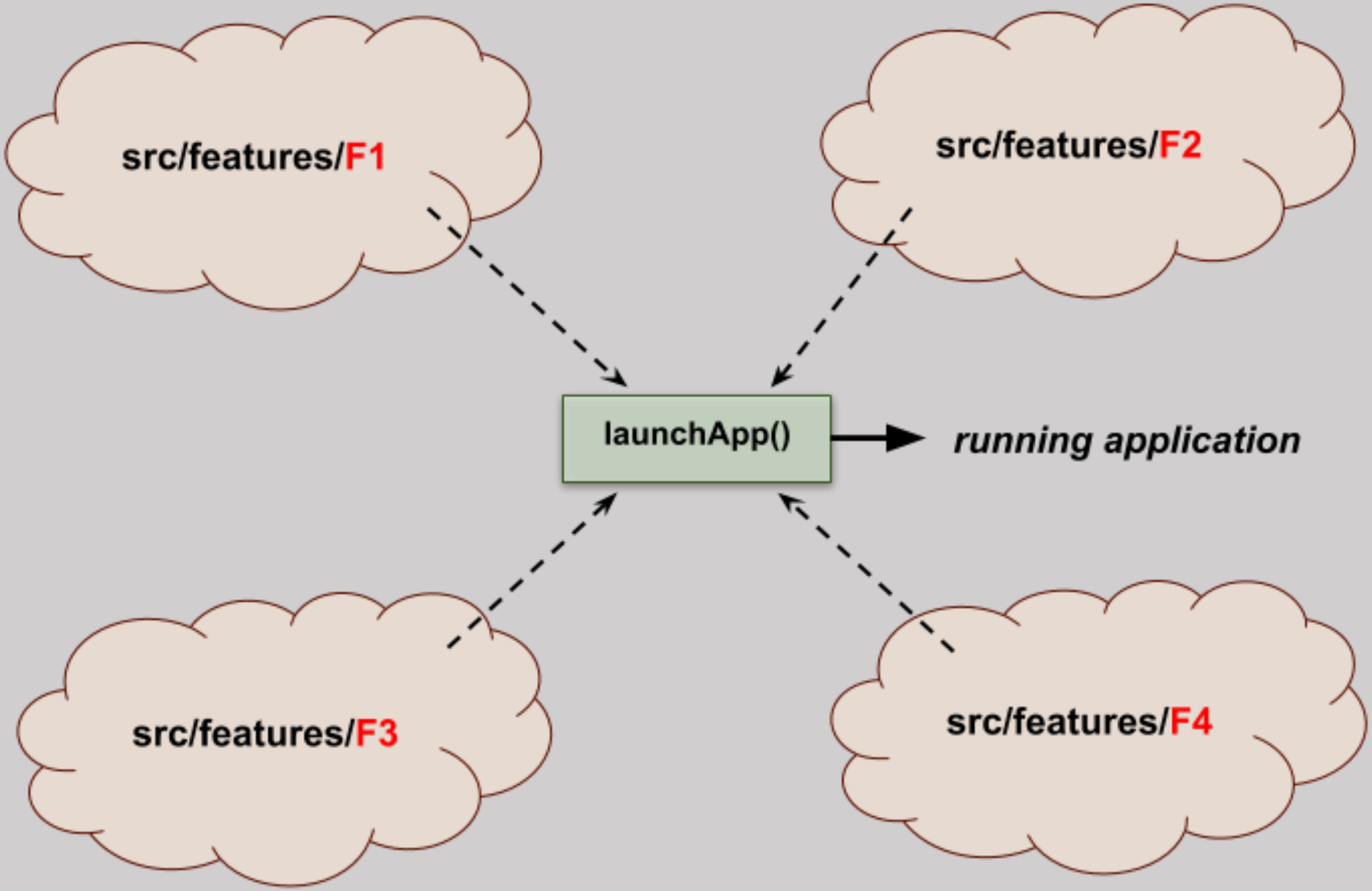
The feature-u Solution

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

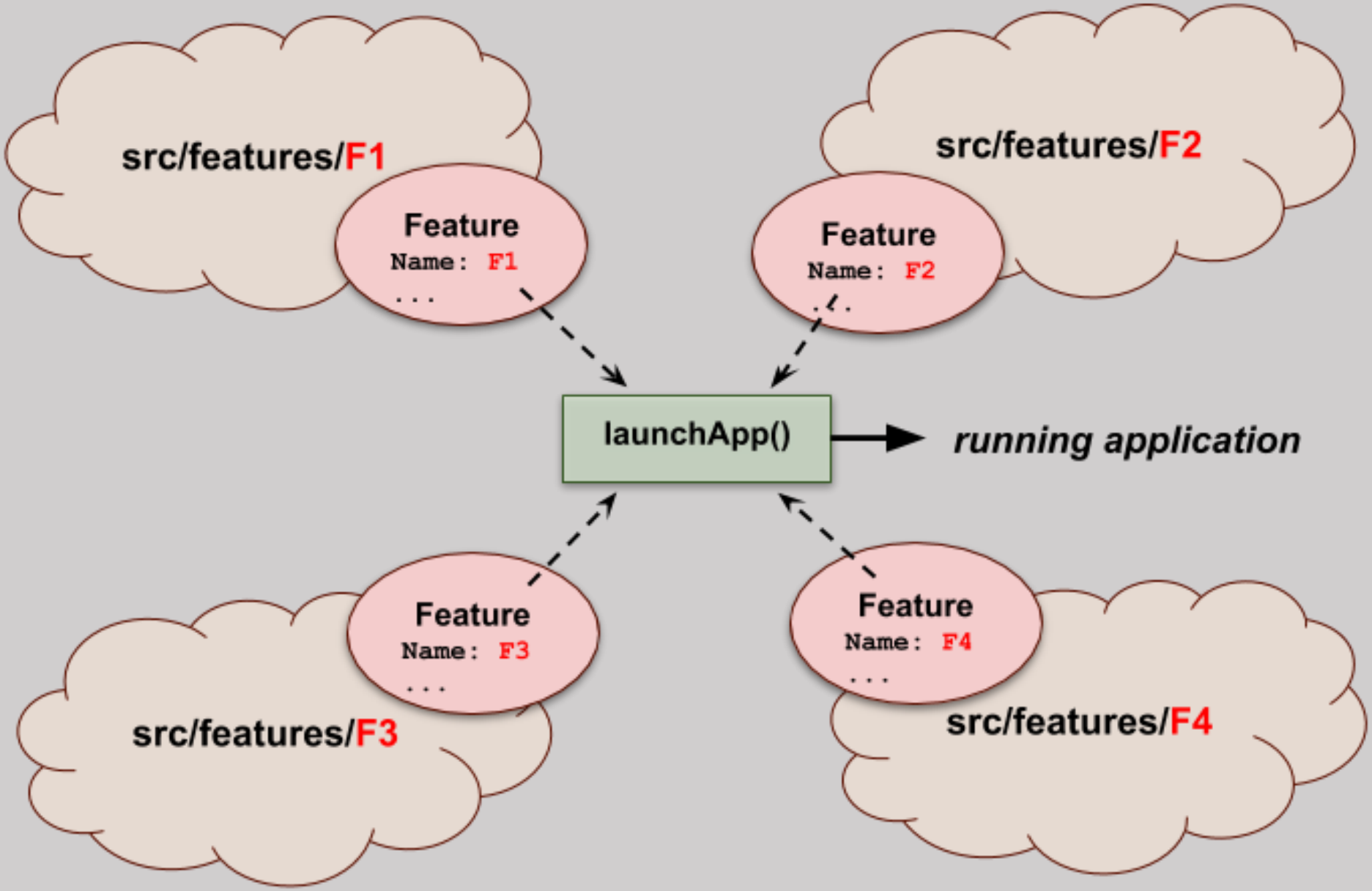
SideBar: feature-u docs



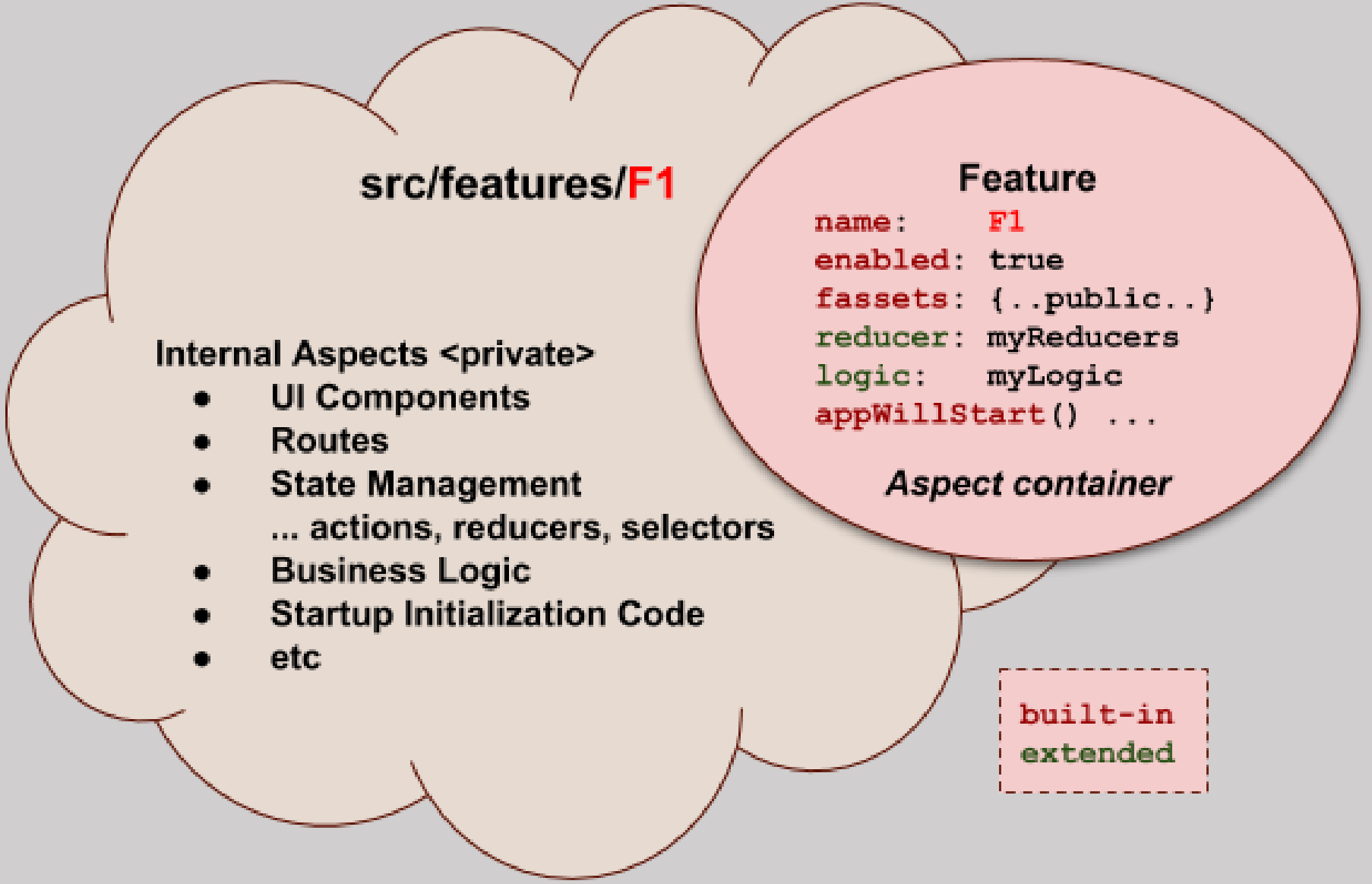
Introducing: launchApp()



Introducing: Feature Object



Introducing: aspects



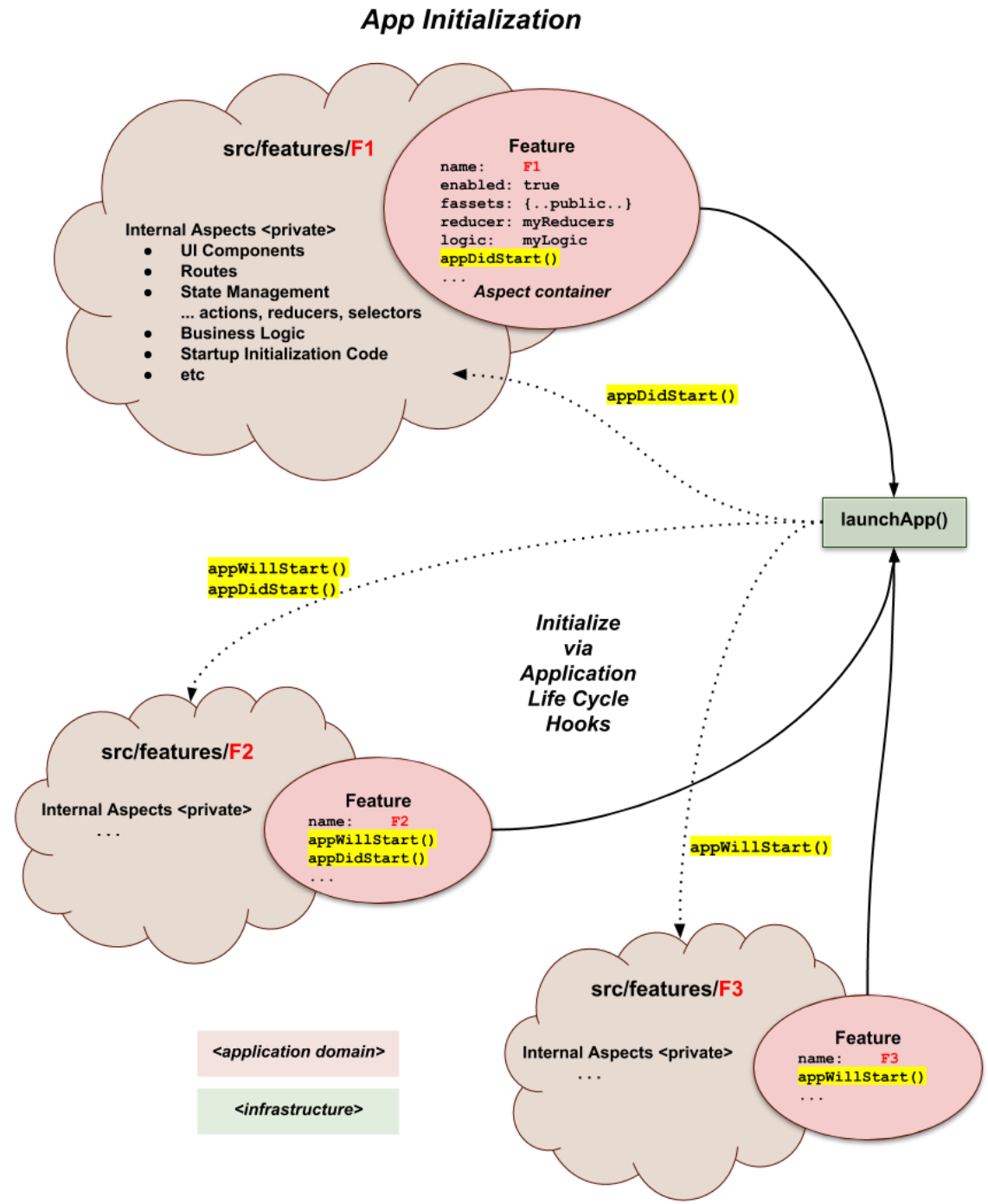
How does
feature-u
accommodate:

Feature Runtime Consolidation

APP INITIALIZATION

FRAMEWORK CONFIGURATION

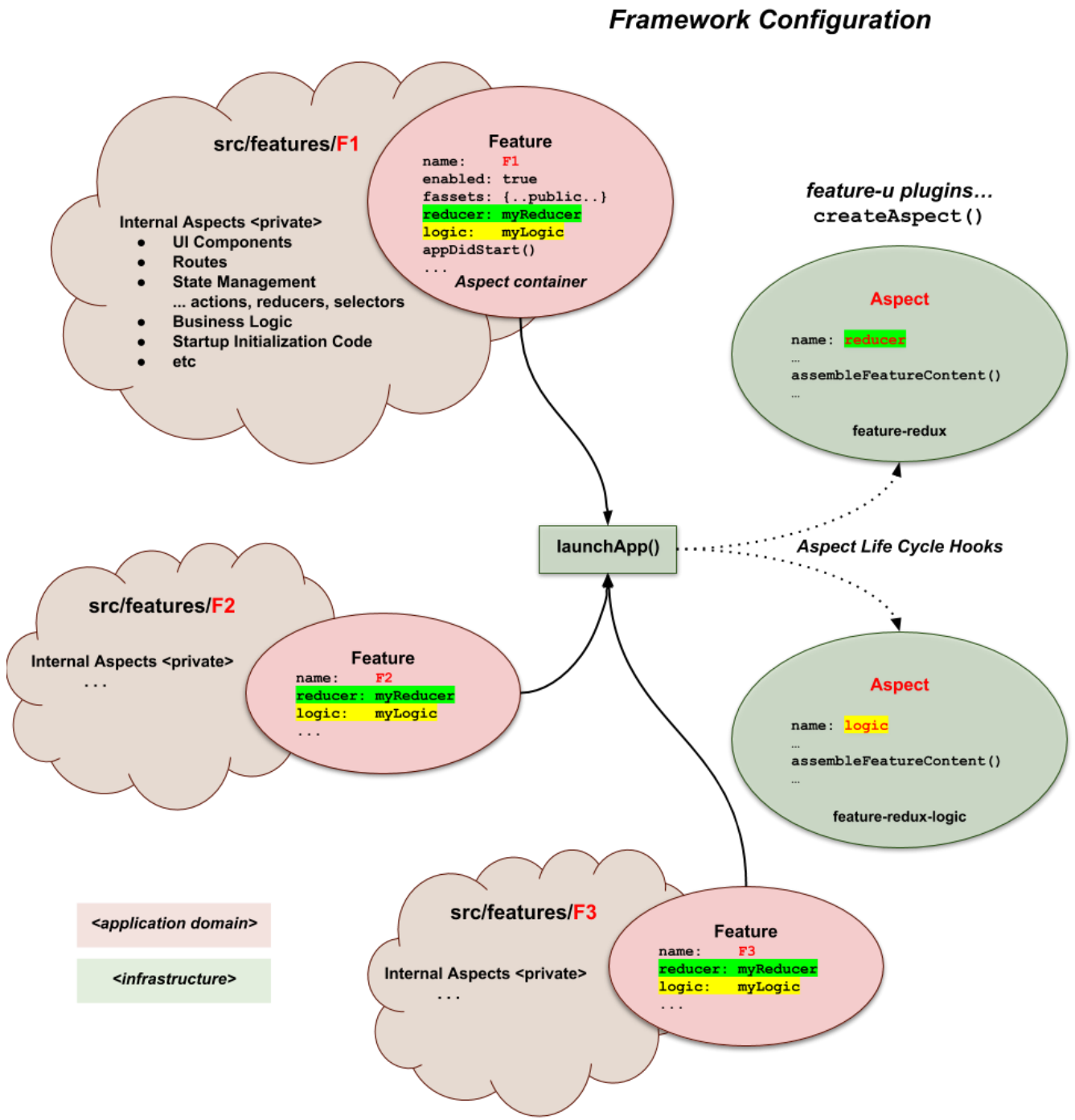
Application Life Cycle Hooks



appWillStart():
invoked one time at app startup time

appDidStart ():
invoked one time immediately after app has started

Extendable Aspect Plugins

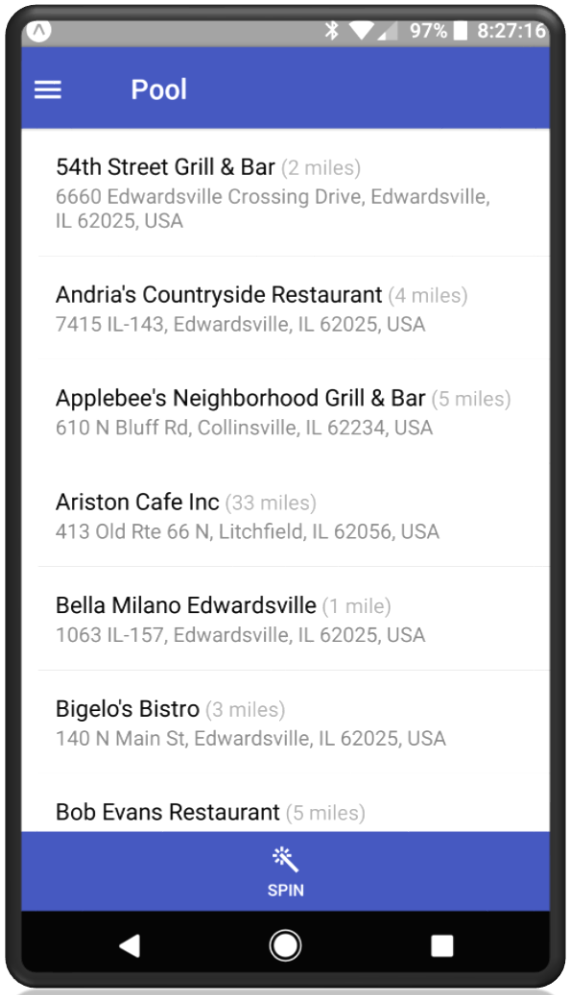


Feature Runtime Consolidation

eatery-nod

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

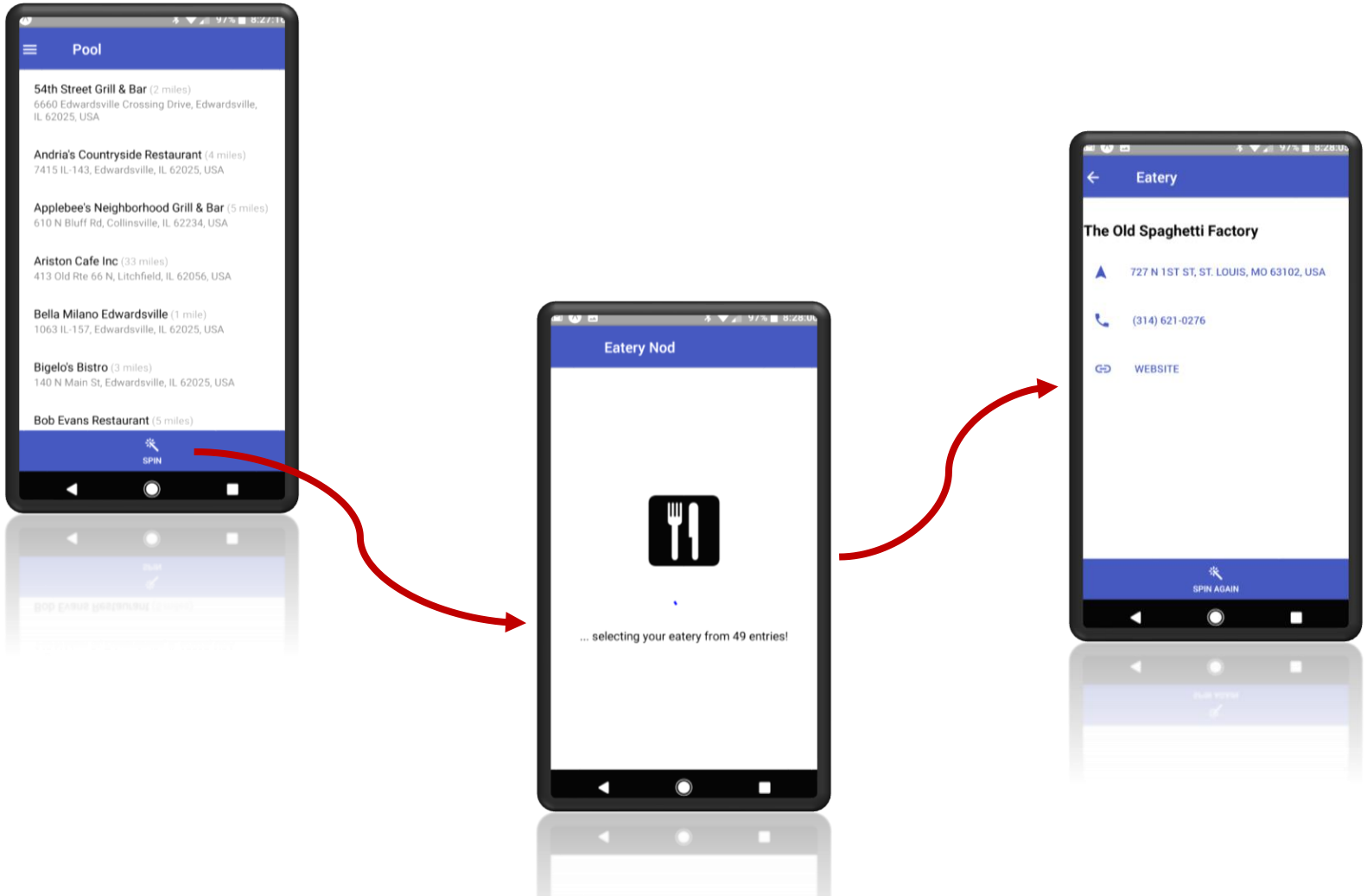
Feature
Runtime
Consolidation



eatery-nod

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

Feature
Runtime
Consolidation



Feature Runtime Consolidation

eatery-nod

directory structure

```
src/
|  app.js           ... our mainline - launches app via launchApp()
|
|--feature/
|   |  index.js      ... accumulate/promote all app Feature objects
|   |
|   |--auth/         ... the app's authorization feature
|   |   |  actions.js
|   |   |  fassets.js
|   |   |  feature.js ... expose aspects of interest to feature-u
|   |   |  featureName.js
|   |   |  index.js
|   |   |  logic.js
|   |   |  route.js
|   |   |  signInFormMeta.js
|   |   |  state.js
|   |   `--comp/
|   |          SignInScreen.js
|   |          SignInVerifyScreen.js
|   |
|   |--currentView/  ... more features
|   |
|   |--device/       ... feature to initialize the device
|   |   |  actions.js
|   |   |  api.js
|   |   |  appDidStart.js
|   |   |  appWillStart.js
|   |   |  fassets.js
|   |   |  feature.js ... expose aspects of interest to feature-u
|   |   |  featureName.js
|   |   |  index.js
|   |   |  logic.js
|   |   |  route.js
|   |   |  state.js
|   |   `--init/
|   |          platformSetup.android.js
|   |          platformSetup.ios.js
|   |
|   |--discovery/   ... more features
|   |--eateries/
|   |--firebase/
|   |--leftNav/
|   |--logActions/
|   `--sandbox/
|
`--util/           ... common utilities used across all features
```

directory structure

Feature
Runtime
Consolidation

launchApp()

src/app.js

```
import React                  from 'react';
import Expo                   from 'expo';
import {LayoutAnimation}      from 'react-native';
import {launchApp}            from 'feature-u';
import {createReducerAspect} from 'feature-redux';
import {createLogicAspect}    from 'feature-redux-logic';
import {createRouteAspect}    from 'feature-router';
import features               from './feature';
import SplashScreen           from './util/comp/SplashScreen';

// launch our application, exposing the feature-u Fassets object
// ... facilitating cross-feature-communication!
export default launchApp({
  aspects: appAspects(),
  features,
  registerRootAppElm(rootAppElm) {
    Expo.registerRootComponent(()=>rootAppElm); // convert rootAppElm to a React Component
  }
});


// accumulate/configure the Aspect plugins matching our app's run-time stack
function appAspects() {

  // define our framework run-time stack
  const reducerAspect = createReducerAspect();
  const logicAspect   = createLogicAspect();
  const routeAspect   = createRouteAspect();
  const aspects = [
    reducerAspect, // redux          ... extending: Feature.reducer
    logicAspect,   // redux-logic    ... extending: Feature.logic
    routeAspect,   // Feature Routes ... extending: Feature.route
  ];

  // configure Aspects (as needed)
  // ... StateRouter fallback screen (when no routes are in effect)
  routeAspect.config.fallbackElm$ = <SplashScreen msg="I'm trying to think but it hurts!"/>;

  // beam me up Scotty :-)
  return aspects;
}
```

directory structure

platformSetup()
and
<Notify> Root Injection

src/app.js

Feature
Runtime
Consolidation

src/feature/device/appWillStart.js

```
import React           from 'react';
import platformSetup   from './init/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 in the root DOM
 */
export default function appWillStart({fassets, curRootAppElm}) {
  // platform-specific setup (iOS/Android)
  platformSetup();

  // initialize notify utility, by injecting it in our App root
  return [React.Children.toArray(curRootAppElm), <Notify key="Notify"/>];
}
```

directory structure

src/app.js

firebase initialization

Feature Runtime Consolidation

src/feature/firebase/feature.js

```
import {createFeature}  from 'feature-u';
import initFireBase     from './init/initFireBase';

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

  appWillStart({fassets, curRootAppElm}) {
    initFireBase(); // initialize FireBase
  },
});
```

src/feature/device/appWillStart.js

directory structure

src/app.js

Feature Runtime Consolidation

create redux appStore

redux auto configured
by
***feature-redux* Aspect Plugin**

src/feature/device/appWillStart.js

src/feature/firebase/feature.js

Feature Runtime Consolidation

directory structure

<LeftNav> root injection

src/feature/**leftNav**/appWillStart.js

```
import React             from 'react';
import {Drawer}          from 'native-base';
import SideBar,
      {registerDrawer,
       closeSideBar}     from './comp/SideBar';

/**
 * Inject our Drawer/SideBar component at the root of our app
 */
export default function appWillStart({fassets, curRootAppElm}) {
  return (
    <Drawer ref={ ref => registerDrawer(ref) }
            content={<SideBar/>}
            onClose={closeSideBar}>
      {curRootAppElm}
    </Drawer>
  );
}
```

src/app.js

src/feature/device/appWillStart.js

src/feature/firebase/feature.js

redux auto configured by feature-redux Aspect Plugin

directory structure

src/app.js

Registration of the React Component Root

Feature Runtime Consolidation

src/app.js

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

src/feature/device/appWillStart.js

src/feature/firebase/feature.js

redux auto configured by feature-redux Aspect Plugin

src/feature/leftNav/appWillStart.js

Bootstrap Process

Feature Runtime Consolidation

directory structure

src/app.js

src/app.js

src/feature/device/appDidStart.js

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

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

src/feature/device/appWillStart.js

src/feature/firebase/feature.js

redux auto configured by feature-redux Aspect Plugin

src/feature/leftNav/appWillStart.js

directory structure

src/app.js

Our App is running from

isolated/independent

features …

- App Initialization
- Framework Configuration

Feature Runtime Consolidation

src/app.js

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:

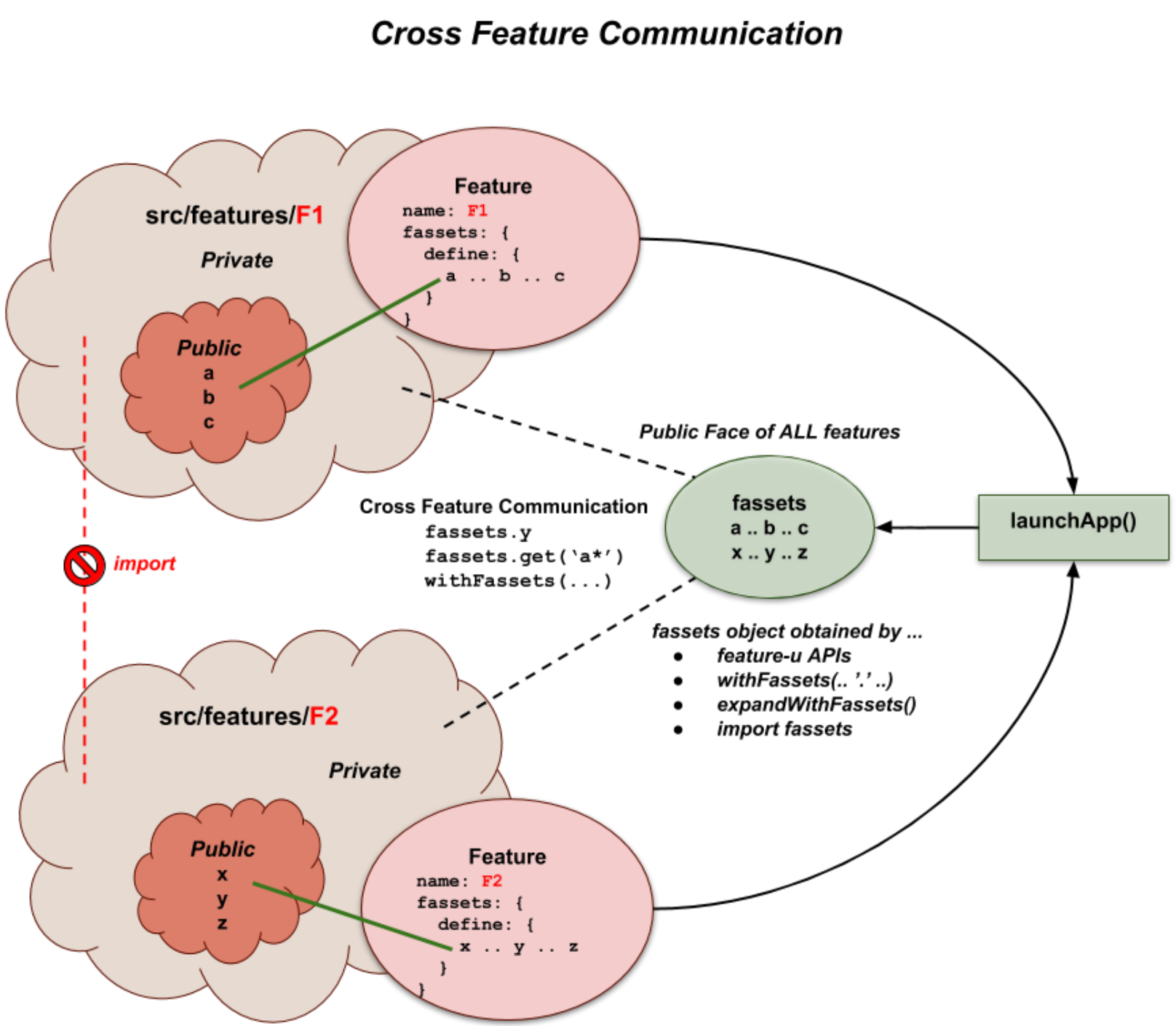
Feature Collaboration

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: {
     'openView':      actions.view.open,      // openView(viewName): Action
     'currentView':   selector.currentView,   // currentView(appState): viewName
     'isDeviceReady': selector.isDeviceReady, // isDeviceReady(appState): boolean
    },
  },

  ...
});
```

using fassets

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

NOTE: Using 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
    },
  },

  ...
});
```

injecting fasset component properties

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

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

UI Composition can be a "contract"

- Supported by additional **fasset** directives
  - **fassets.use**: specify a series of injection needs
  - **fassets.defineUse**: supply this content

- **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

```
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

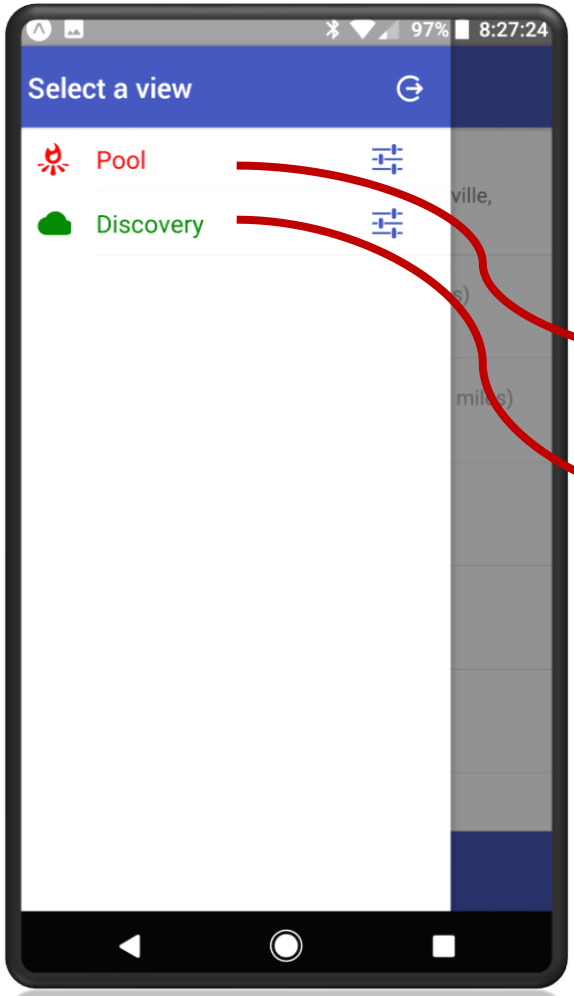
```
createFeature({
  name: 'search',

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

eatery-nod

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

UI Composition

Resource Contracts

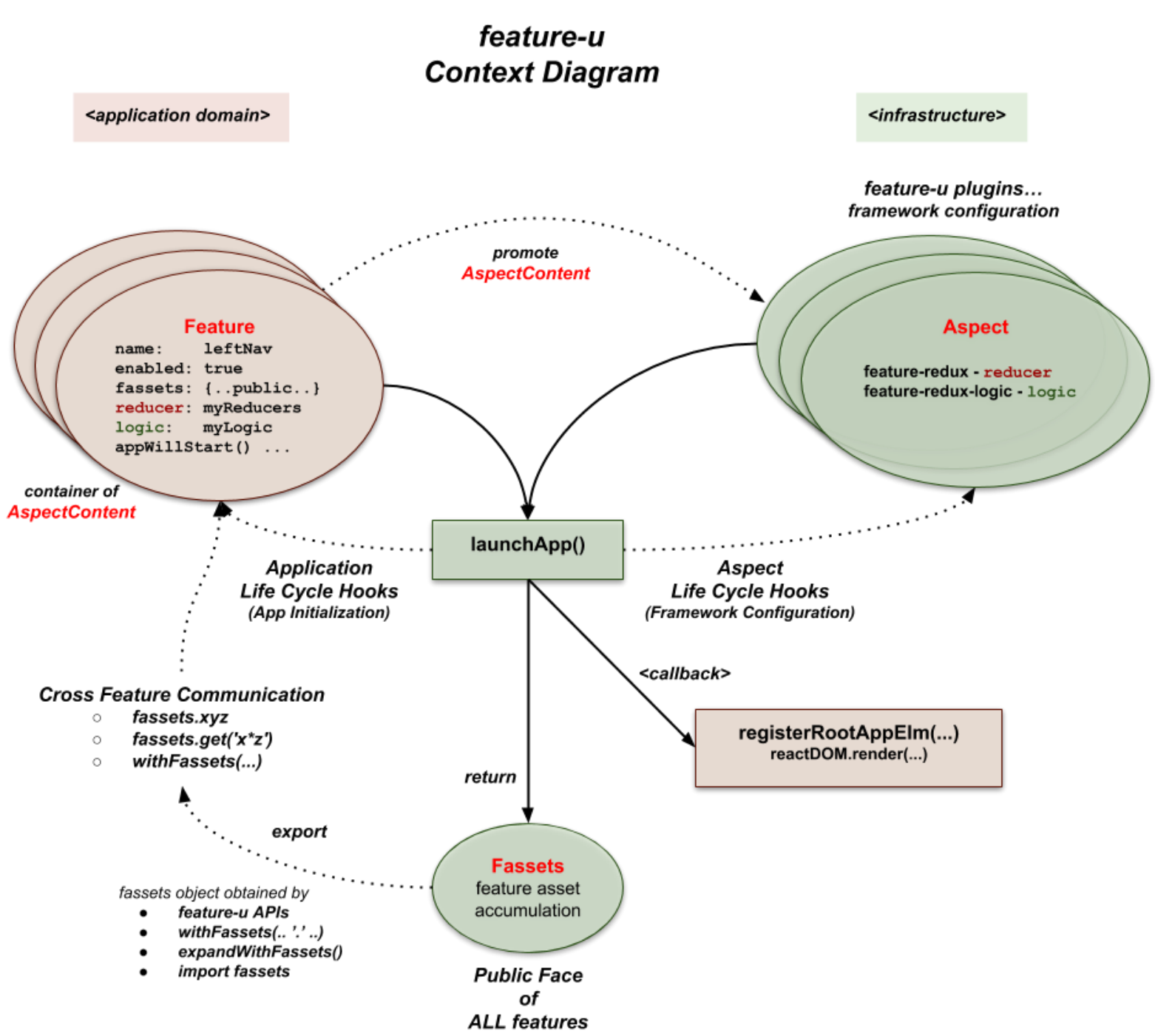


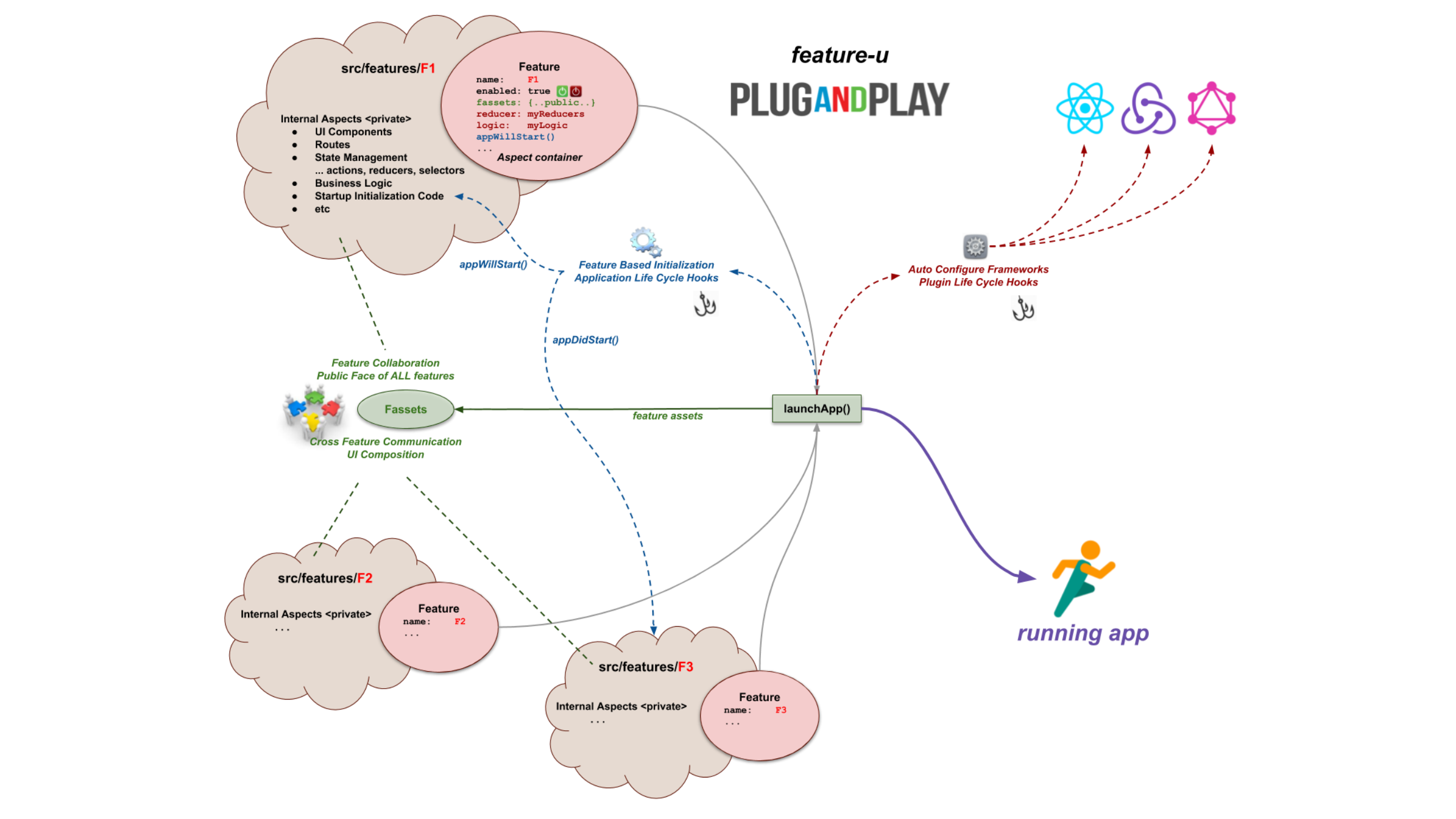
leftNav **feature**

eatery **screens**

discovery **screens**

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)
<callback>
registerRootAppElm(...)
reactDOM.render(...)
Cross Feature Communication
fassets.xyz
fassets.get('x*z')
withFassets(...)
return
export
Fassets
feature asset
accumulation
fassets object obtained by
feature-u APIs
withFassets(.. '.' ..)
expandWithFassets()
import fassets
Public Face
of
ALL 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
...