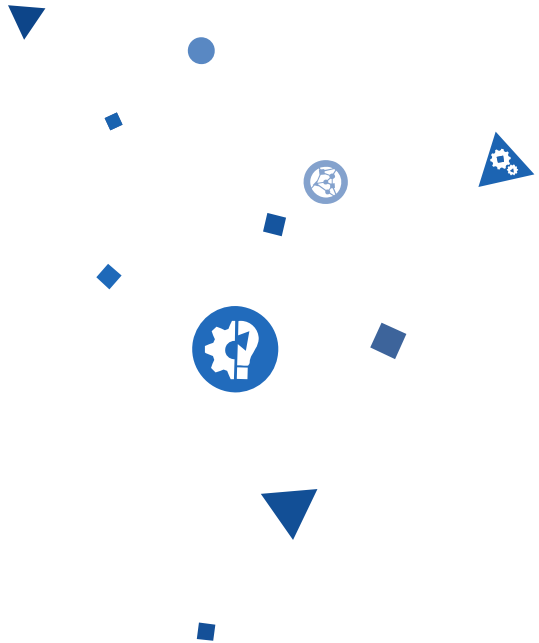


ПОРТАЛЫ НА JAVASCRIPT
ЗАЧЕМ, КАК И НАДО ЛИ ОНО

HolyJS.ru – Михаил Дружинин @ Luxoft - Июнь 2016

LUXOFT

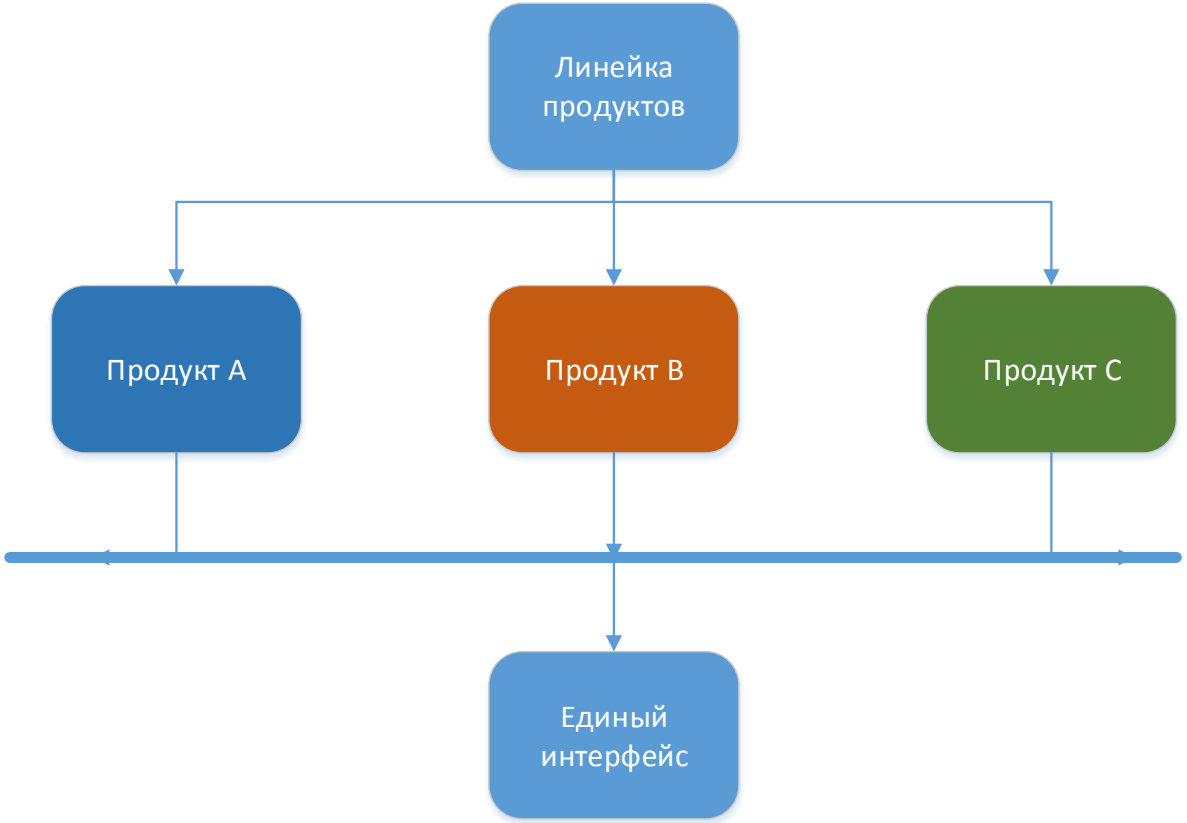
ОБО МНЕ

- Сейчас Technical Manager / Software Architect
- 10 лет в IT / 5 лет в Luxoft
- Тренер по архитектуре и Java разработке
- Опыт управления проектами в роли Team Lead и PM
- Участвовал в многих проектах в роле архитектора

ЧТО МЫ ДЕЛАЕМ

- «Поток» проектов для банковской сферы
- Типовые проекты
  - Разработка UI
  - Интеграция с API, который делается другими командами
  - AngularJS, REST API
- Агрессивные рамки выхода проектов

ПОРТАЛЬНЫЙ ПРОЕКТ



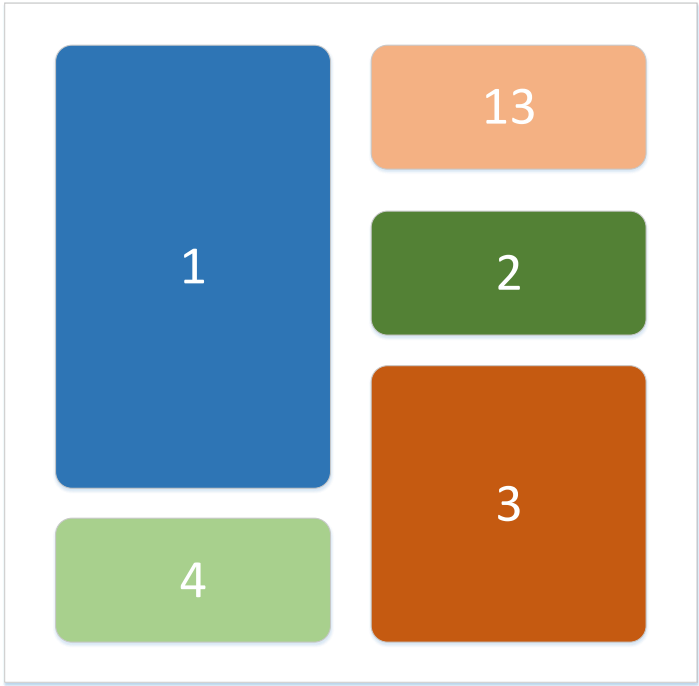

ДОБРО ПОЖАЛОВАТЬ!

О ЧЁМ ЭТО МЫ?

- Приложение с одним «общим» пользовательским интерфейсом
- Большой проект
  - состоящий из большого количества модулей
  - которые слабо связаны с друг другом

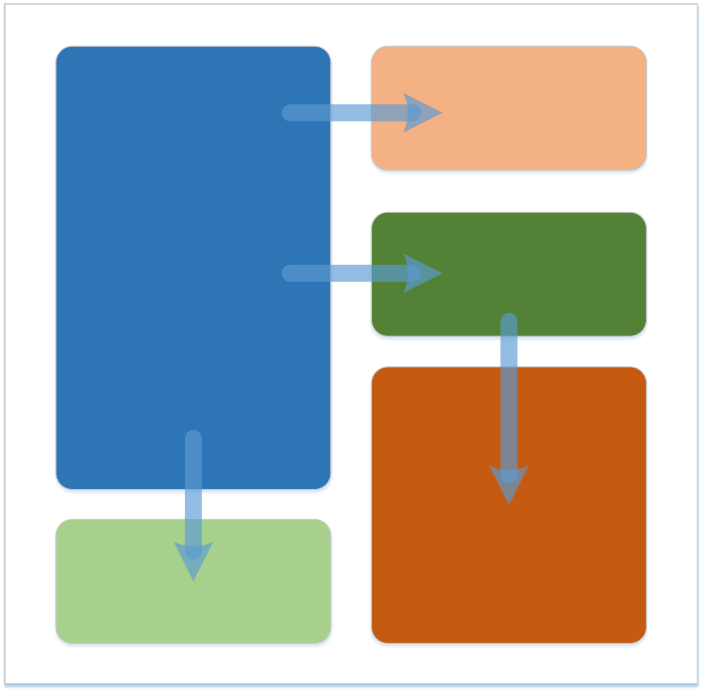
О ЧЁМ ЭТО МЫ?

- Приложение с одним «общим» пользовательским интерфейсом
- Большой проект
  - состоящий из большого количества модулей
  - которые слабо связаны с друг другом
- Разный «жизненный цикл» модулей
  - даты релизов
  - разные команды разработки

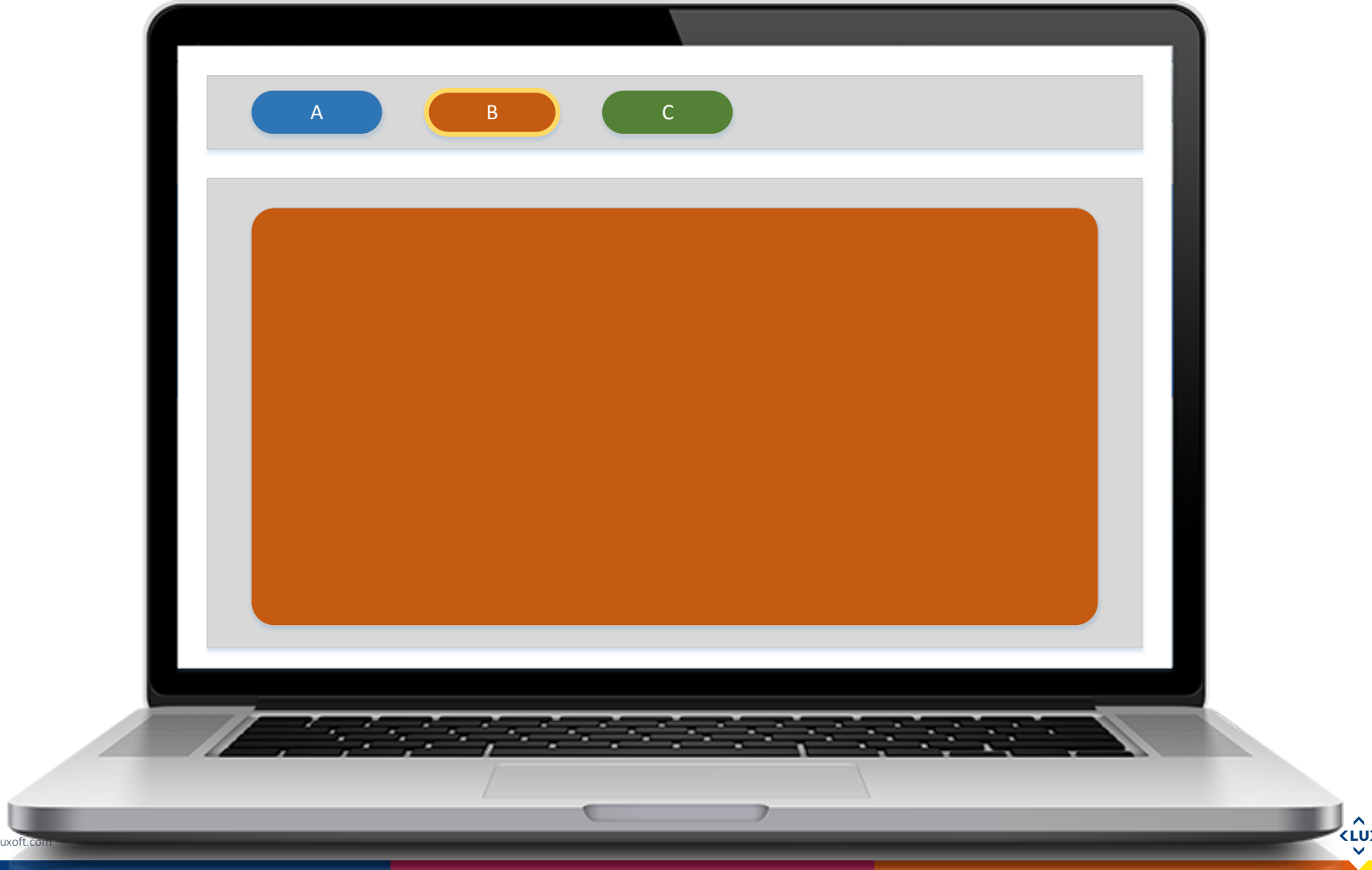


О ЧЁМ ЭТО МЫ?

- Приложение с одним «общим» пользовательским интерфейсом
- Большой проект
  - состоящий из большого количества модулей
  - которые слабо связаны с друг другом
- Разный «жизненный цикл» модулей
  - даты релизов
  - разные команды разработки
- Модули взаимодействуют между собой




A
B
C

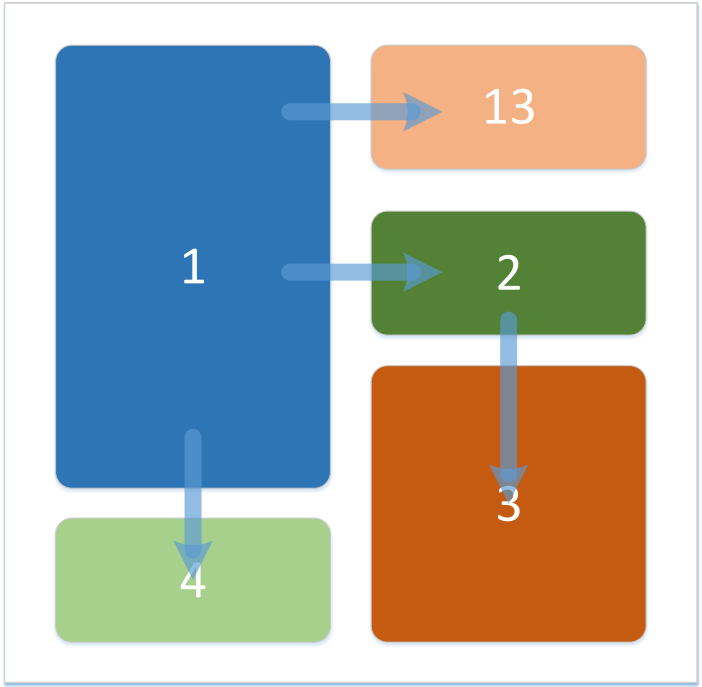

A
B
C


A
B
C


A
B
C

О ЧЁМ ЭТО МЫ?

- Приложение с одним «общим» пользовательским интерфейсом
- Большой проект
  - состоящий из большого количества модулей
  - которые слабо связаны с друг другом
- Разный «жизненный цикл» модулей
  - даты релизов
  - разные команды разработки
- Модули взаимодействуют между собой



ВАРИАНТЫ

- Статическая вёрстка и сборка - все счастливы ☺
- Генерация на сервере
- Компоновка на клиенте

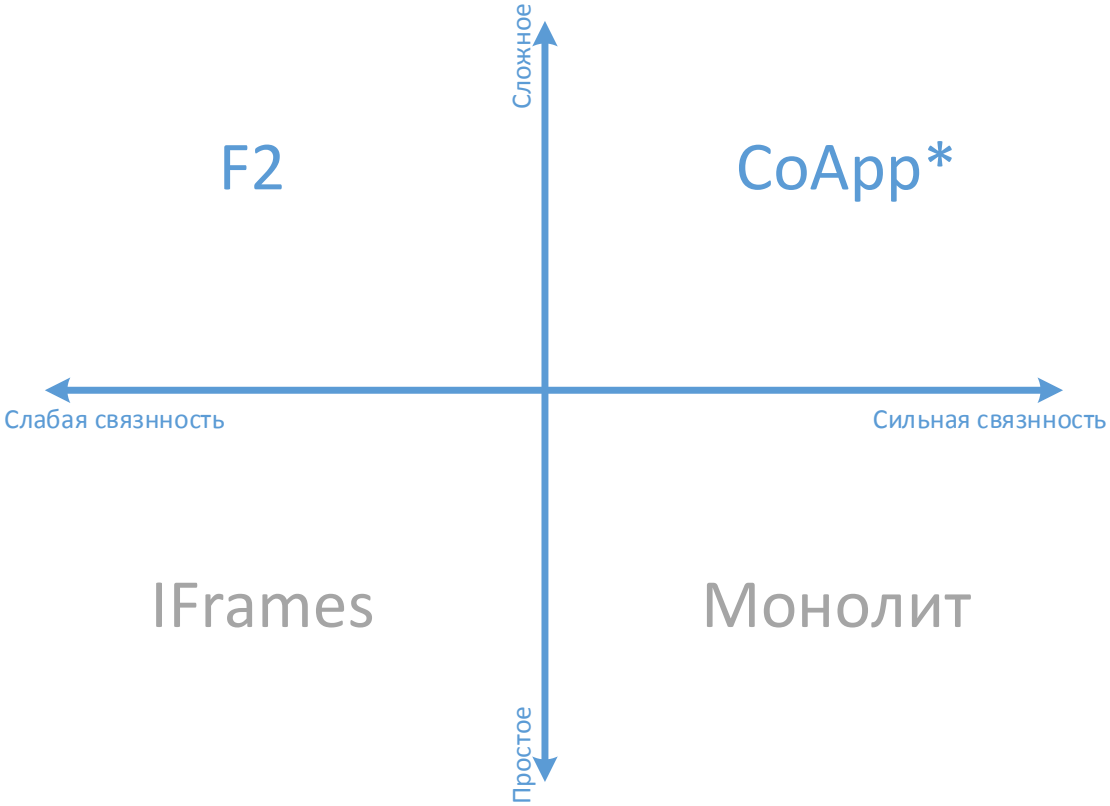
ЦЕЛЬ

- Уйти с пониманием что и когда использовать
  - Рассмотреть какие подходы можно использовать, чтобы сделать «портальное» решение используя только браузер и JavaScript
  - Обозначить плюсы и минусы каждого подхода
  - Поднять «подводные камни» на поверхность
- Обсудить какие ещё подходы возможны для решения тех же проблем


VCM
PRIMARY
ERICALLIE
CNSNEWS.COM
I'M
HAVING
SECOND
THOUGHTS

НАЧНЁМ?

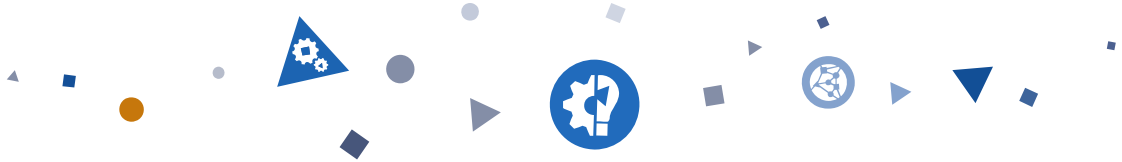
ВАРИАНТЫ


Сложное
F2
CoApp*
Слабая связнность
Сильная связнность
IFrames
Монолит
Простое

КРИТЕРИИ СРАВНЕНИЯ

- Возможность взаимодействия между модулями
- Возможность раздельного выпуска модулей

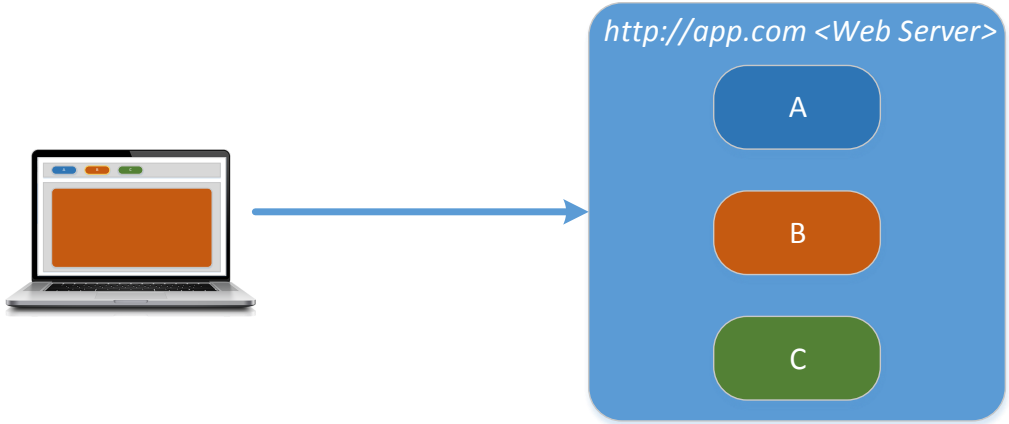
- Тестируемость
- Безопасность отдельных модулей
- Сложность разработки
- Сложность развёртывания



МОНОЛИТ

КАК ОНО ВЫГЛЯДИТ

- Отдельные компоненты и модули в рамках общей кодовой базы
- Все собирается в общий пакет для поставки



ПЛЮСЫ / МИНУСЫ - КОГДА ИСПОЛЬЗОВАТЬ И КОГДА НЕТ

Плюсы

- Простой и понятный
- Любые варианты взаимодействия между модулями

Когда использовать

- Всегда когда можете
- Когда необходимо взаимодействие между модулями

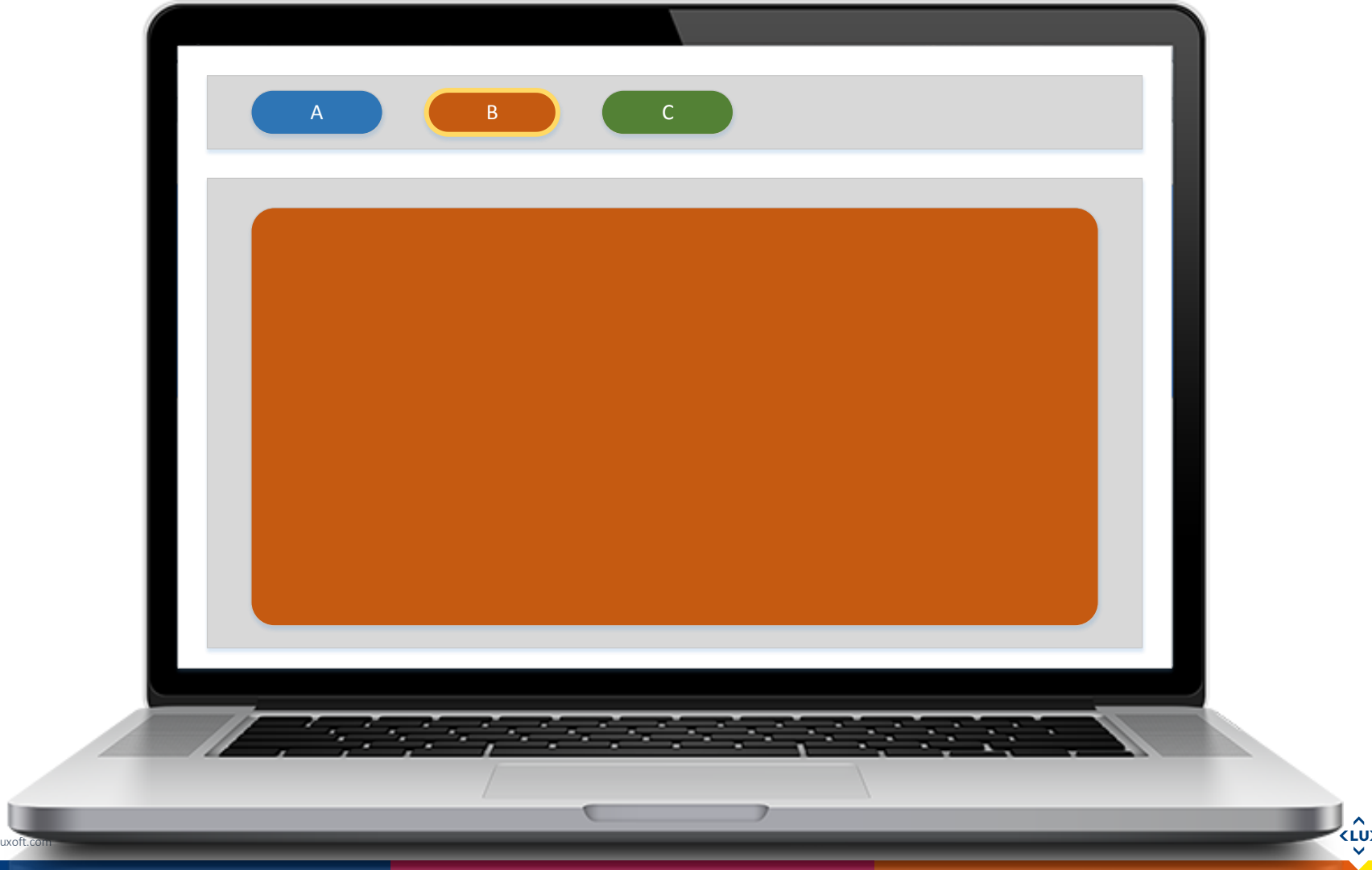
Минусы

- Выкатывать в live нужно целиком
- Изменения в одном модуле могут затронуть другие

Когда стоит отказываться

- Необходим раздельный выпуск модулей
- Необходима работа нескольких, не связанных друг с другом команд

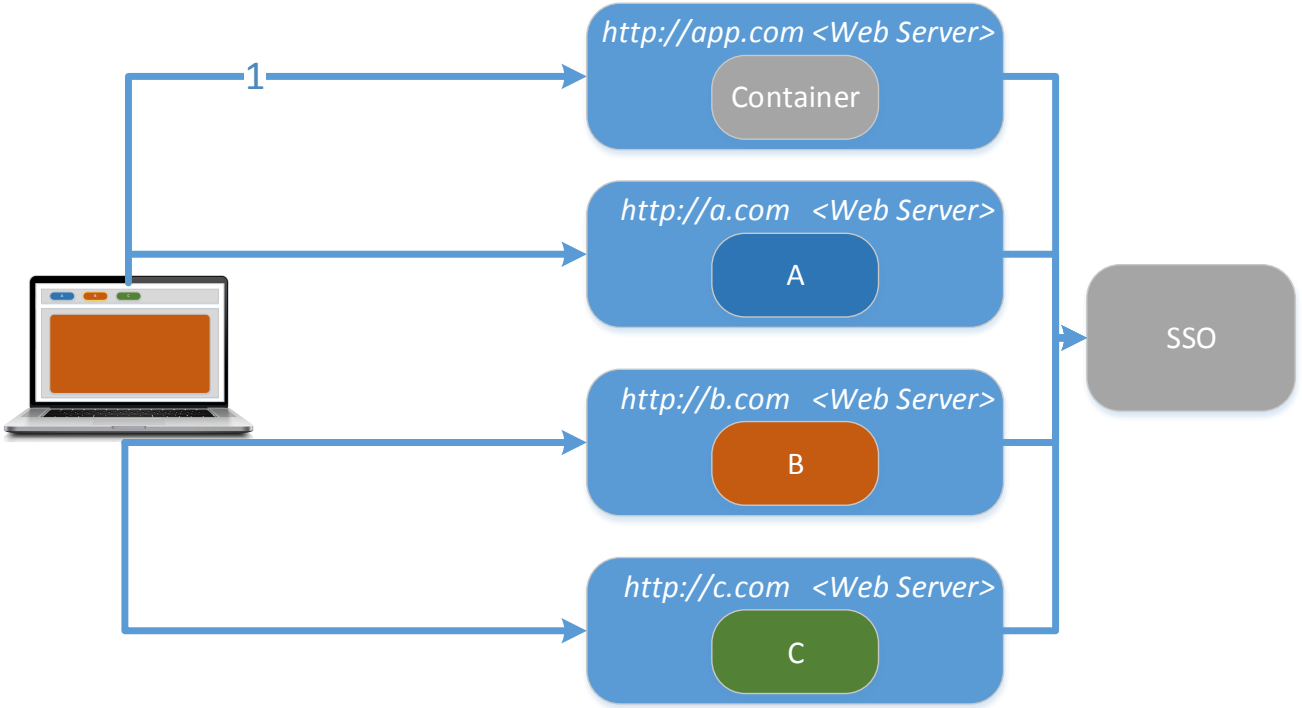
IFRAMES


A
B
C

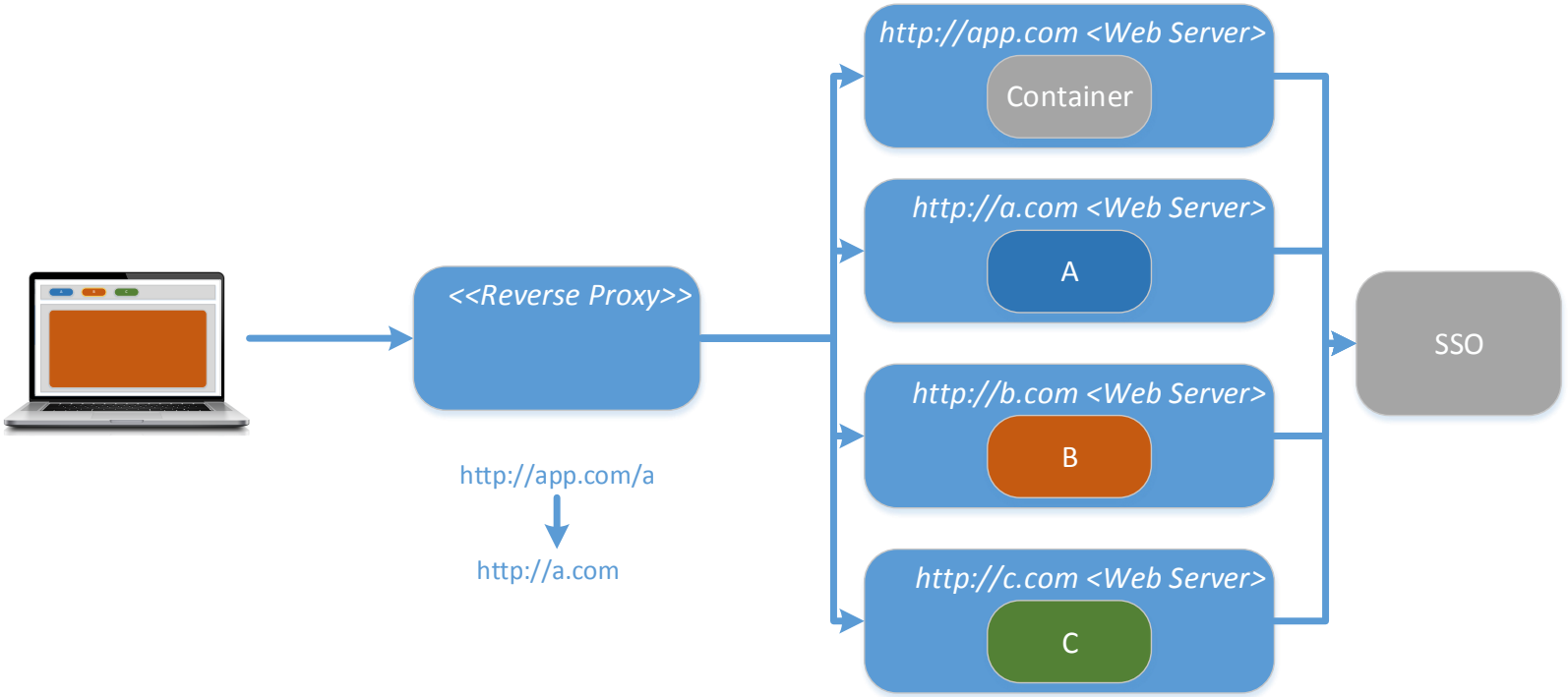
КАК ОНО ВЫГЛЯДИТ

- Проект «обёртка» который просто открывает пункт меню в IFrame
- Несколько отдельных приложений, которые разрабатываются и развёртываются независимо
- Общая авторизация

КАК ОНО ВЫГЛЯДИТ



КАК ОНО ВЫГЛЯДИТ



ПЛЮСЫ / МИНУСЫ - КОГДА ИСПОЛЬЗОВАТЬ И КОГДА НЕТ

- Плюсы
  - Простота
  - Нет влияния изменений одного модуля на другие
- Когда использовать
  - Приложения независимы – нужна только общая авторизация
- Минусы
  - Не возможно какое либо взаимодействие приложений
  - Поддержка истории
- Когда стоит отказываться
  - Необходимо взаимодействие между приложениями

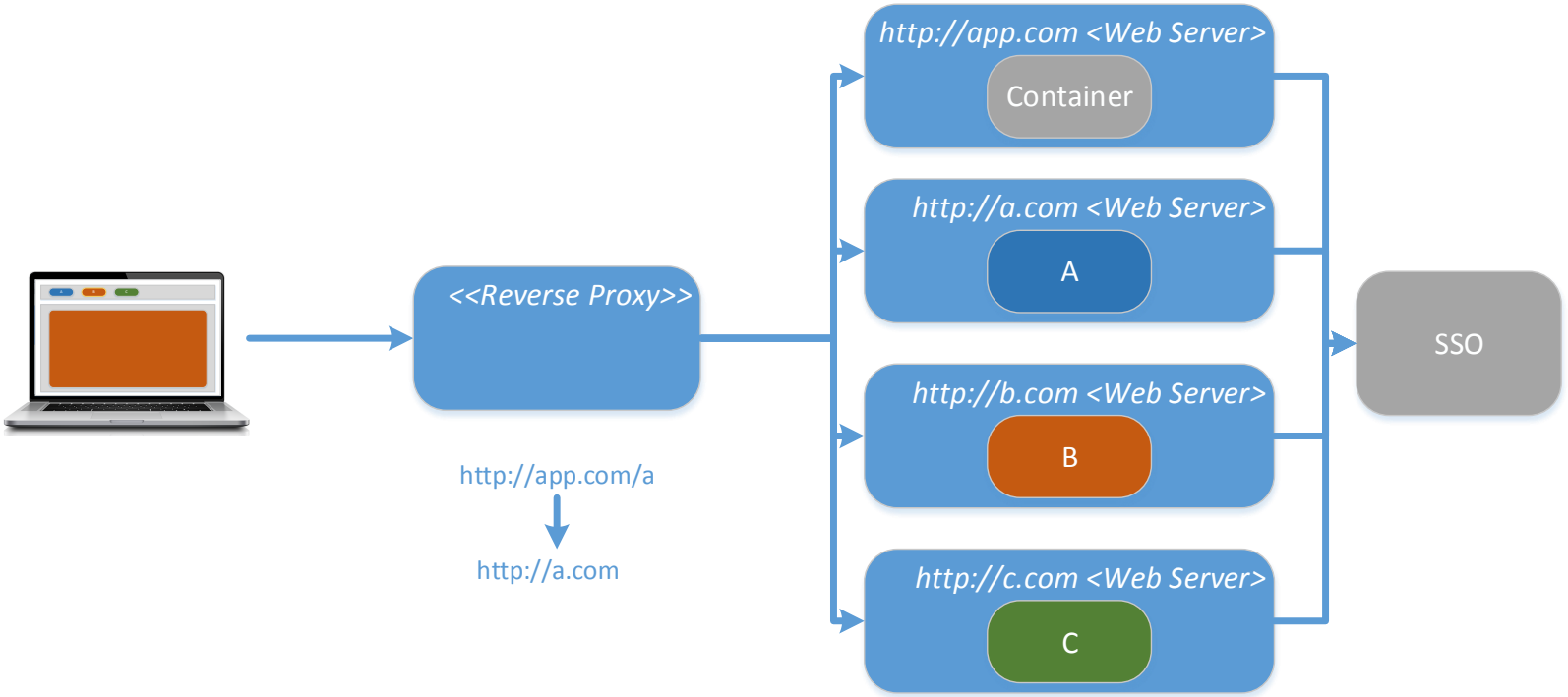
COAPP


A
B
C

КАК ОНО ВЫГЛЯДИТ

- Проект «контейнер»
- Каждое приложение собирается в два файла app.js и app.css
  - Скрипты приложений, загружаются с того же ресурса через прокси
  - Приложения после загрузки по сути представляют из себя одно AngularJS приложение
- Набор правил именования, чтобы не было коллизий имен

КАК ОНО ВЫГЛЯДИТ



КАК ОНО ВЫГЛЯДИТ

```
<html>
<head>
        <link href="/appA/app.css" rel="stylesheet" type="text/css">
        <link href="/appA/app.css" rel="stylesheet" type="text/css">
</head>
<body>
        <div ng-app="App">  <ui-view></ui-view> </div>
        <script src="/appA/app.js"></script>
        <script src="/appB/app.js"></script>
</body>
</html>
```

КАК ОНО ВЫГЛЯДИТ

```
<div>
   <coapp-list-widget select-event="'listSelected'"></coapp-list-widget>
   <div class="row">
      <div class="col-xs-6">
         <appA-price-widget update-event="'listSelected'"></appA-price-widget>
      </div>
      <div class="col-xs-6">
         <appB-volume-widget update-event="'listSelected'"></appB-volume-widget>
      </div>
   </div>
</div>
```

ПЛЮСЫ / МИНУСЫ - КОГДА ИСПОЛЬЗОВАТЬ И КОГДА НЕТ

Плюсы

- Полный свобода в том, что отдельные модули могут делать друг с другом

Минусы

- Изменения одного модуля могут затронуть другие – сложно тестировать
- Модули не защищены друг от друга

Когда использовать

- Нужна раздельная поставка модулей
- Нужен комплексный UI из нескольких модулей

Когда стоит отказываться

- Важна безопасность отдельных модулей
- Изменения модулей не должны влиять друг на друга

F2

ЧТО ЭТО

- https://www.openf2.org
- Фреймворк для загрузки в одну страницу контента с разных ресурсов
- Предоставляет функции для взаимодействия между контейнером и приложениями и между приложениями внутри контейнера


A
B
C

КАК ОНО ВЫГЛЯДИТ

- Проект «контейнер»
- Модули-«приложения» для контейнера, которые загружаются с произвольных ресурсов
  - Каждый модуль имеет «манифест», по которому F2 его загружает
- Модули могут «общаться» посредством событий
  - F2.Events.emit / F2.Events.on

МАНИФЕСТ

```
F2_jsonpCallback_ru_jug_appA_widget({
      "scripts": [ "//localhost:9001/appA/widget.js‟ ],
      "styles": [],
      "apps": [   {
          "html":
            '<div ng-app=‟ru.jug.appA. widget">' +
                  '<ru:jug:appA:widget></ru:jug:appA:widget>' +
            '</div>'
       }],
      "inlineScripts": [
            "angular.bootstrap($('.ru_jug_appA_widget'),
                  [‘ru.jug.events', ‘ru.jug.appA.widget']);"
      ]
})
```

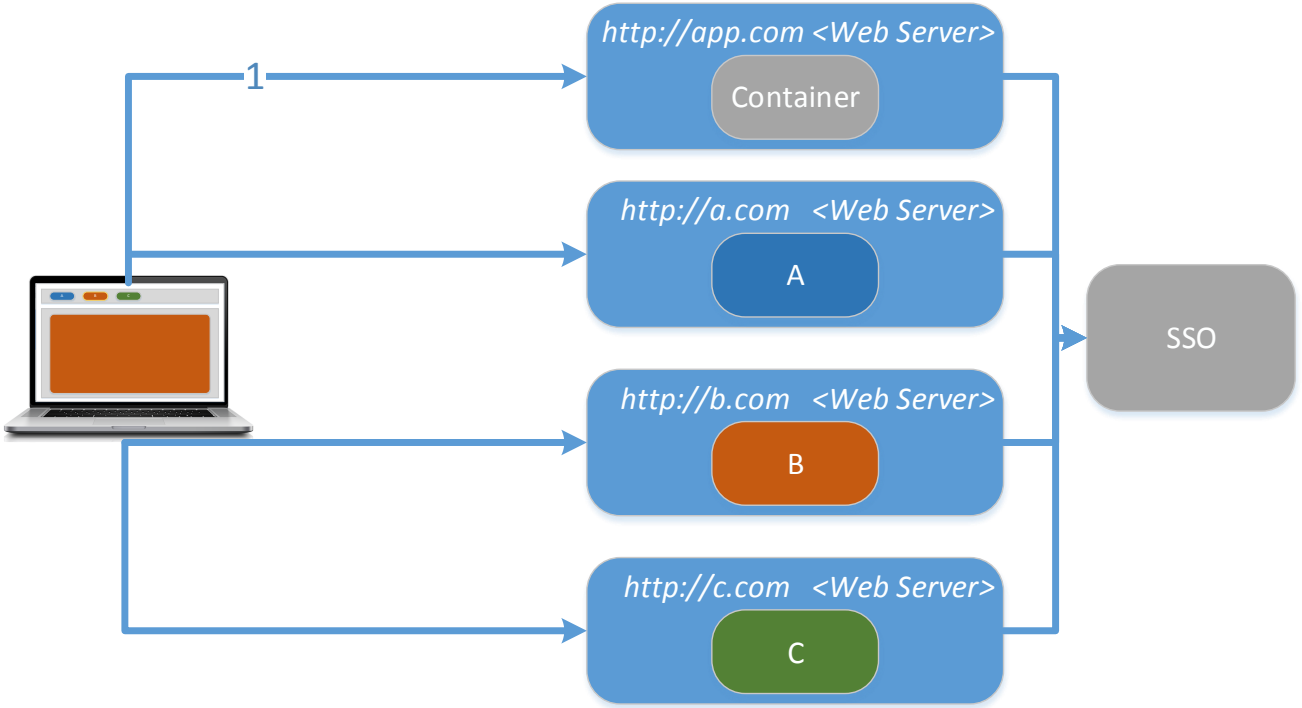
КОНТЕЙНЕР

```
<div
     data-f2-autoload
     data-f2-appid
          =“ru_jug_appA_widget"
    data-f2-manifesturl
          ="//localhost:9001/appA/f2.manifest.js">
</div>
```

КОММУНИКАЦИЯ

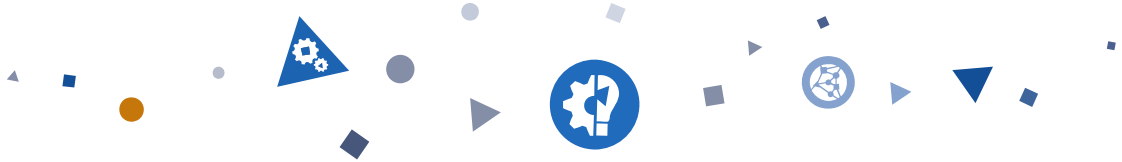
```
F2.Events.emit('event_name', value);


F2.Events.on('event_name', function(value){
    doSomething(value);
});
```

КАК ОНО ВЫГЛЯДИТ



ПЛЮСЫ / МИНУСЫ - КОГДА ИСПОЛЬЗОВАТЬ И КОГДА НЕТ

- Плюсы
  - Загрузка модулей с различных ресурсов
  - Нет влияния изменений одного модуля на другие
  - Event модель взаимодействия
- Когда использовать
  - Нужно целевую страницу собирать «из кусочков»
- «Минусы»
  - Ограниченная модель взаимодействия через события
  - Один уровень контейнер-приложение
- Когда стоит отказываться
  - Событийной модели взаимодействия недостаточно
  - Концепция приложений ограничивает требуемый UI



ЧТО ВЫБИРАЕМ?

ЧТО ВЫБИРАЕМ?

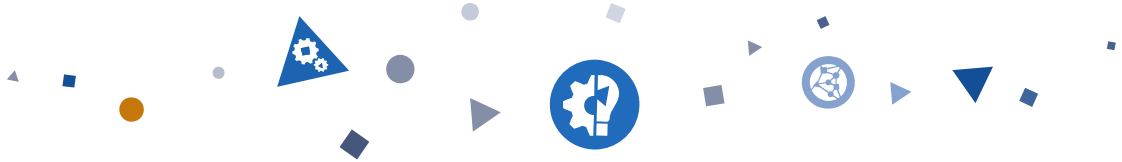
1. Используем инструмент, подходящий под цели проекта
2. Чем проще – тем лучше

ЧТО ВЫБИРАЕМ?

- Если можно – делаем одно монолитное приложение.
- Забываем про IFrames – используем F2 где нужен контент и «приложения» с различных ресурсов
- Если ничего не помогло – делаем велосипед

О ЧЁМ НУЖНО НЕ ЗАБЫТЬ ПРИ ВЫБОРЕ

- Как будем тестировать?
- Модель взаимодействия между модулями?
- Как узнать в каком приложении есть нужная функциональность?
- Откуда модули будут брать данные (API, same origin, etc…)?



УХОДИМ С LEGACY

КАК ИДТИ К СЧАСТЬЮ

1. Выделяем кусочек
2. Делаем его независимым
3. Переписываем
4. Повторить пункт 1, пока не наступило счастье

ВОПРОСЫ? ПРЕДЛОЖЕНИЯ?

СПАСИБО ЗА ВНИМАНИЕ

Михаил Дружинин
Email: mdruzhinin@luxoft.com