Rethinking of mobile app architecture

Presented by Geek-Zoo Studio 2015 @ QCON

郭虹宇 | @老郭为人民服务

https://github.com/gavinkwoe

The creator of BeeFramework & samurai-native. from Geek-Zoo Studio.

A coder, a geek, a ghost of samurai in human shell.

Review the history

1994

- 20 years ago
    - Netscape **navigator** was released.
    - People can build **web page** using **HTML+CSS**.

2008

- 7 years ago
    - Apple **iOS** was released.
    - People can build iOS **native app** using **C/OC**.

2011

- 4 years ago
    - Adobe **PhoneGap** was released.
    - People can build iOS **hybrid app** using **HTML+CSS**.

2013

- 2 years ago
    - GeekZoo **BeeFramework** was released.
    - People can build iOS **hybrid app** using **XML+CSS**.

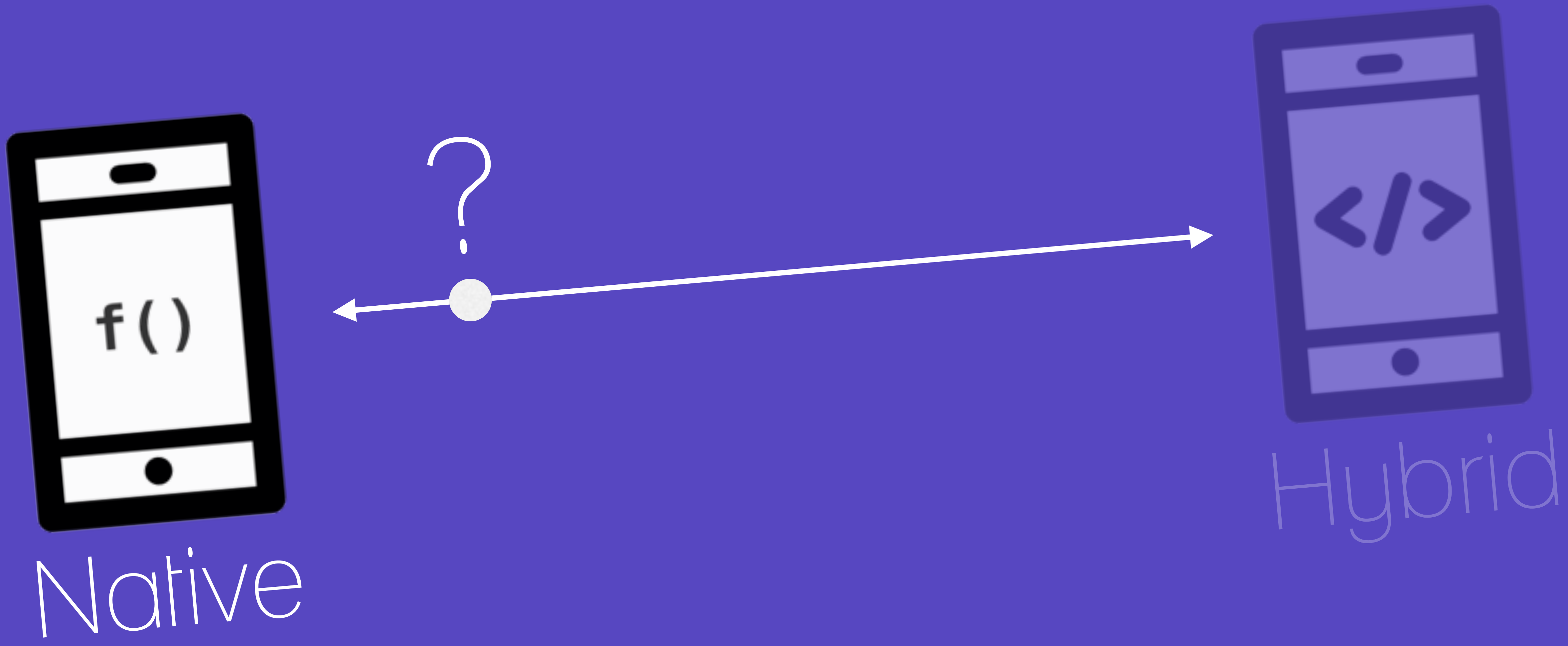
2015

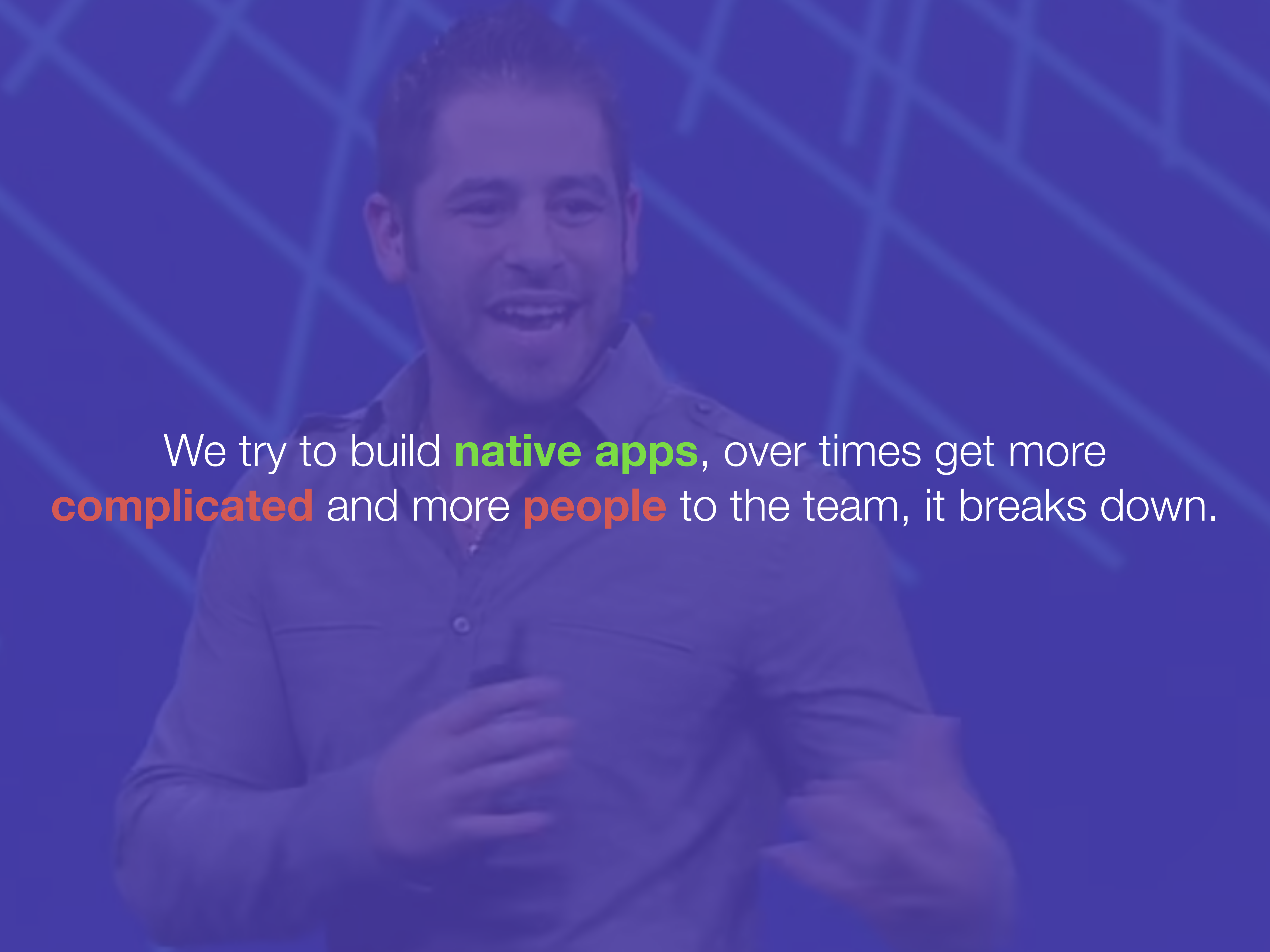
- 1 month ago
  - Facebook **react-native** was released.
  - People can build iOS **hybrid app** using **JS+CSS**.

Today

We need to re-thinking about app architecture …

Problem

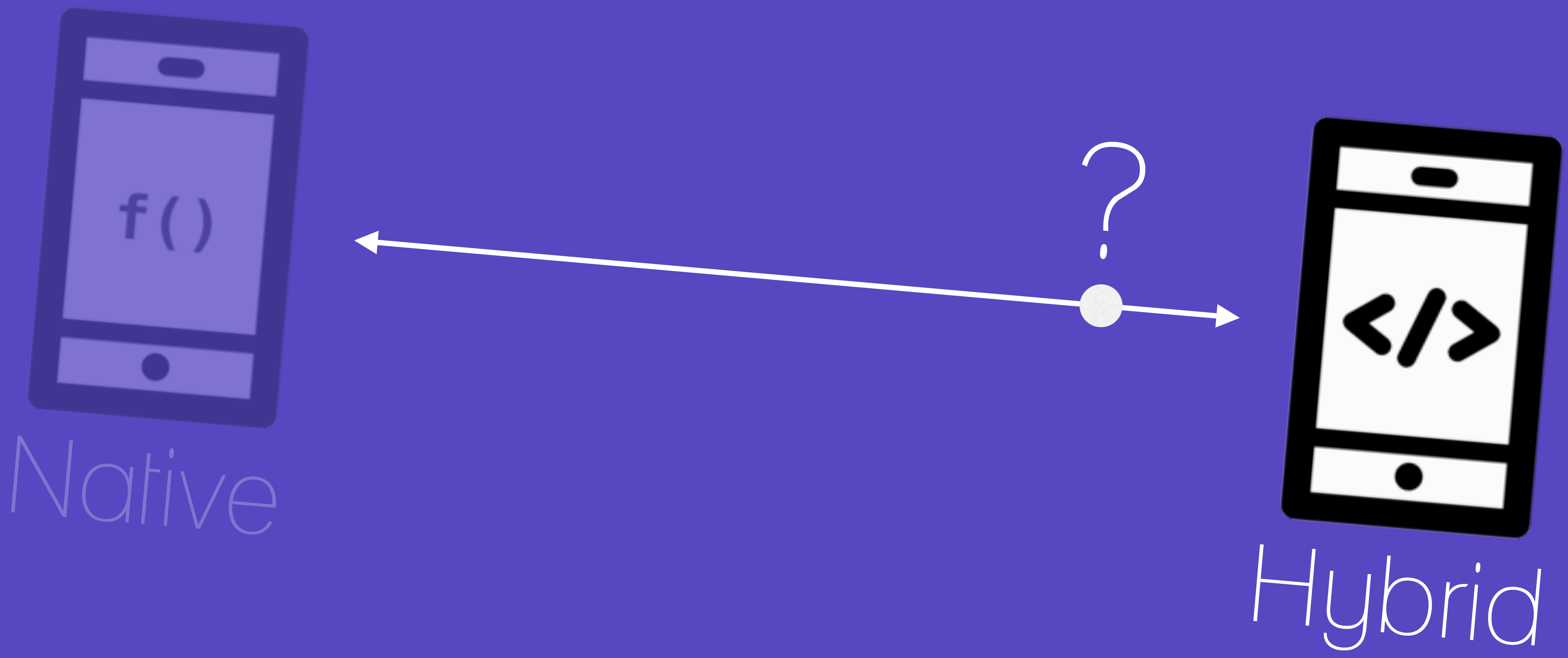

f()
?
</>
Native
Hybrid

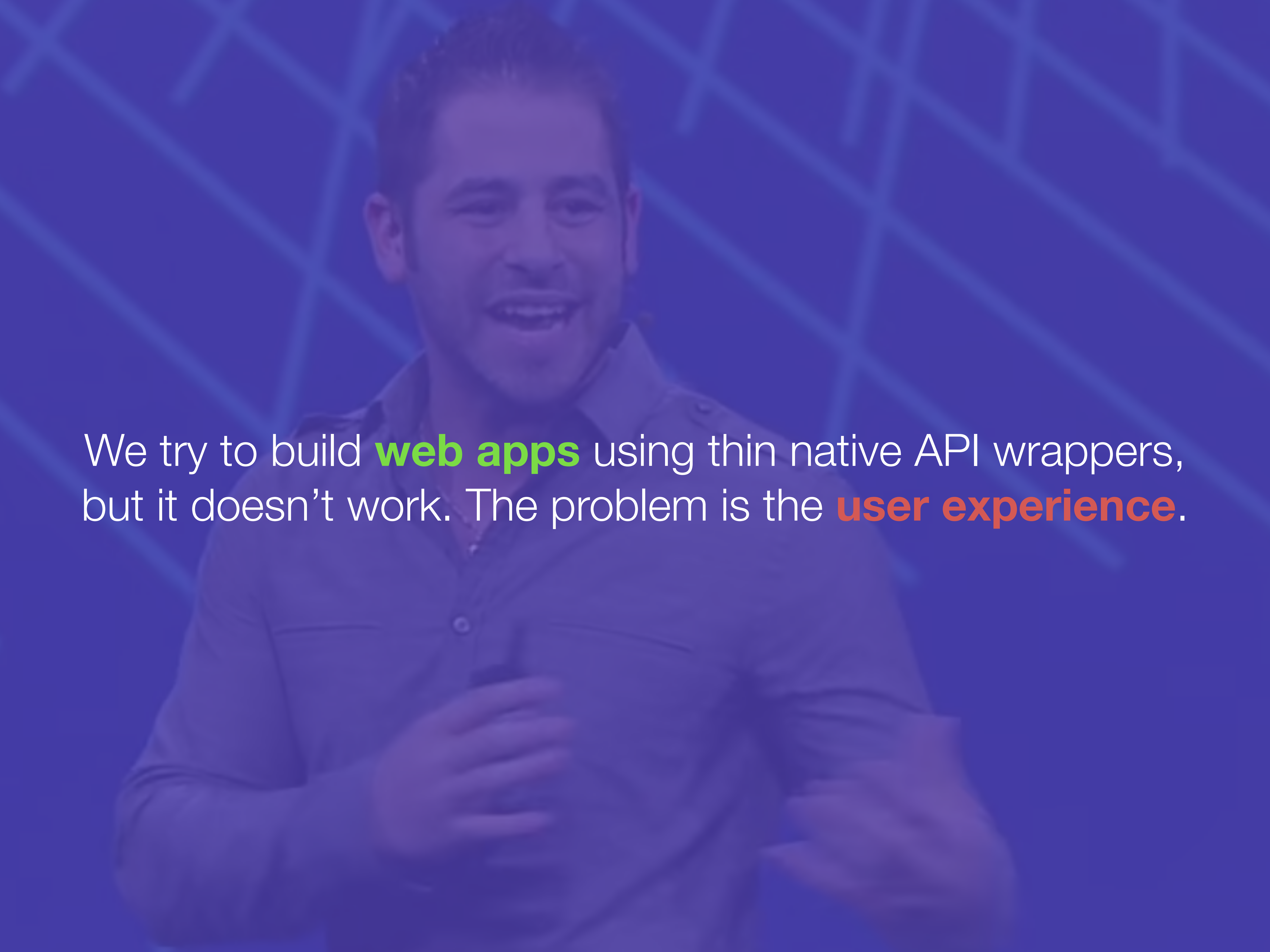


We try to build **native apps**, over times get more **complicated** and more **people** to the team, it breaks down.

Native

- Well
  - Good user experience
  - Low level API
- Less well
  - Hard to deploy
  - High learning cost


f()
Native
?
</>
Hybrid

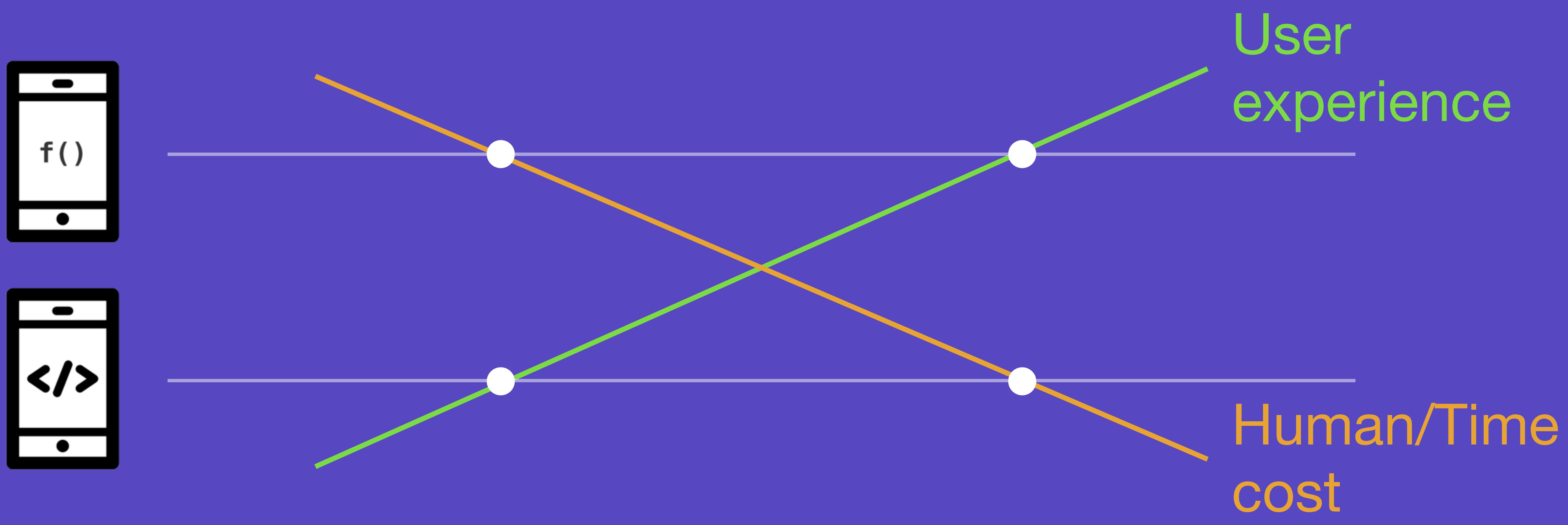


We try to build **web apps** using thin native API wrappers, but it doesn't work. The problem is the **user experience**.

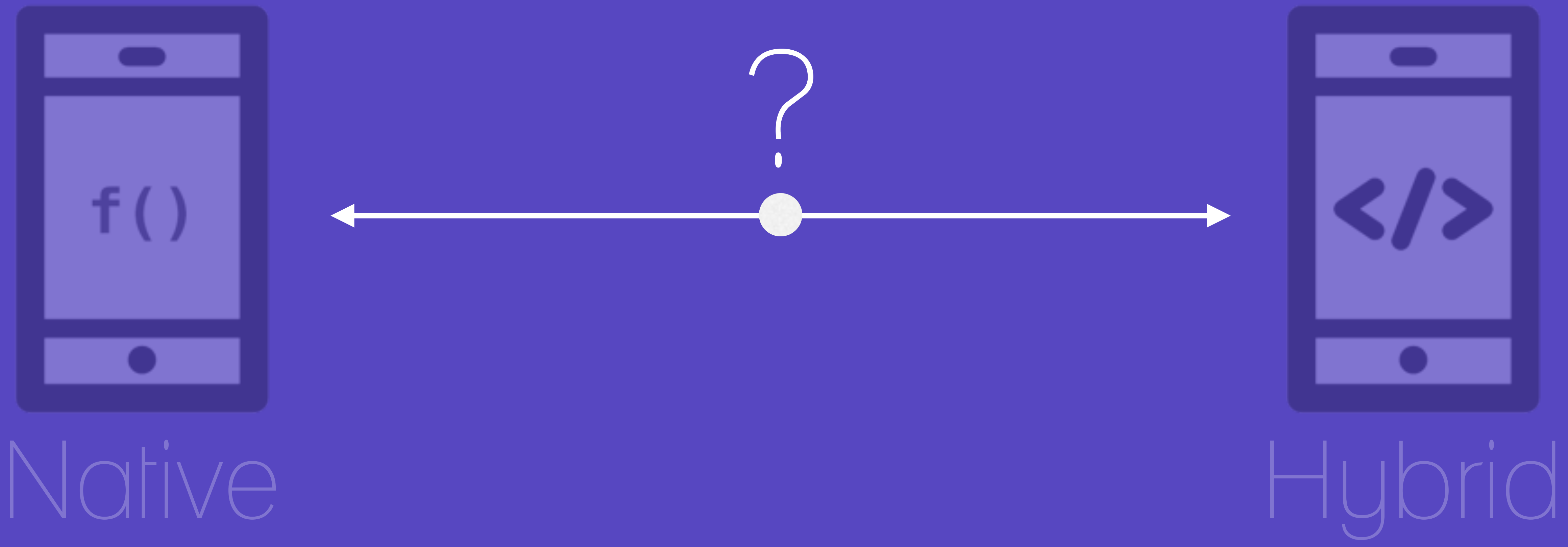
Hybrid

- Well
  - Cross platform
  - Easy to deploy
- Less well
  - Bad user experience
  - Low learning cost

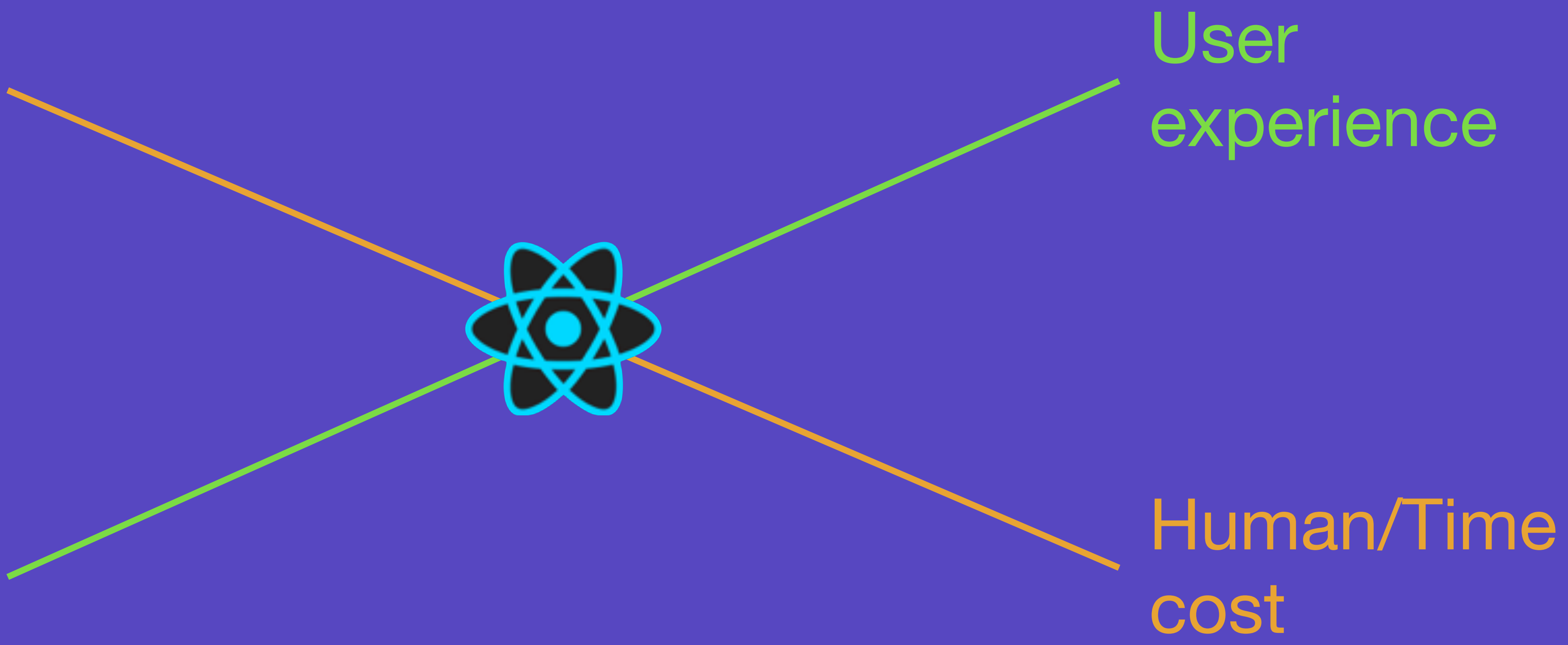
The Fish & Bear problem



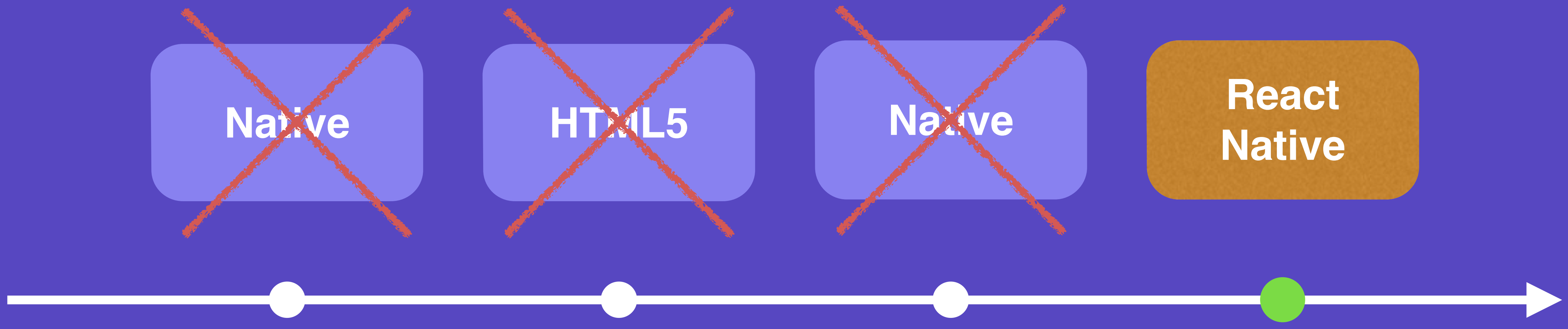
Native + Hybrid ?

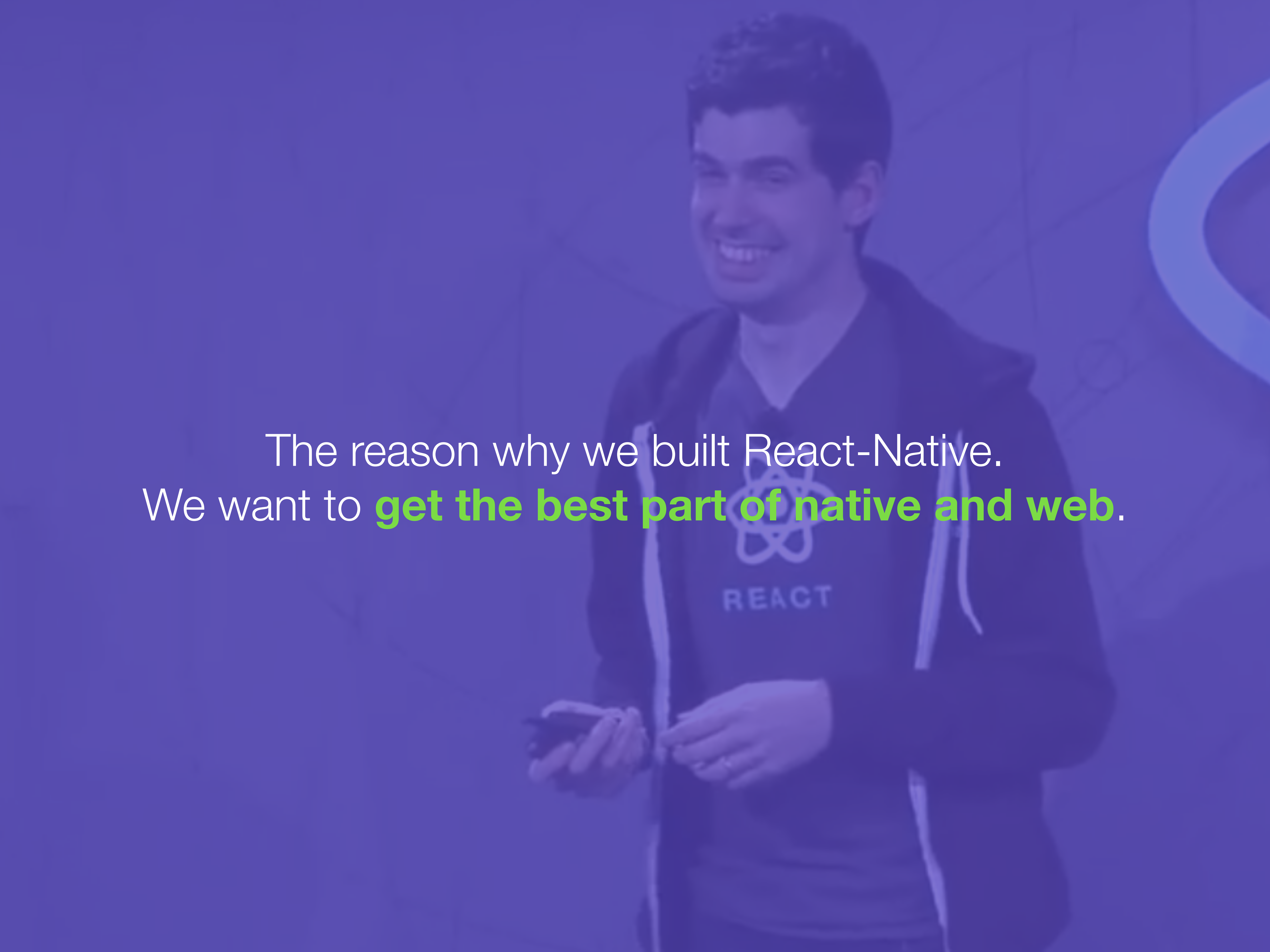


The Fish & Bear answer



Semi-Hybrid solution



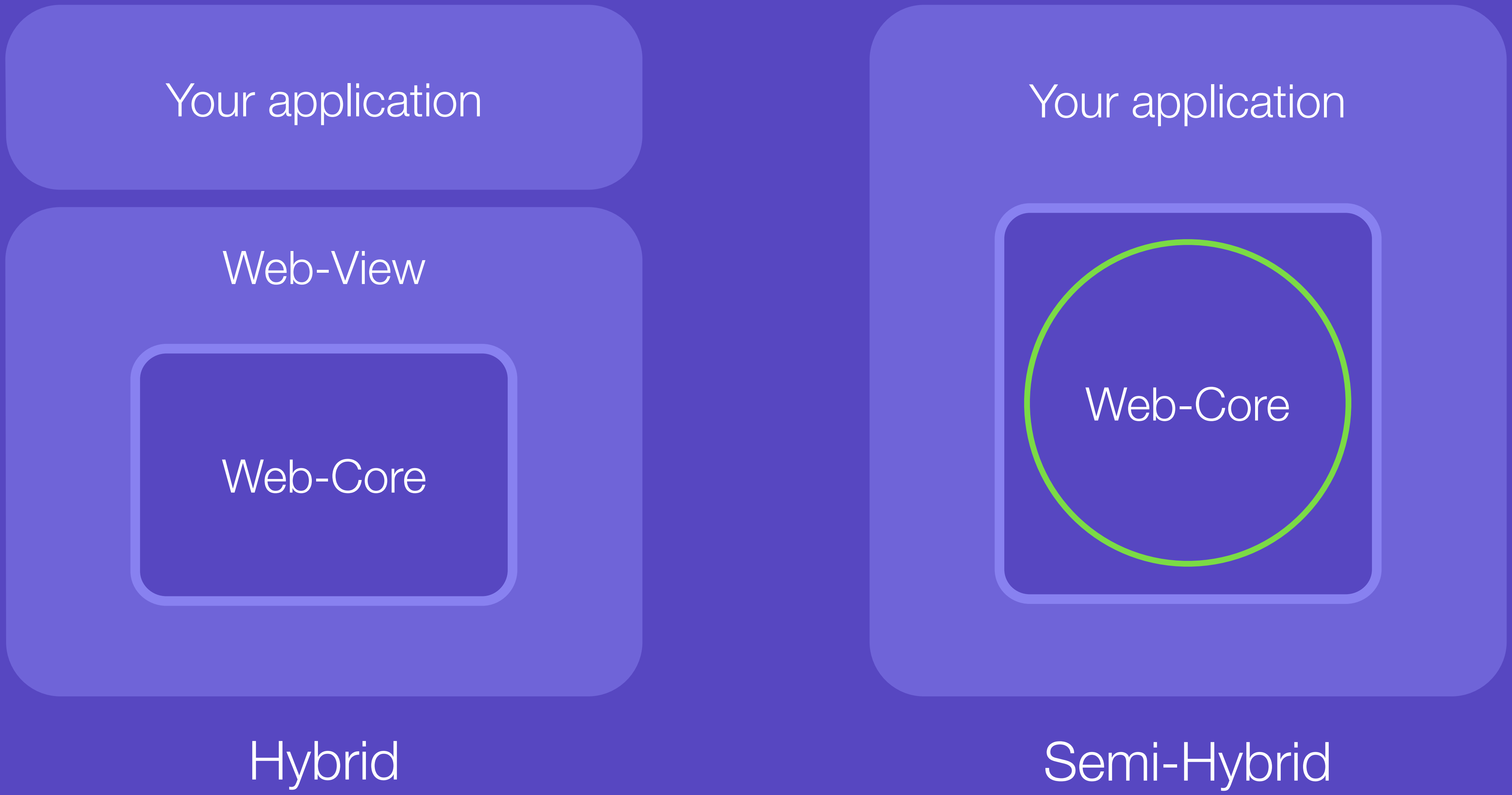


The reason why we built React-Native.
We want to **get the best part of native and web**.

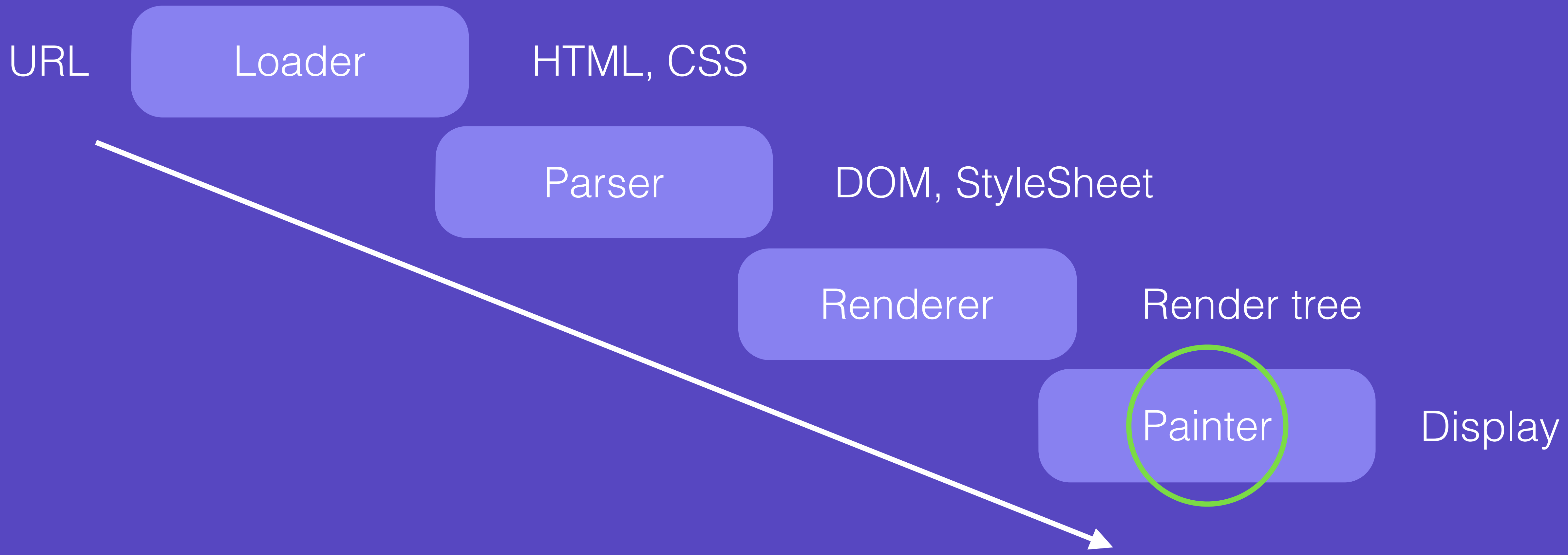
The mission is to build a good enough **native framework** + **web-core**.

We call it ‘Semi-Hybrid’

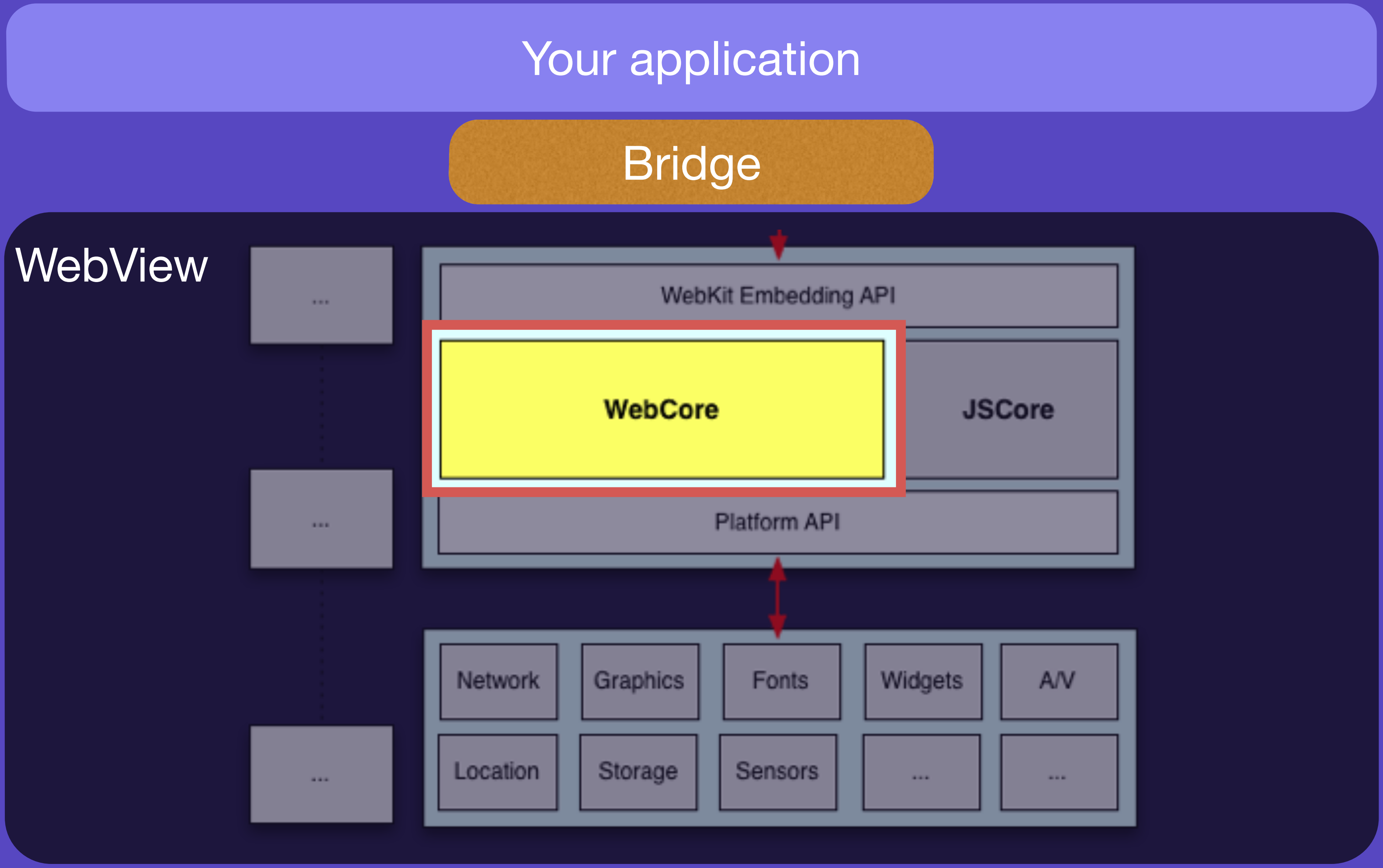
Hybrid vs Semi-Hybrid



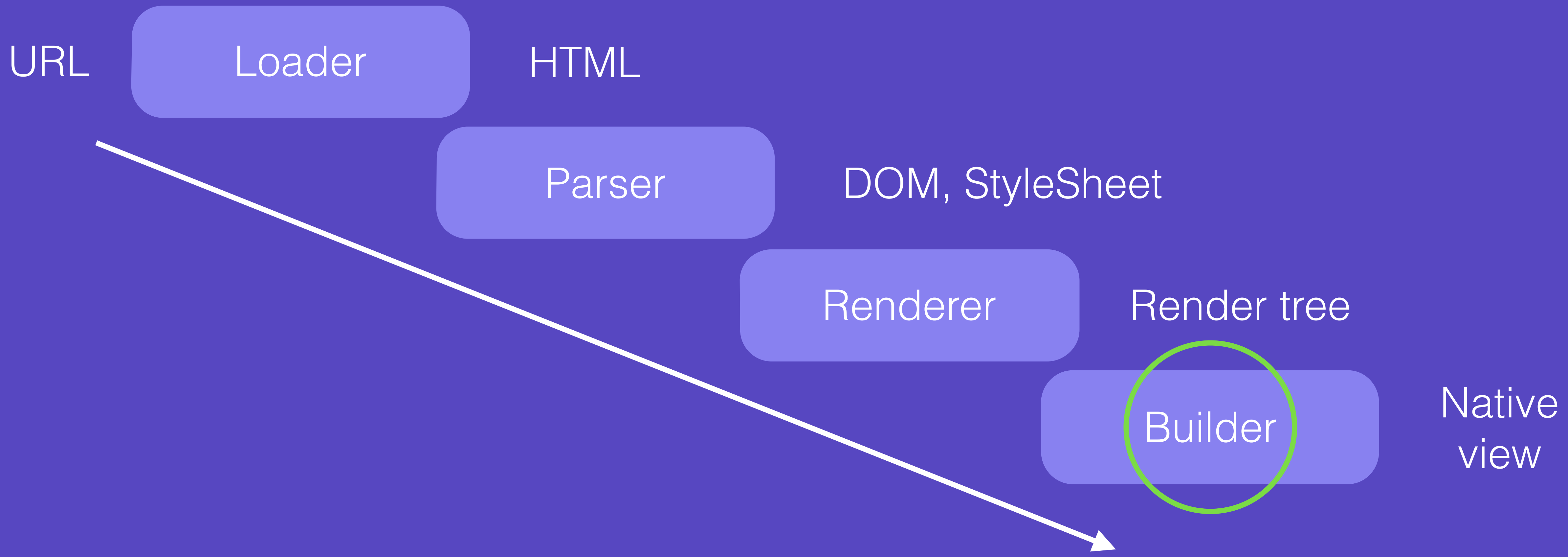
Traditional hybrid app



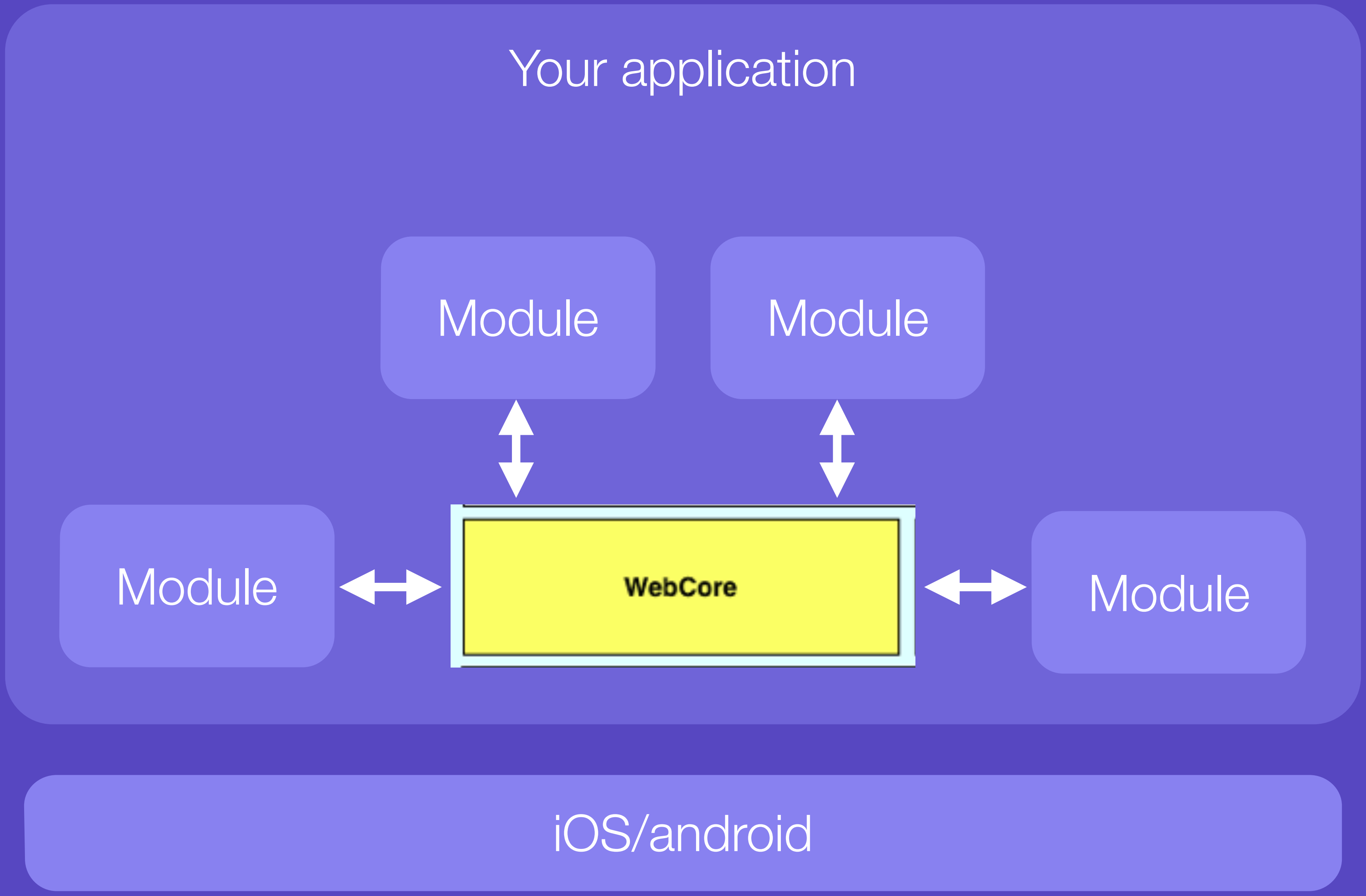
Traditional hybrid app



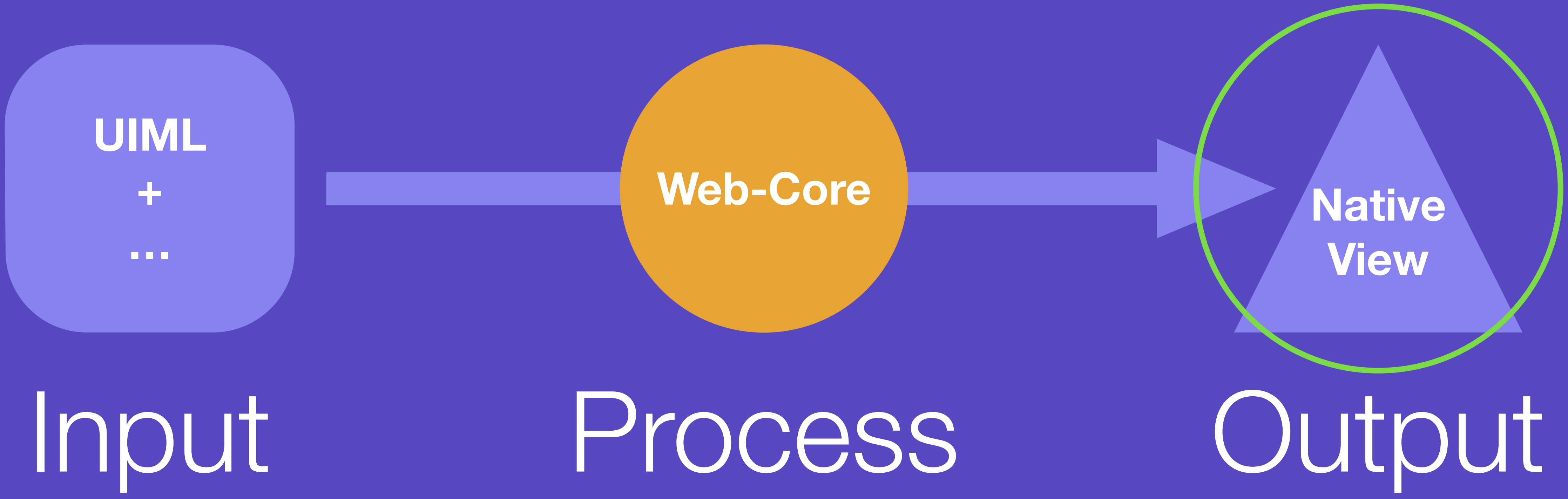
Semi-hybrid app



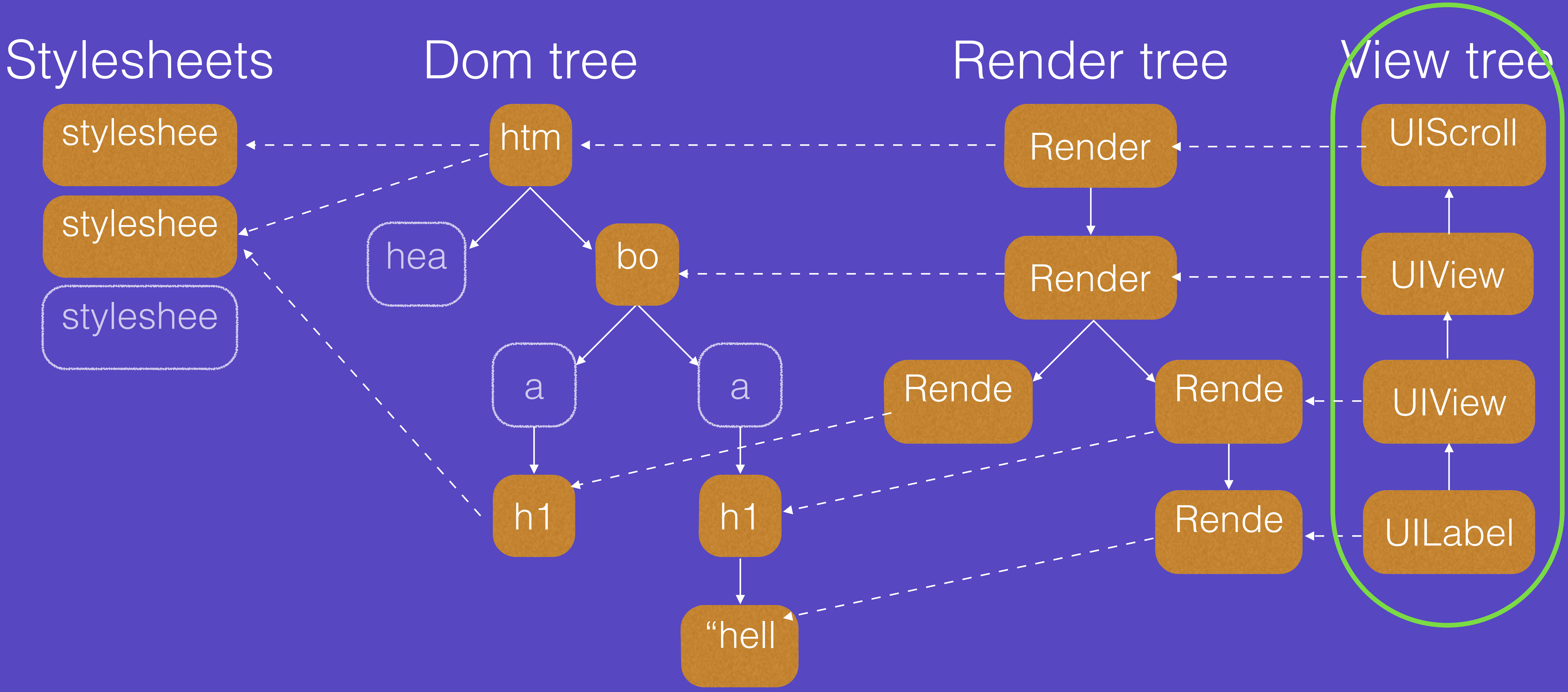
Semi-hybrid app



The key difference



The key difference



Well

- Good user experience
- Good performance
- Good expansibility
- Rapid development (HTML+CSS)
- Easy to deploy, and easy to share
- Low level API

Less well

- High R&D costs (WebCore)
- High maintenance costs (WebCore)
- Stack is too deep
- Need to learn basic front-end & basic iOS knowledge

Features of Semi-Hybrid

- Support **HTML/CSS** or other **UIML** syntax
- Support **native** components
- Support **gesture** handling
- Remote update & live reload

New things

2015

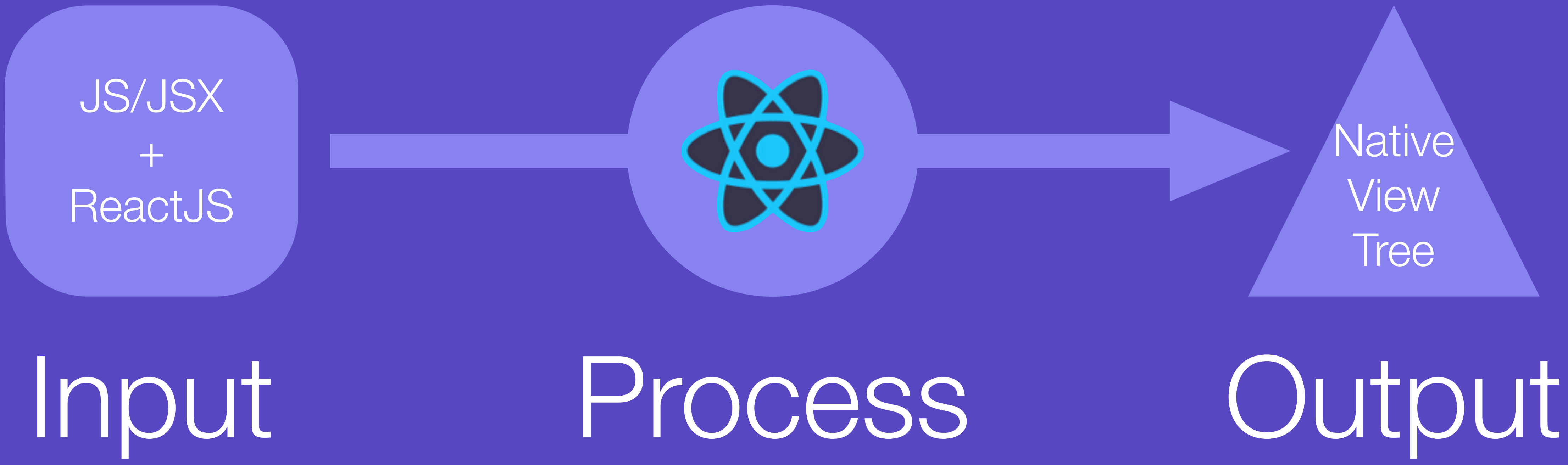
SAMURAI

2015

The similarity

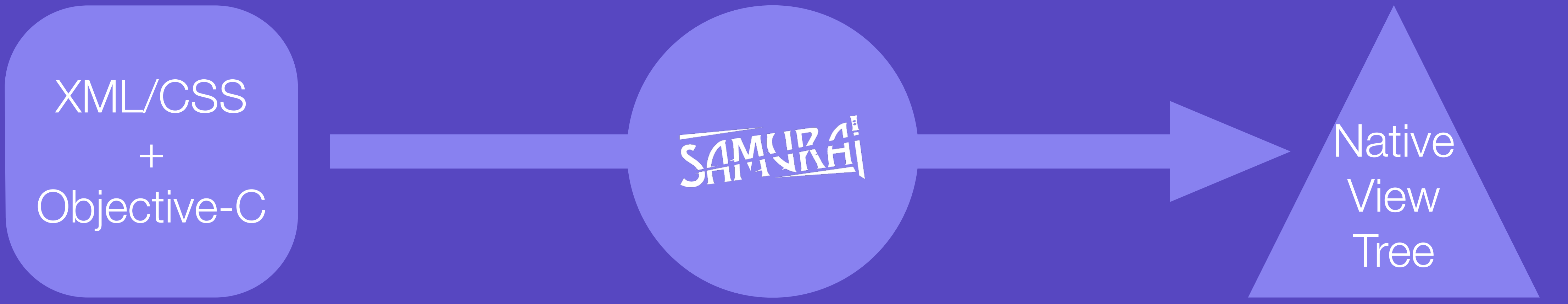
- Private Web-Core (No WebView)
- Support HTML+CSS or UIML
- Support native components
- UIKit as a backend

react-native



React UIML = JS/JSX + CSS-layout

samurai-native



Input Process Output

Samurai UIML = Standard HTML + Standard CSS

3 pillar

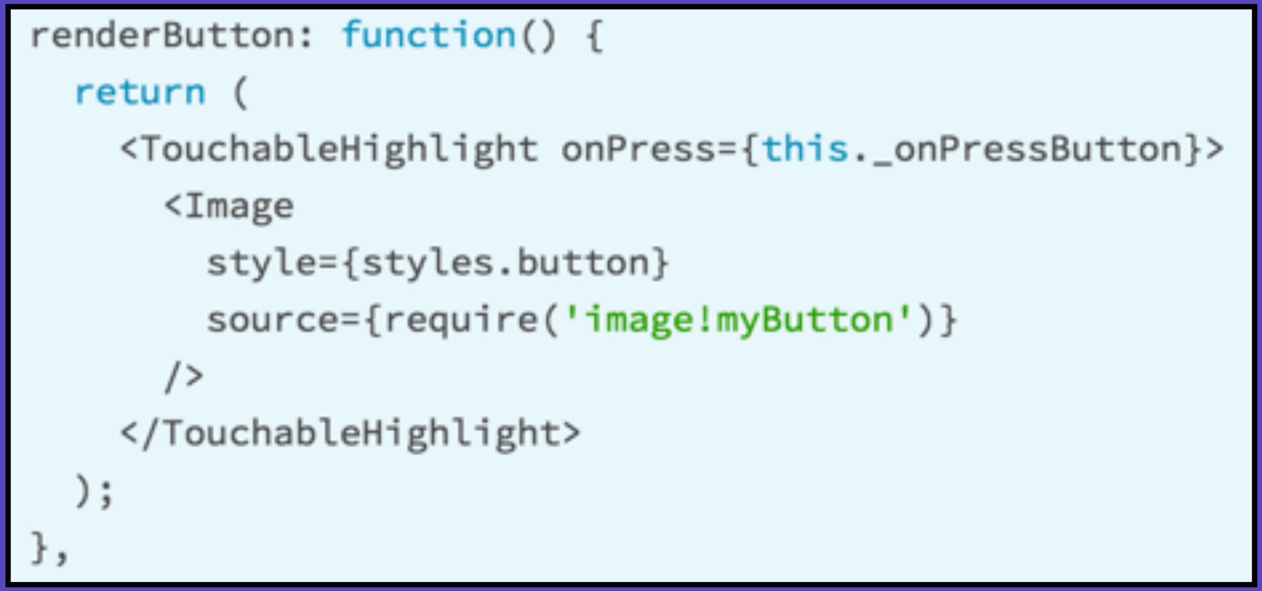
Style & Layout

Touch Handling

Native Components

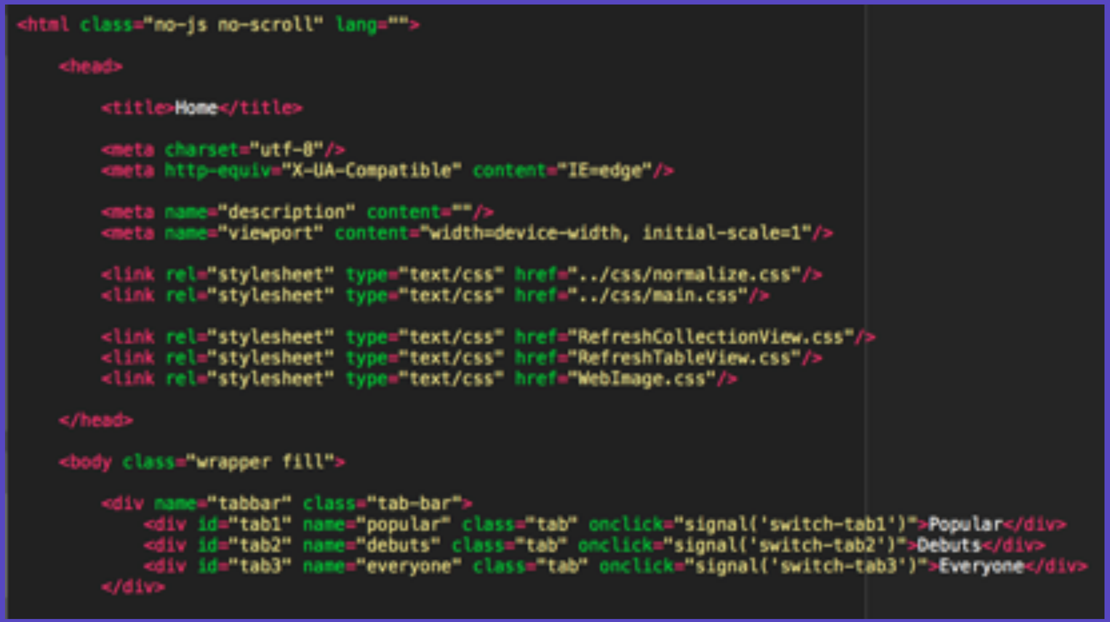
Style & layout

JSX + CSS-layout



```
renderButton: function() {
  return (
    <TouchableHighlight onPress={this._onPressButton}>
      <Image
        style={styles.button}
        source={require('image!myButton')}
      />
    </TouchableHighlight>
  );
},
```

HTML + CSS2/3



```
<html class="no-js no-scroll" lang="">

    <head>

        <title>Home</title>

        <meta charset="utf-8"/>
        <meta http-equiv="X-UA-Compatible" content="IE=edge"/>

        <meta name="description" content=""/>
        <meta name="viewport" content="width=device-width, initial-scale=1"/>

        <link rel="stylesheet" type="text/css" href="../css/normalize.css"/>
        <link rel="stylesheet" type="text/css" href="../css/main.css"/>

        <link rel="stylesheet" type="text/css" href="RefreshCollectionView.css"/>
        <link rel="stylesheet" type="text/css" href="RefreshTableView.css"/>
        <link rel="stylesheet" type="text/css" href="WebImage.css"/>

    </head>

    <body class="wrapper fill">

        <div name="tabbar" class="tab-bar">
            <div id="tab1" name="popular" class="tab" onclick="signal('switch-tab1')">Popular</div>
            <div id="tab2" name="debuts" class="tab" onclick="signal('switch-tab2')">Debuts</div>
            <div id="tab3" name="everyone" class="tab" onclick="signal('switch-tab3')">Everyone</div>
        </div>
```

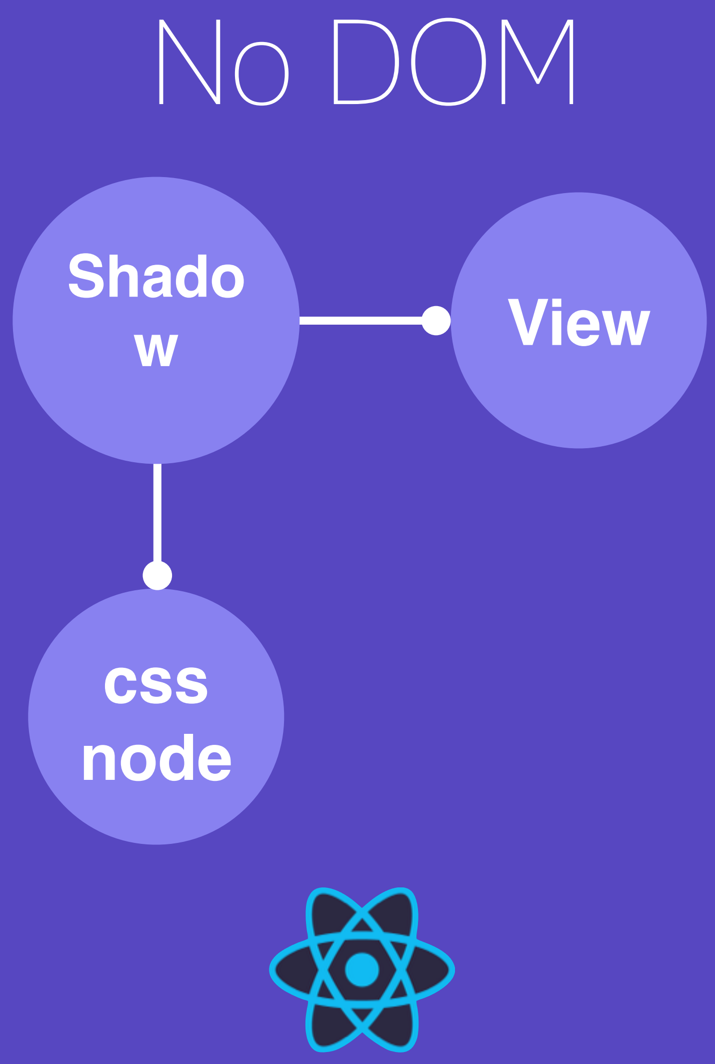
SAMURAI

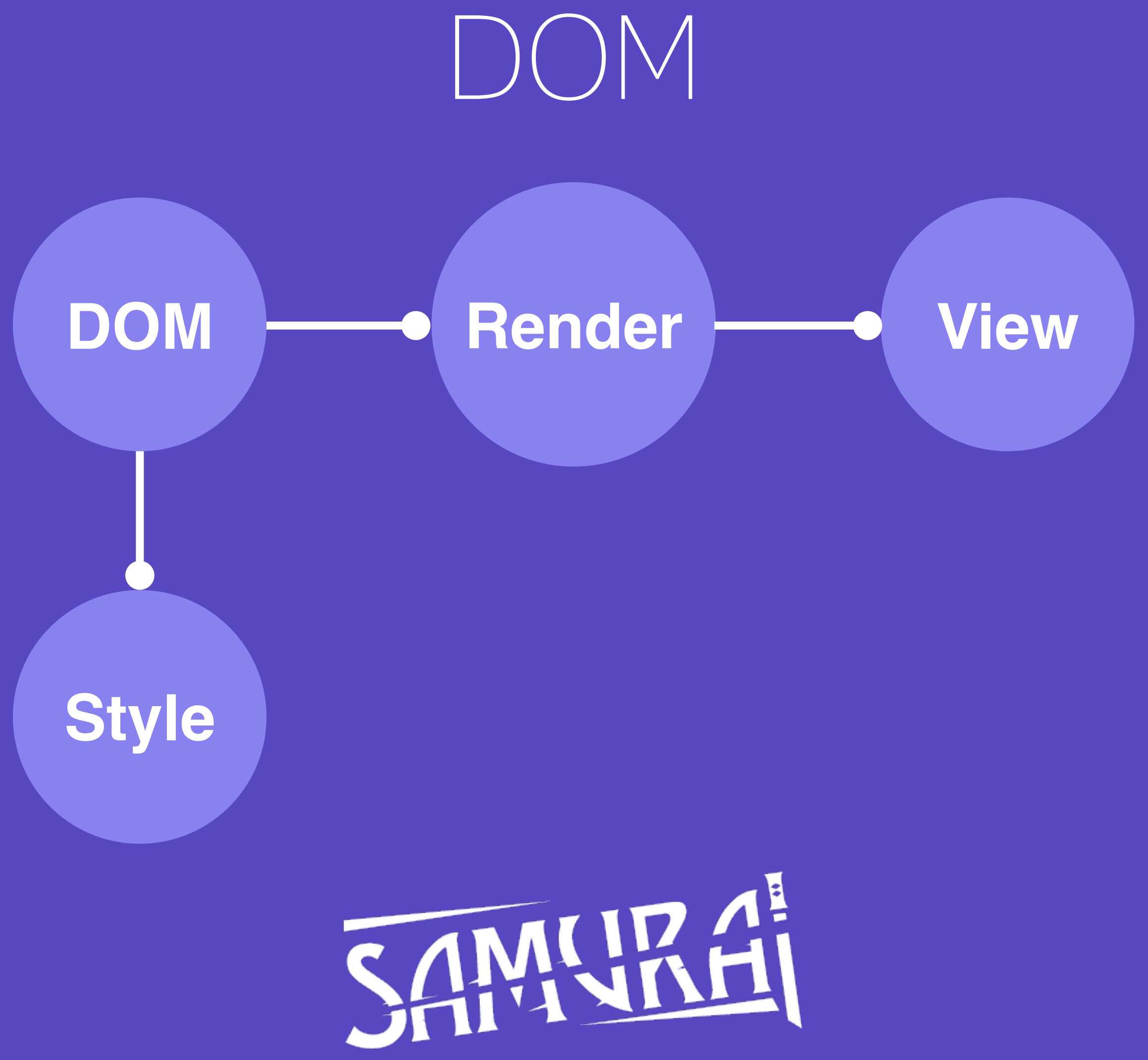
Style & layout

- Box-model
- Absolute/Relative positioning
- ShadowNode / css_node
- FlexBox layout

- Box-model
- Absolute/Relative positioning
- WebKit architecture
- Fluid layout

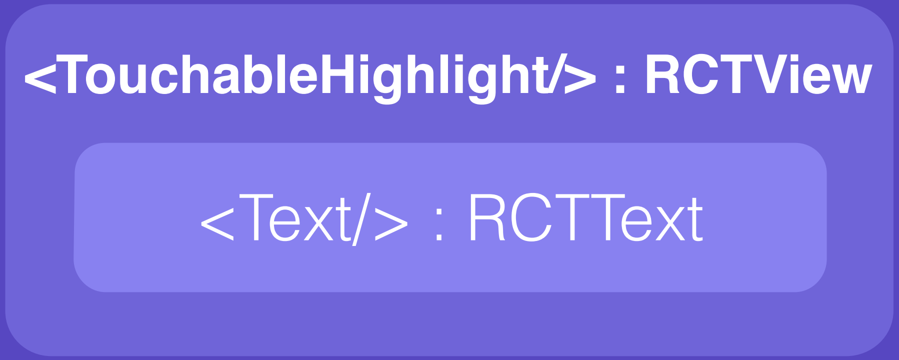
Style & layout





Touch handling

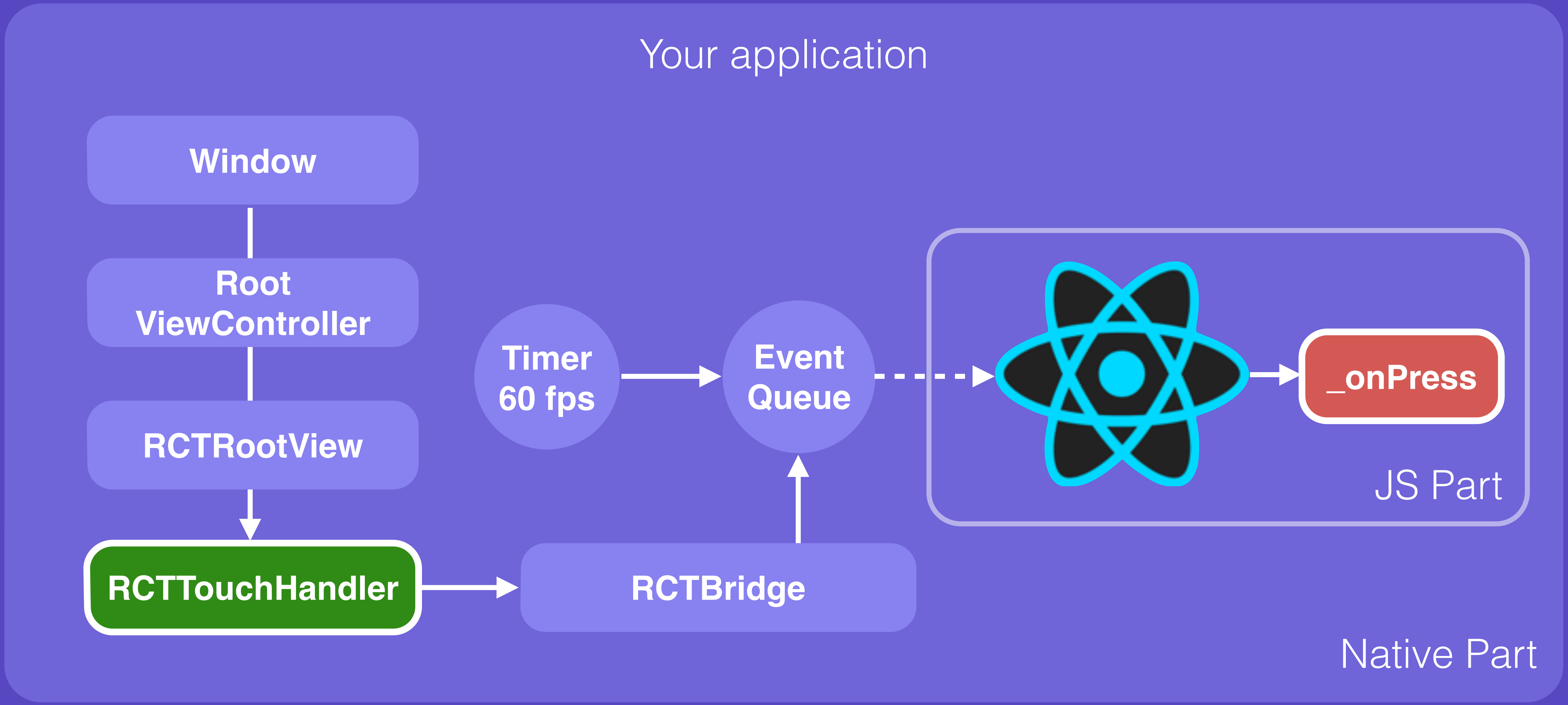
UIView mask



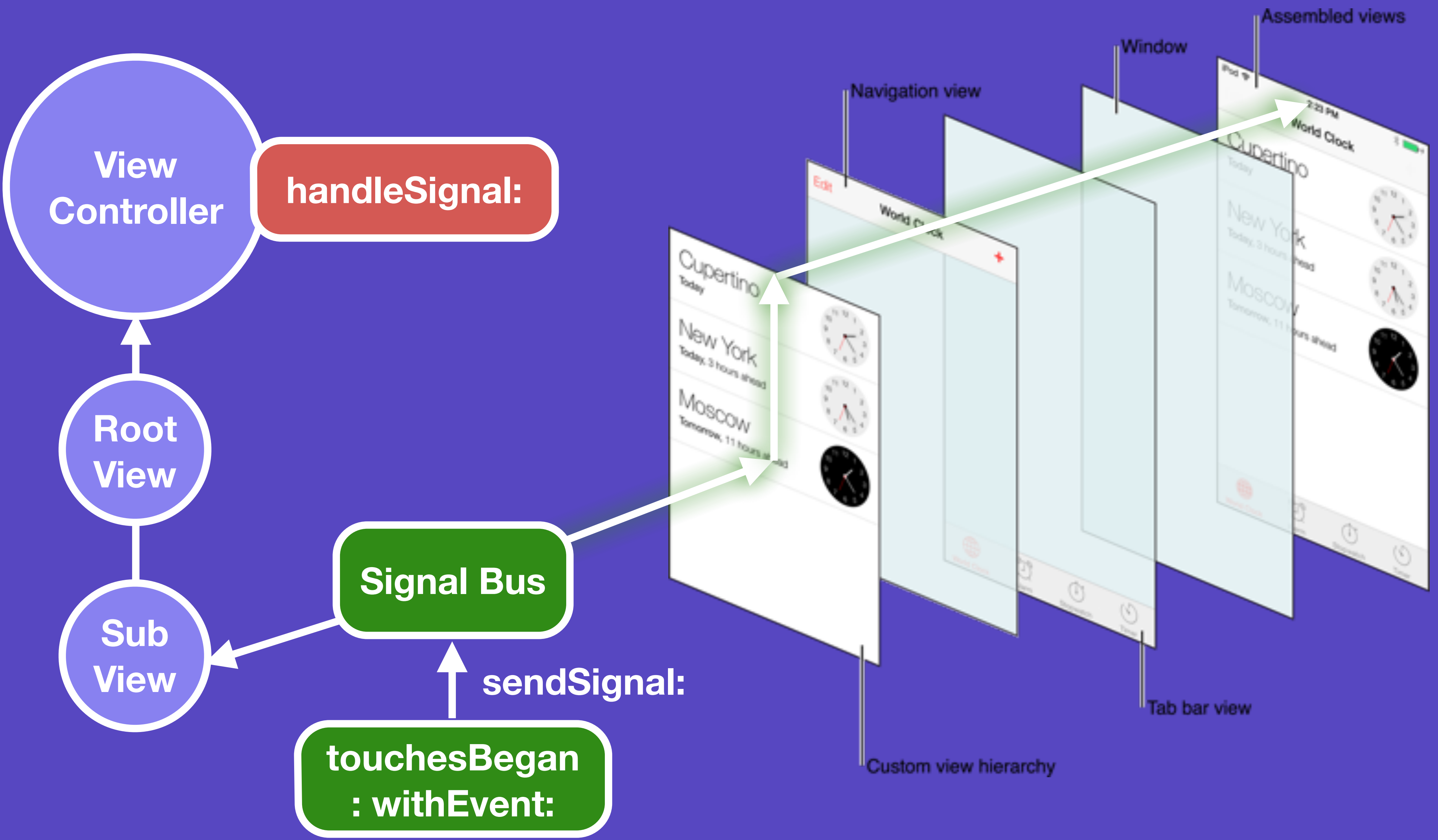
Gesture recognizer

UIView + iOS gesture recognizer

Touch handling - react-native



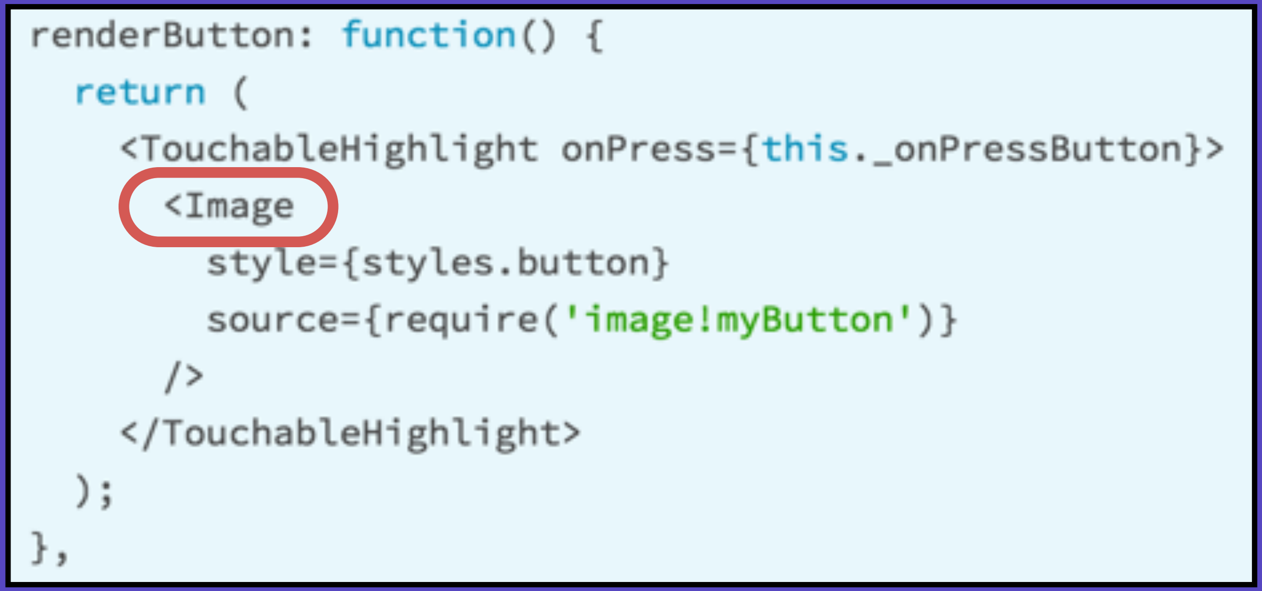
Touch handling - samurai-native



Responder chain

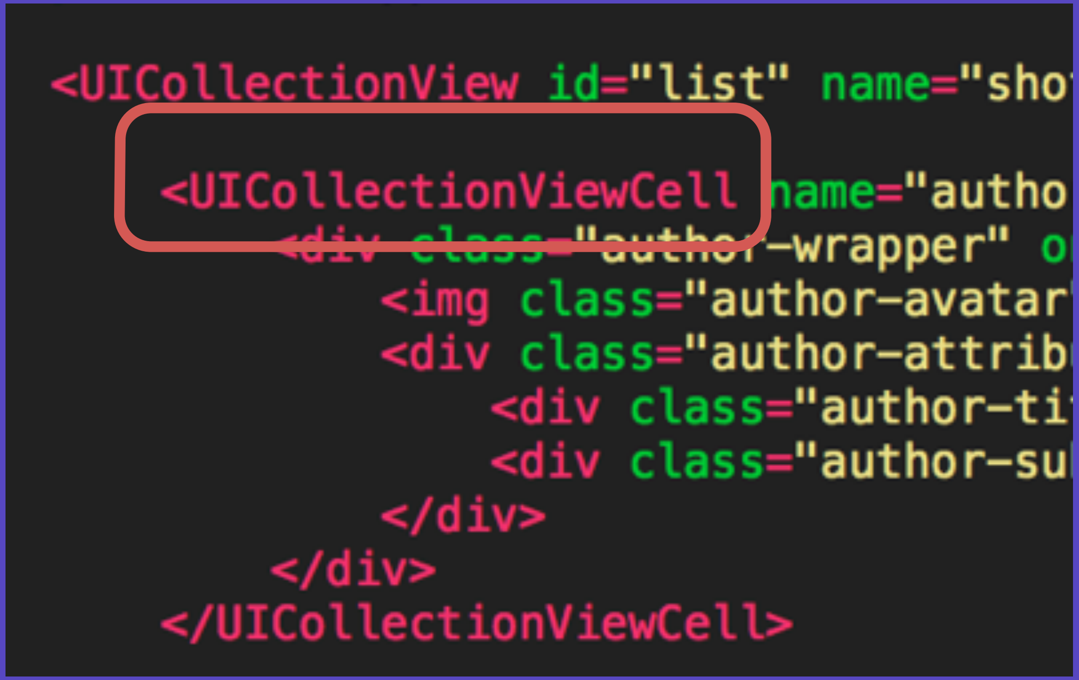
Native components

Custom components



```
renderButton: function() {
  return (
    <TouchableHighlight onPress={this._onPressButton}>
      <Image
        style={styles.button}
        source={require('image!myButton')}
      />
    </TouchableHighlight>
  );
},
```

Native components

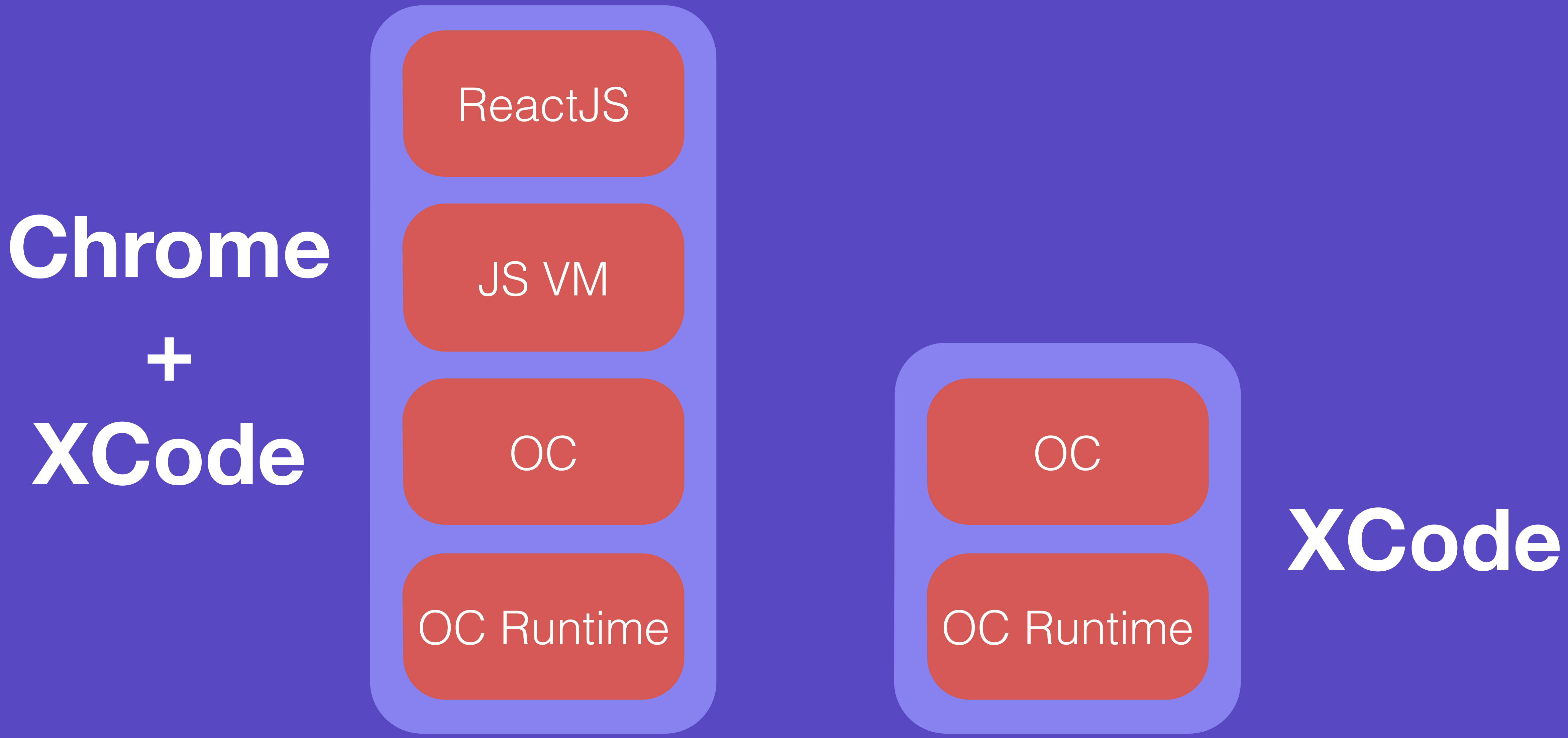


```
<UICollectionView id="list" name="sho

    <UICollectionViewCell name="autho
        <div class="author-wrapper" o
            <img class="author-avatar
            <div class="author-attrib
                <div class="author-ti
                <div class="author-su
            </div>
        </div>
    </UICollectionViewCell>
```

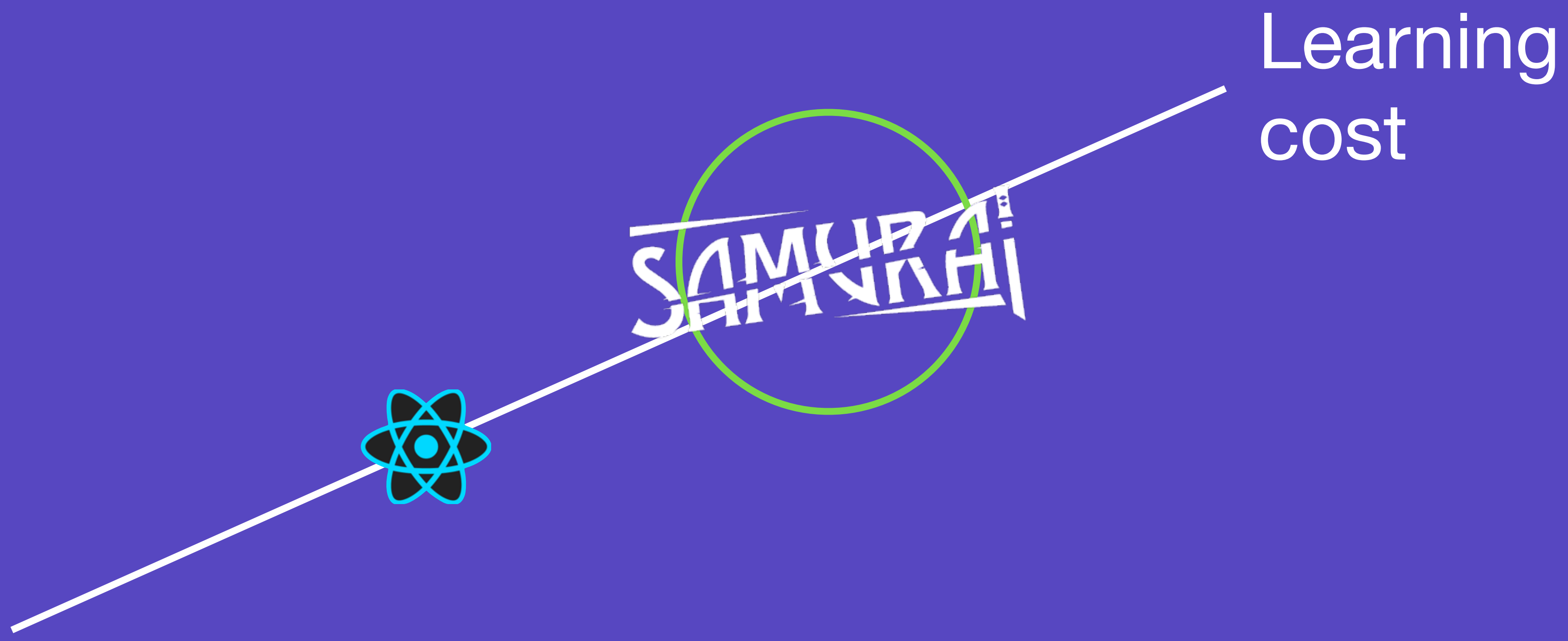
The difference

- Do you want <Text> or UILabel ?
- Do you want <List> or UITableView ?
- Do your want RCTView(drawRect) or AttributedString ?
- Do you want <TouchableHighlight> or onClick = ‘’ ?
- Do you want Fluid-layout or Flex-Box layout ?
- Do you want native gesture recognizer ?
- Do you want ResponderChain ?
- Do you want Xcode or Sublime ?
- Do you want iPhoneSimulator or Chrome?

The difference

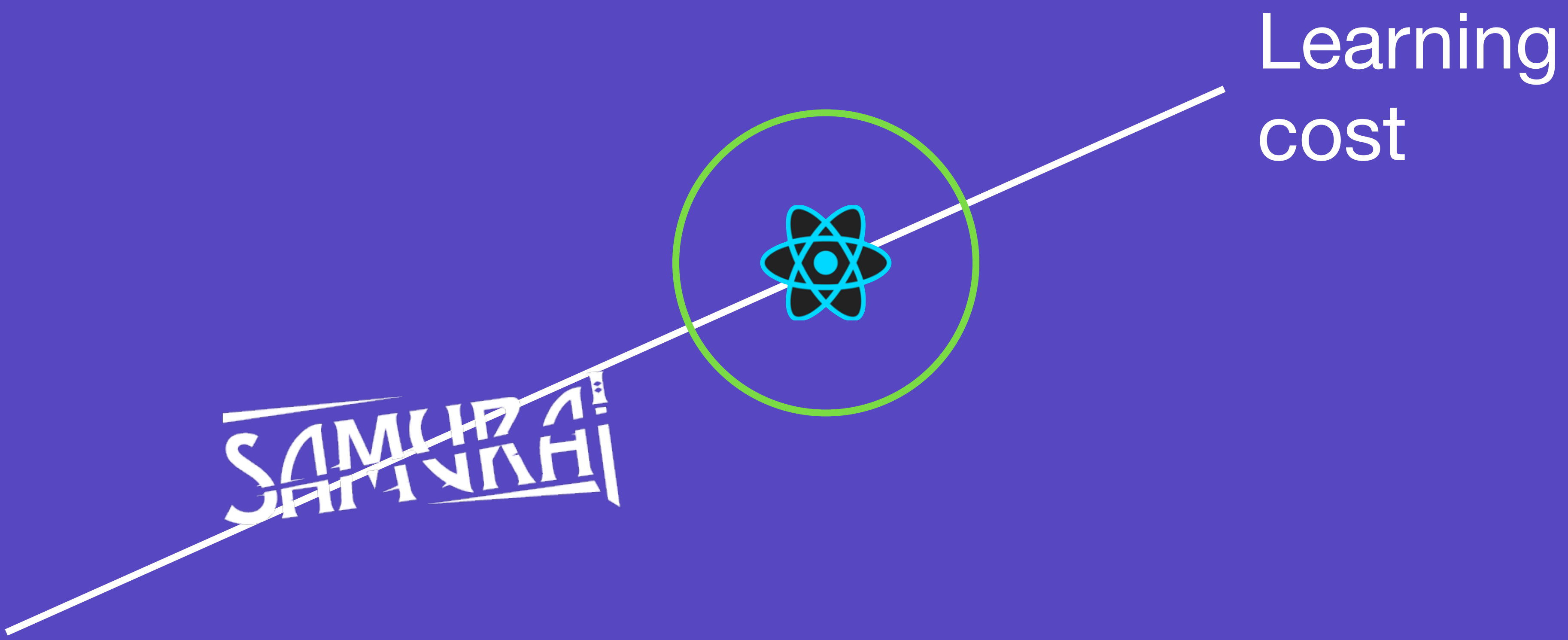


To front-end developer



HTML + CSS + JS/JSX + RectJS/RectNative + iOS/Android

To iOS developer



XML + CSS + Objective-C + BeeFramework + iOS/Android

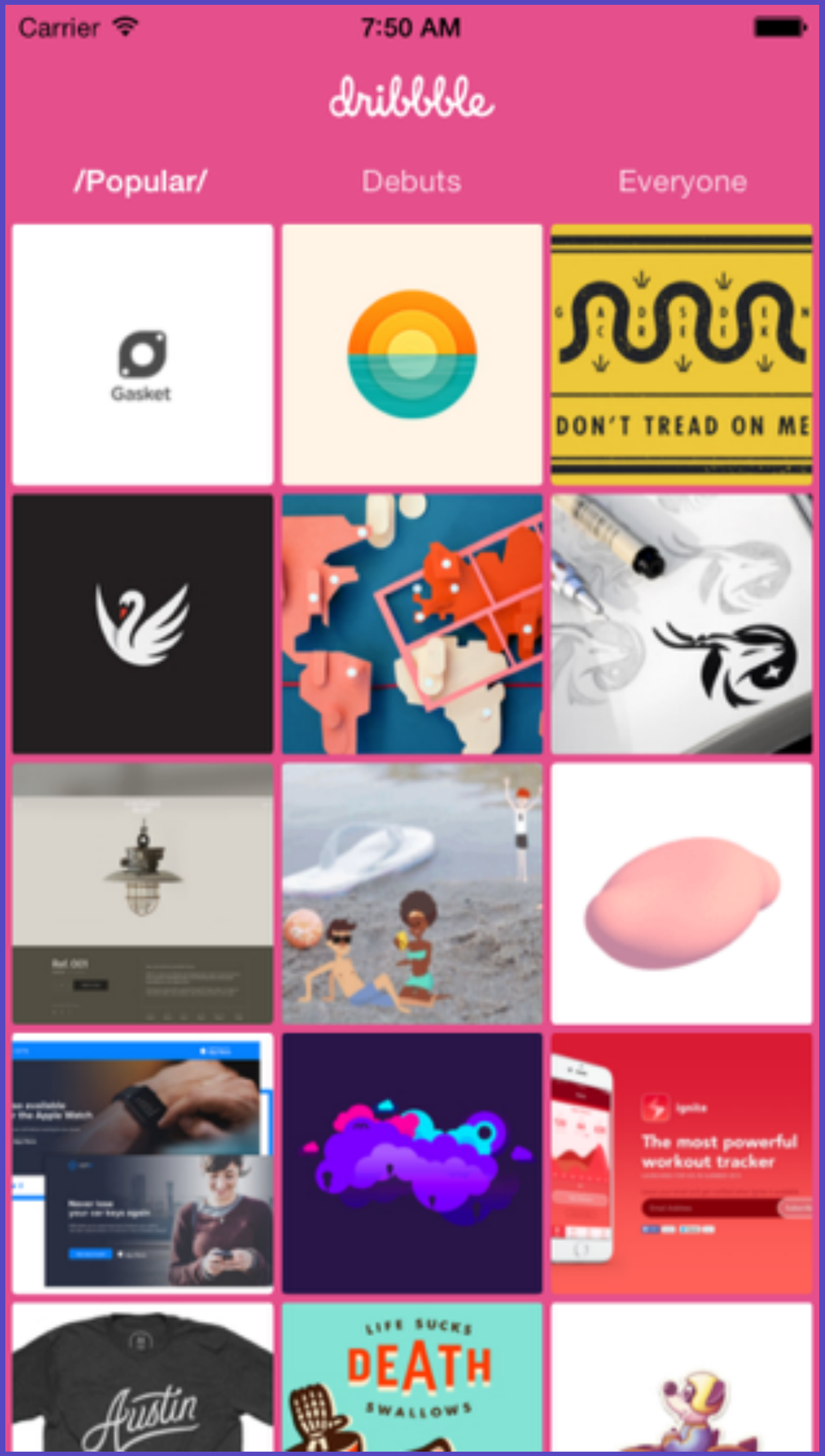
More native? or more web? Your team decide.

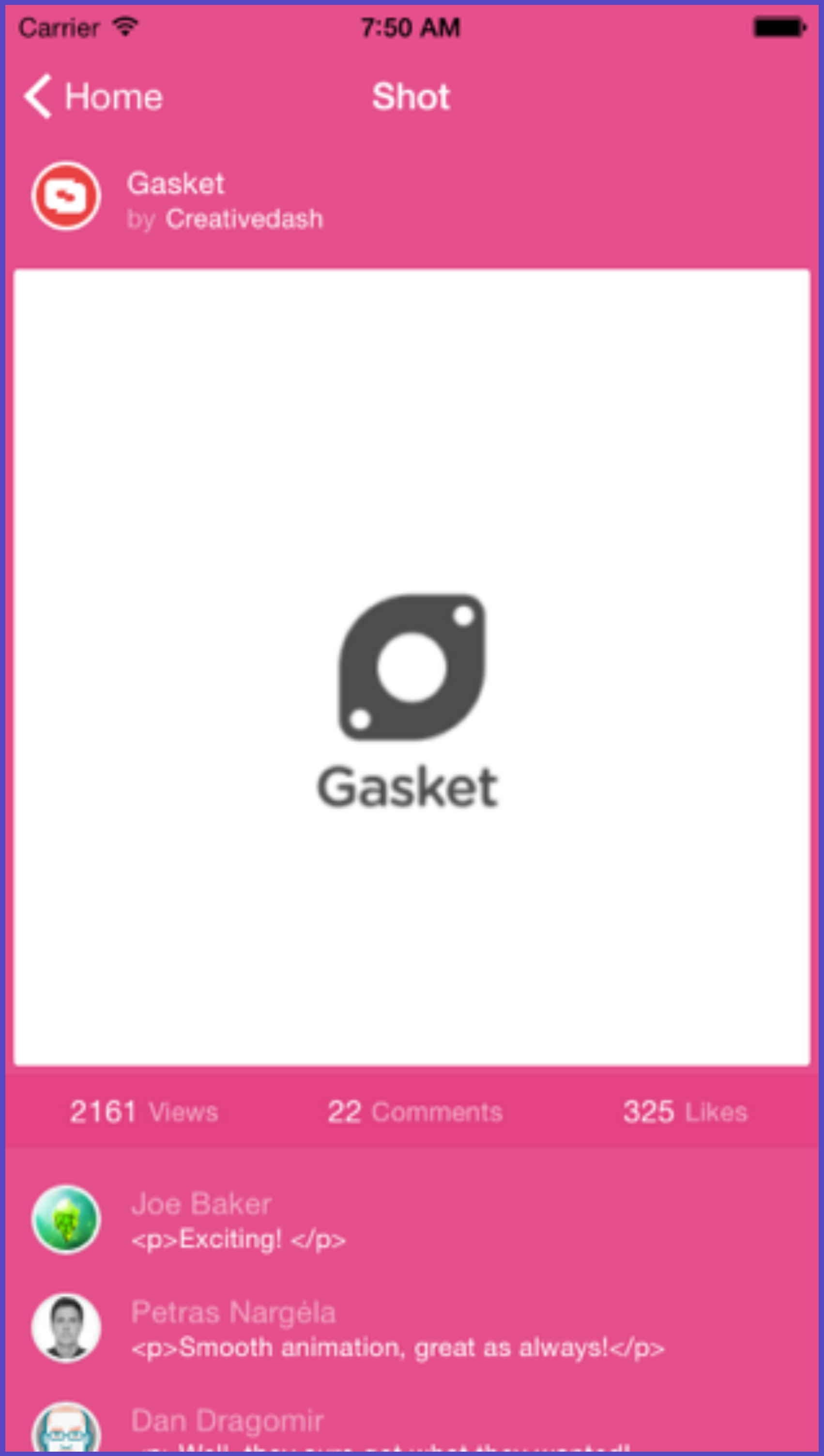
SAMURAI

Yet another Semi-Hybrid framework.

https://github.com/hackers-painters/samurai-native

Demo

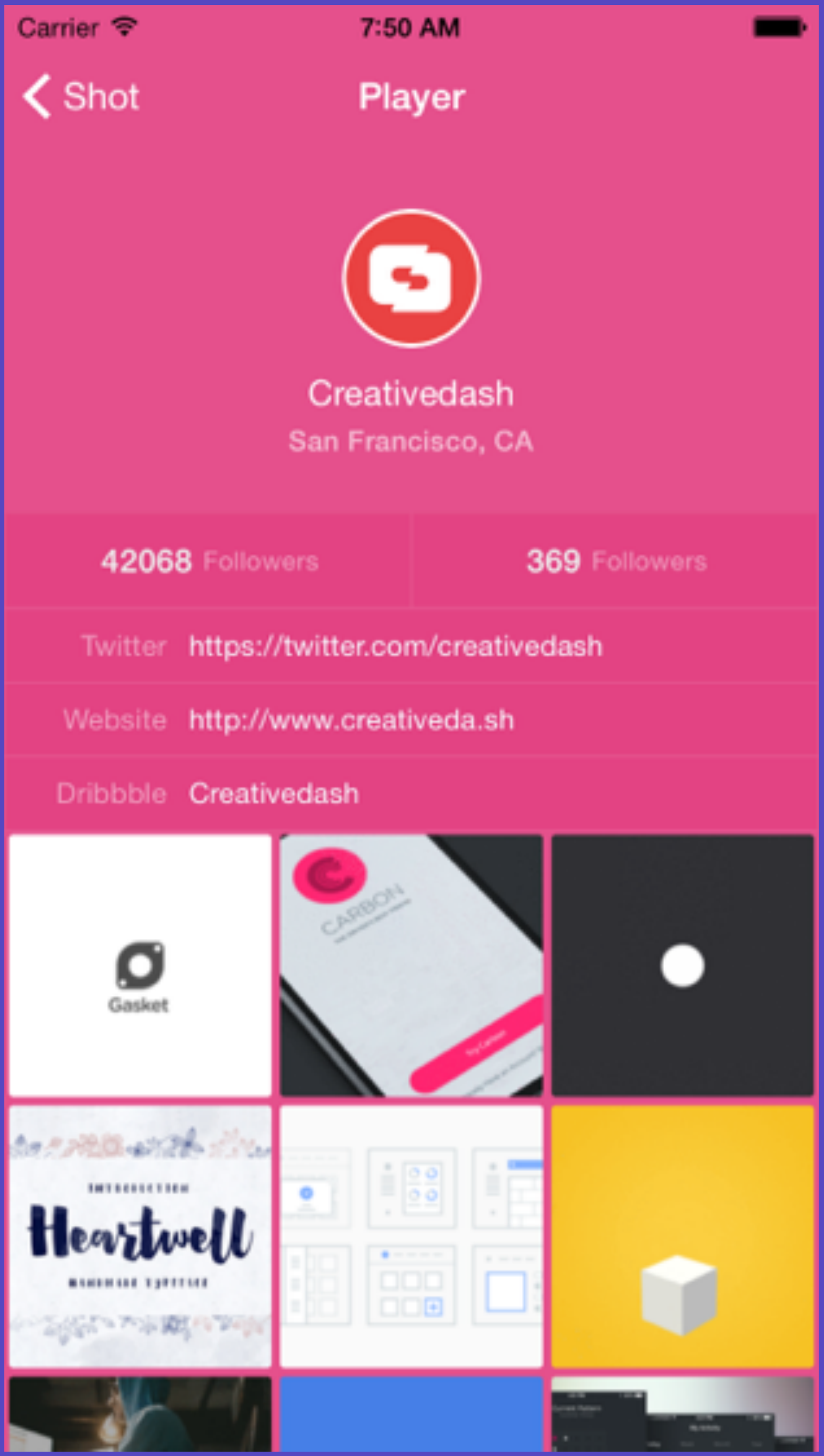




Joe Baker
<p>Exciting! </p>

Petras Nargėla
<p>Smooth animation, great as always!</p>

Dan Dragomir



Creativedash
San Francisco, CA

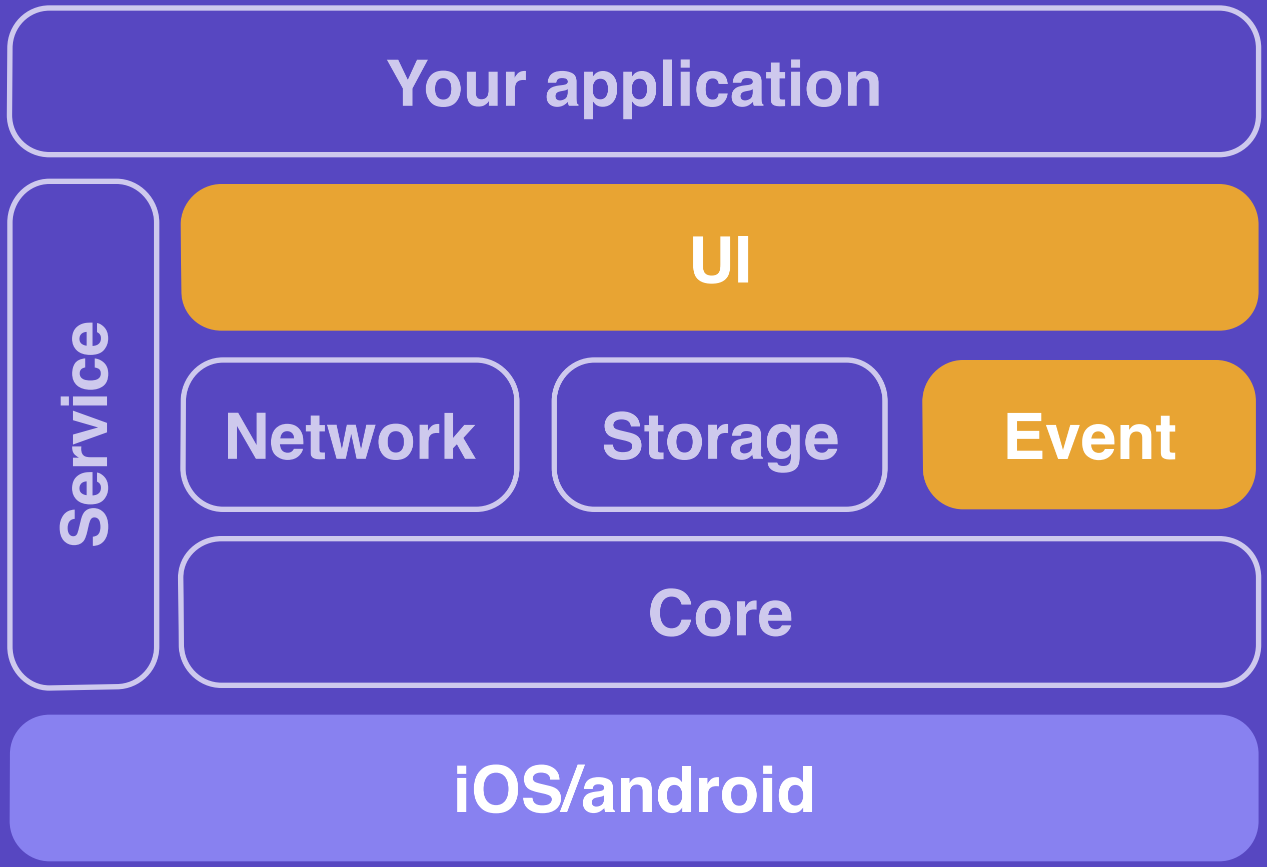
42068 Followers | 369 Followers

Twitter	https://twitter.com/creativedash
Website	http://www.creativeda.sh
Dribbble	Creativedash

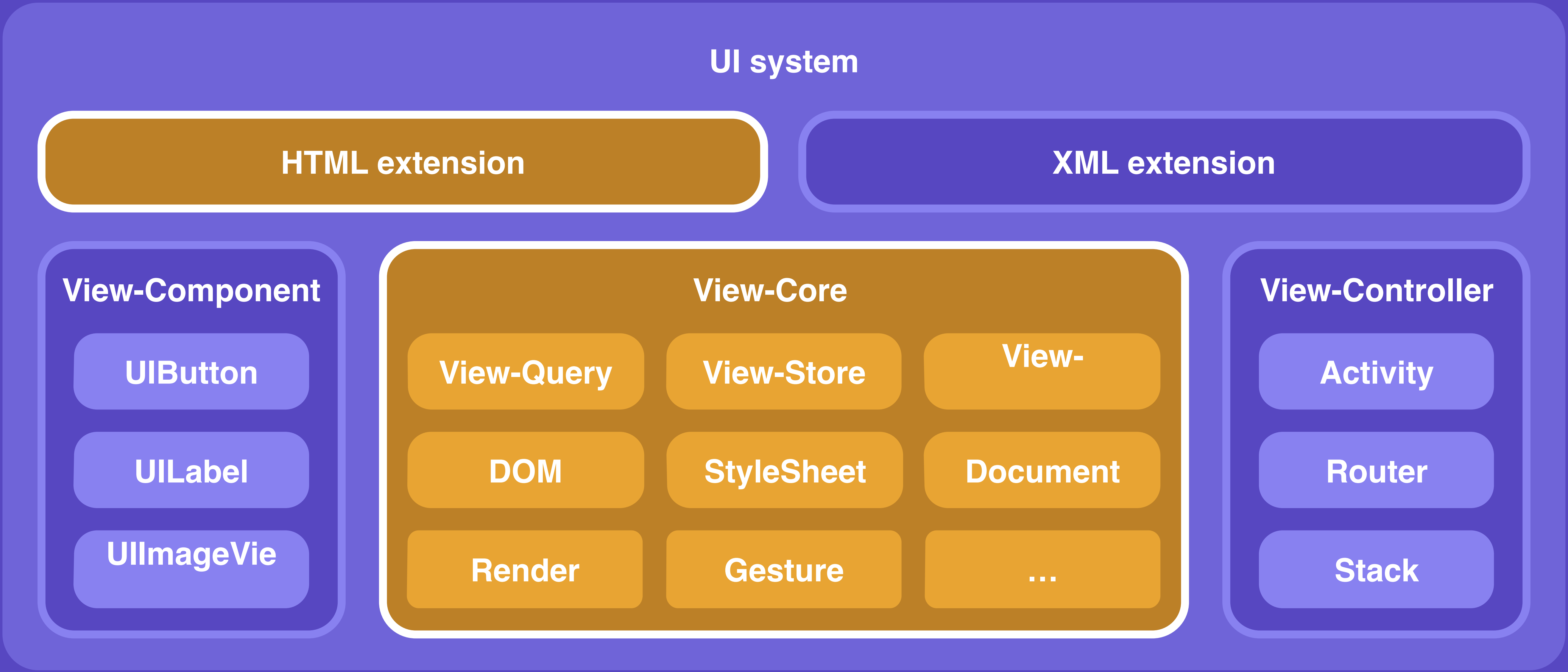
The advantages

- Simple, more native, using XCode/iPhoneSimulator.
- Browser architecture.
- Full decoupling UI, data and business logic.
- Good alternative to WebView.
- Good native implementation.
- Low learning cost to native developers.
- Low learning cost to front-end developers.

Architecture



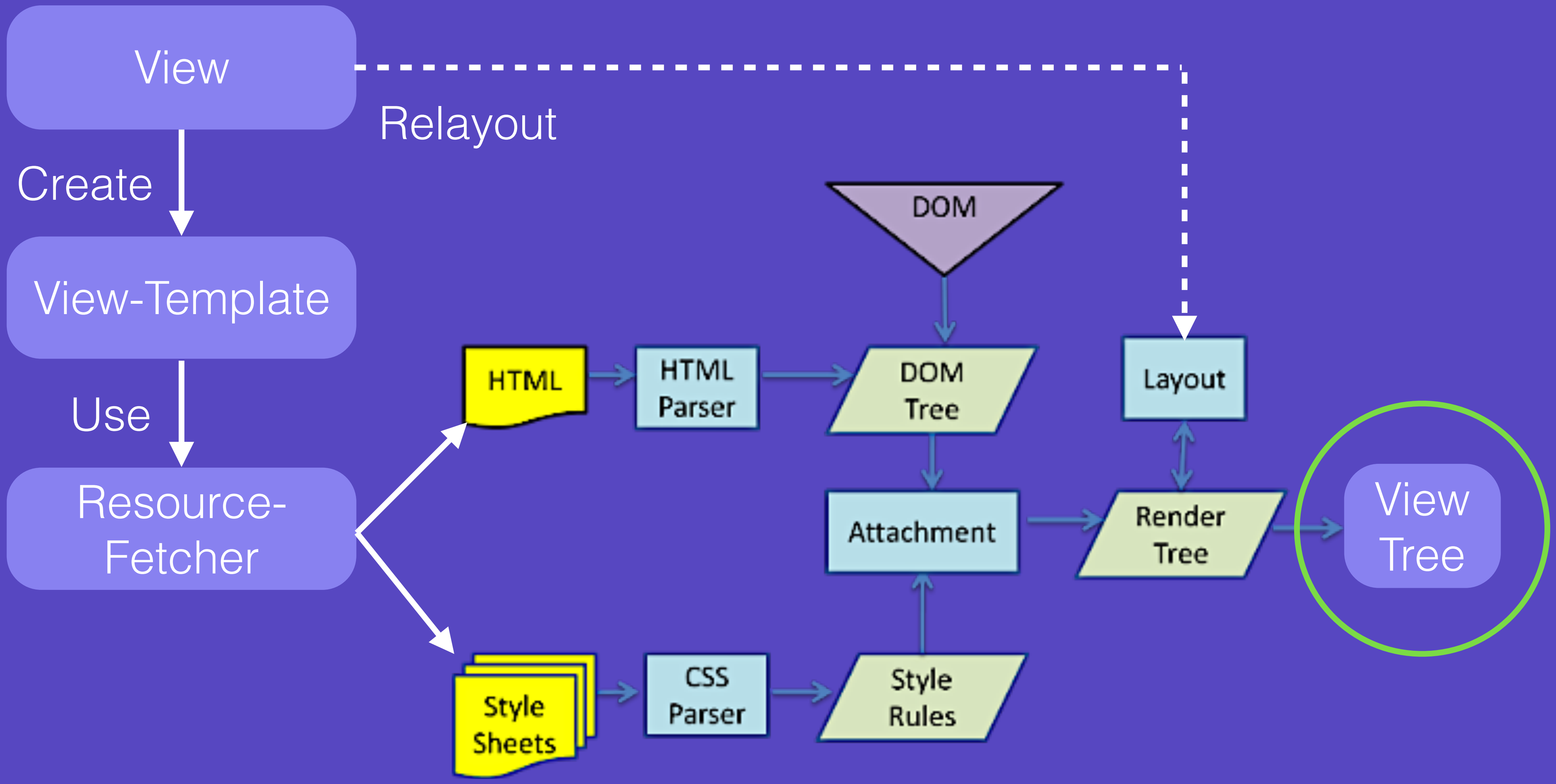
Architecture



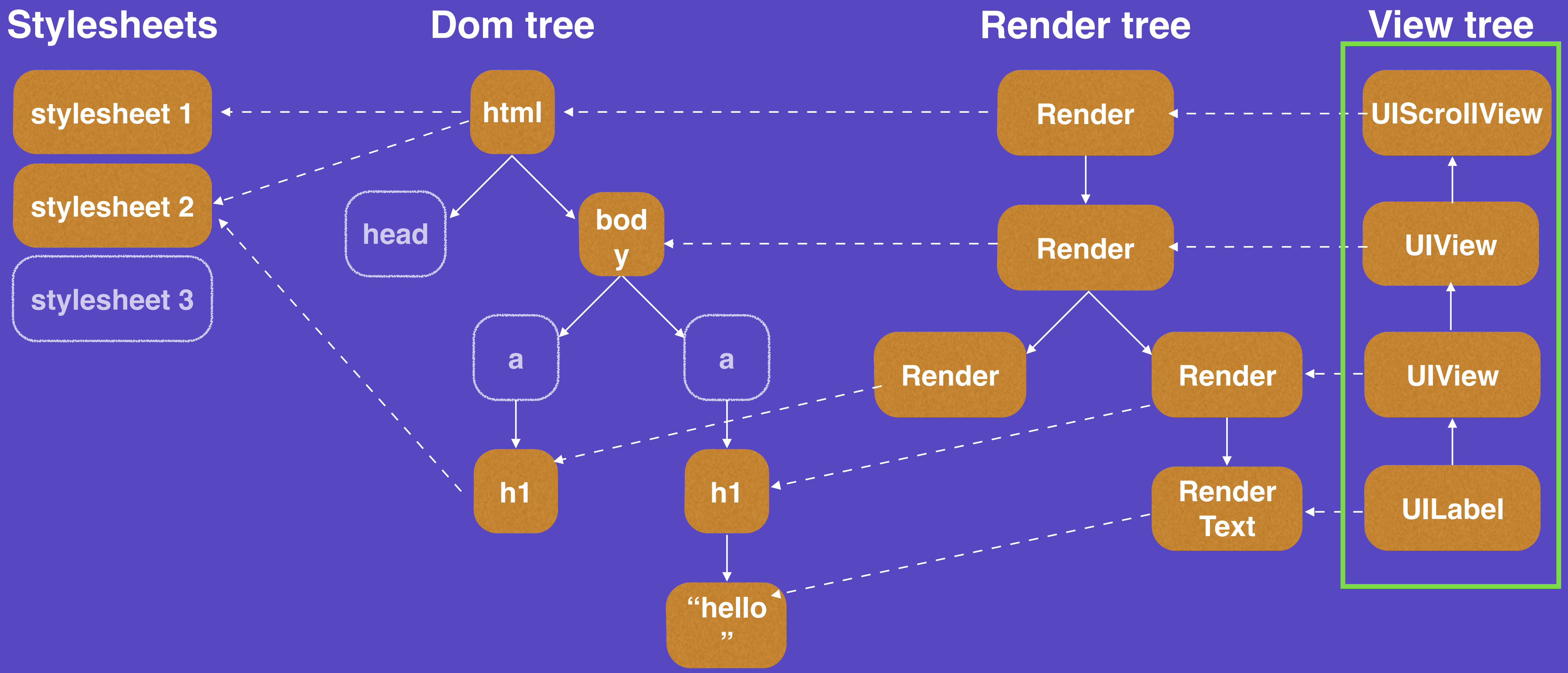
Key modules

- Gumbo parser, from Google, pure C99
- Katana parser, from GeekZoo, pure C99
- samurai-native WebCore from GeekZoo, Objective-C

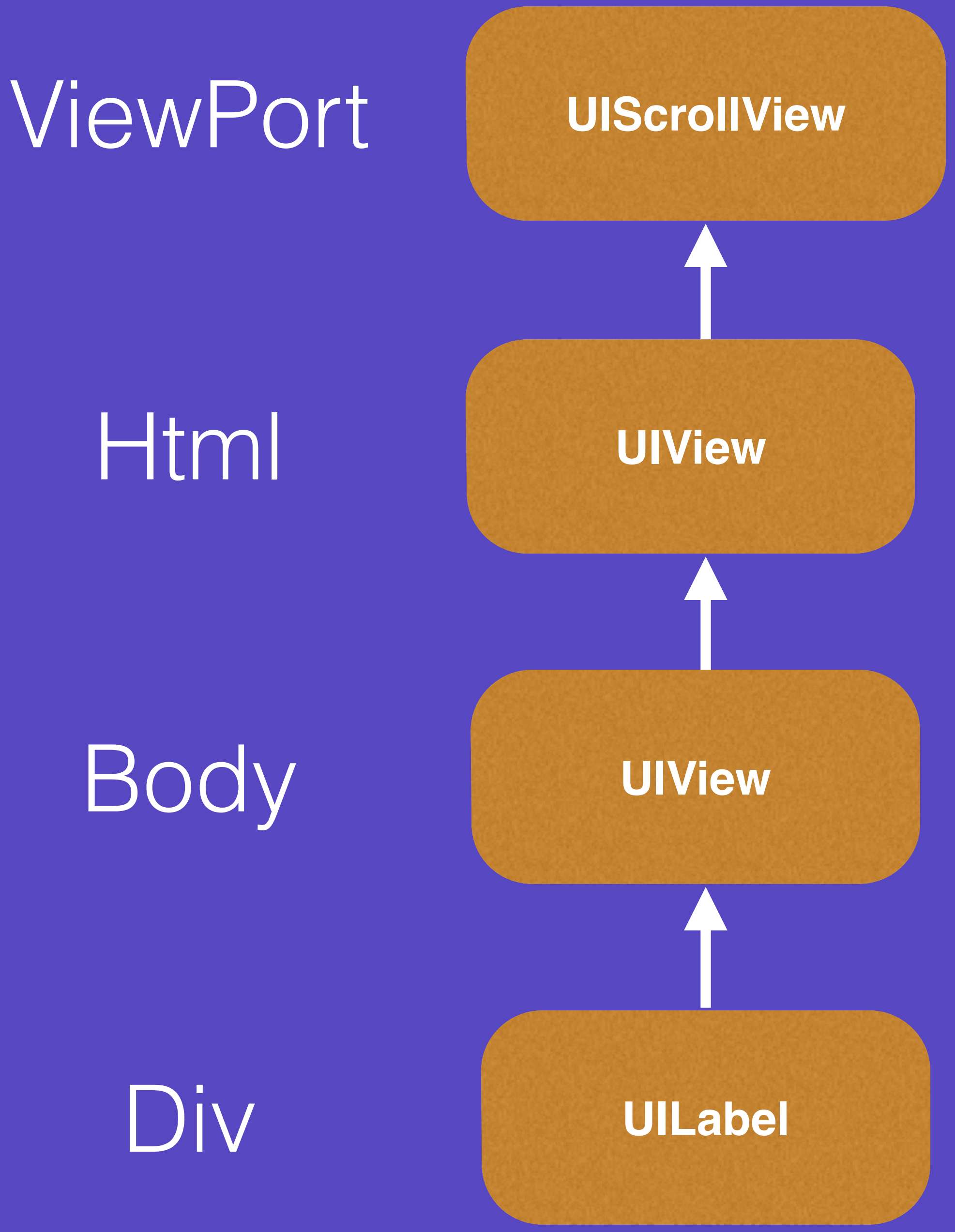
Workflow


View
Relayout
Create
View-Template
Use
Resource-Fetcher
DOM
HTML
HTML Parser
DOM Tree
Layout
Attachment
Render Tree
View Tree
Style Sheets
CSS Parser
Style Rules

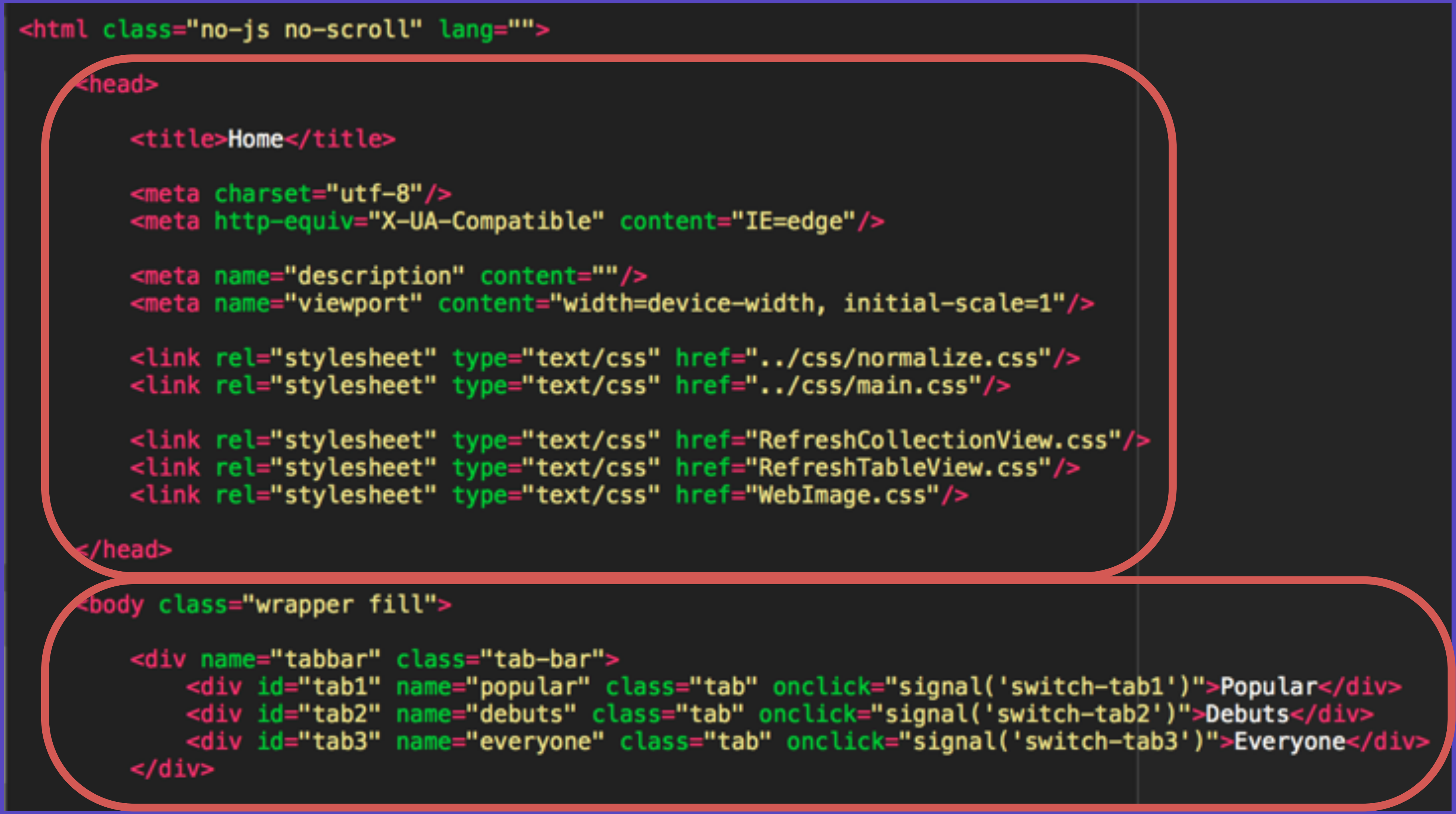
Memory model



View model

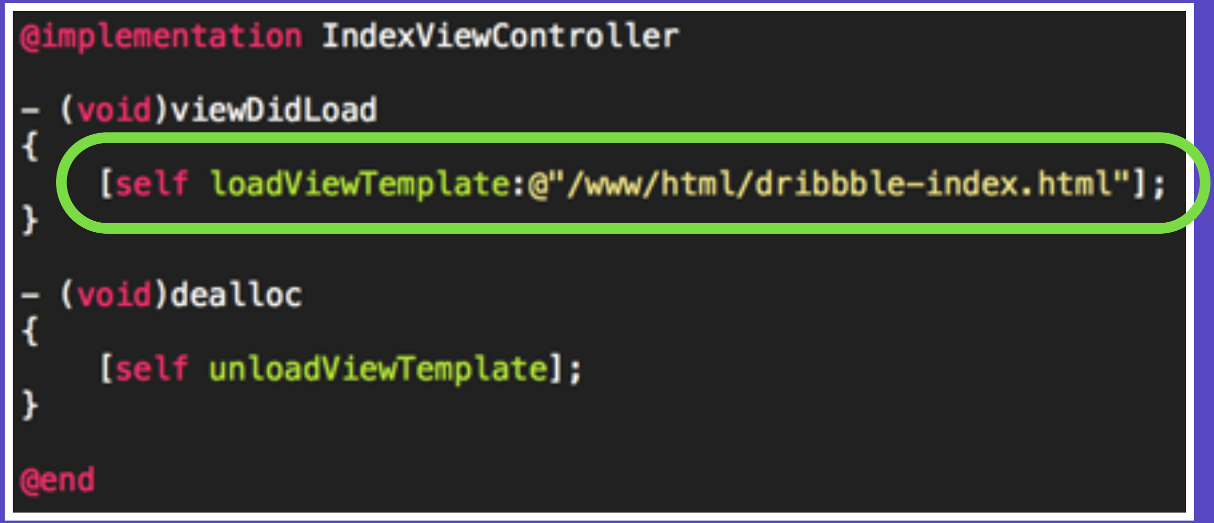

ViewPort
UIScrollView
Html
UIView
Body
UIView
Div
UILabel

Standard HTML

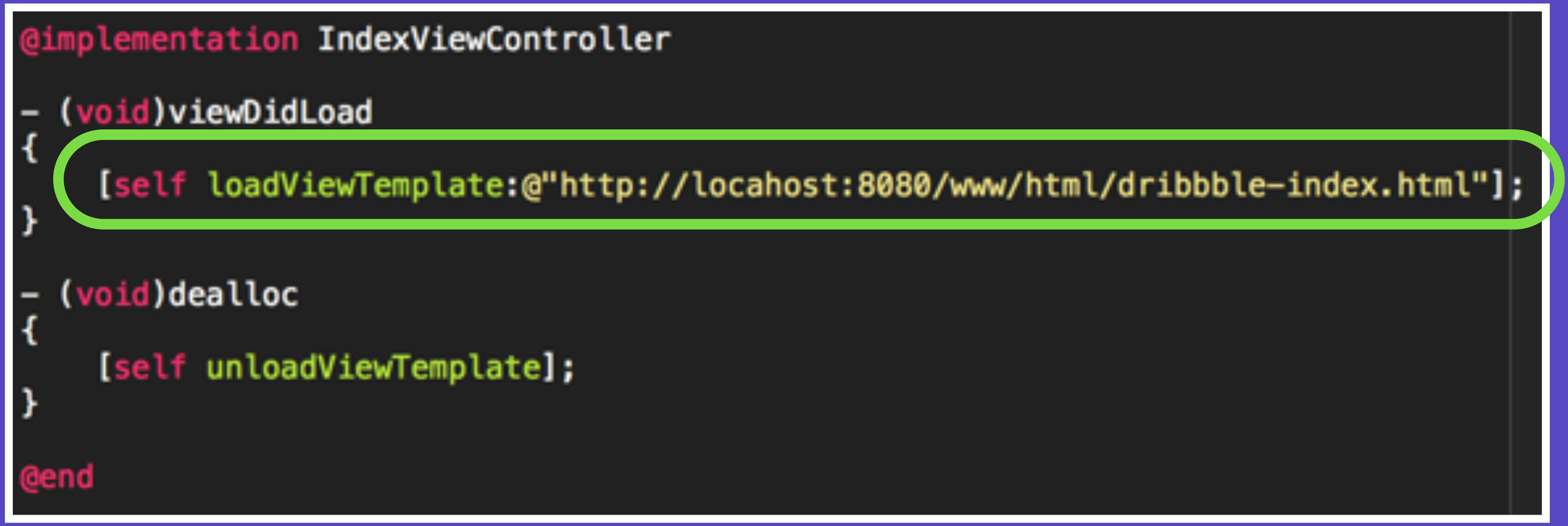


```html
<html class="no-js no-scroll" lang="">

    <head>

        <title>Home</title>

        <meta charset="utf-8"/>
        <meta http-equiv="X-UA-Compatible" content="IE=edge"/>

        <meta name="description" content=""/>
        <meta name="viewport" content="width=device-width, initial-scale=1"/>

        <link rel="stylesheet" type="text/css" href="../css/normalize.css"/>
        <link rel="stylesheet" type="text/css" href="../css/main.css"/>

        <link rel="stylesheet" type="text/css" href="RefreshCollectionView.css"/>
        <link rel="stylesheet" type="text/css" href="RefreshTableView.css"/>
        <link rel="stylesheet" type="text/css" href="WebImage.css"/>

    </head>
    <body class="wrapper fill">

        <div name="tabbar" class="tab-bar">
            <div id="tab1" name="popular" class="tab" onclick="signal('switch-tab1')">Popular</div>
            <div id="tab2" name="debuts" class="tab" onclick="signal('switch-tab2')">Debuts</div>
            <div id="tab3" name="everyone" class="tab" onclick="signal('switch-tab3')">Everyone</div>
        </div>
```

Easy API



```
@implementation IndexViewController

- (void)viewDidLoad
{
    [self loadViewTemplate:@"/www/html/dribbble-index.html"];
}

- (void)dealloc
{
    [self unloadViewTemplate];
}

@end
```



```
@implementation IndexViewController

- (void)viewDidLoad
{
    [self loadViewTemplate:@"http://locahost:8080/www/html/dribbble-index.html"];
}

- (void)dealloc
{
    [self unloadViewTemplate];
}

@end
```

HTML (1)

- Support standard HTML tag (see html.css)
  - <p> -> UILabel
  - <div> -> UIView
  - <img> -> UIImageView
  - <span> -> UILabel
  - and more …

HTML (2)

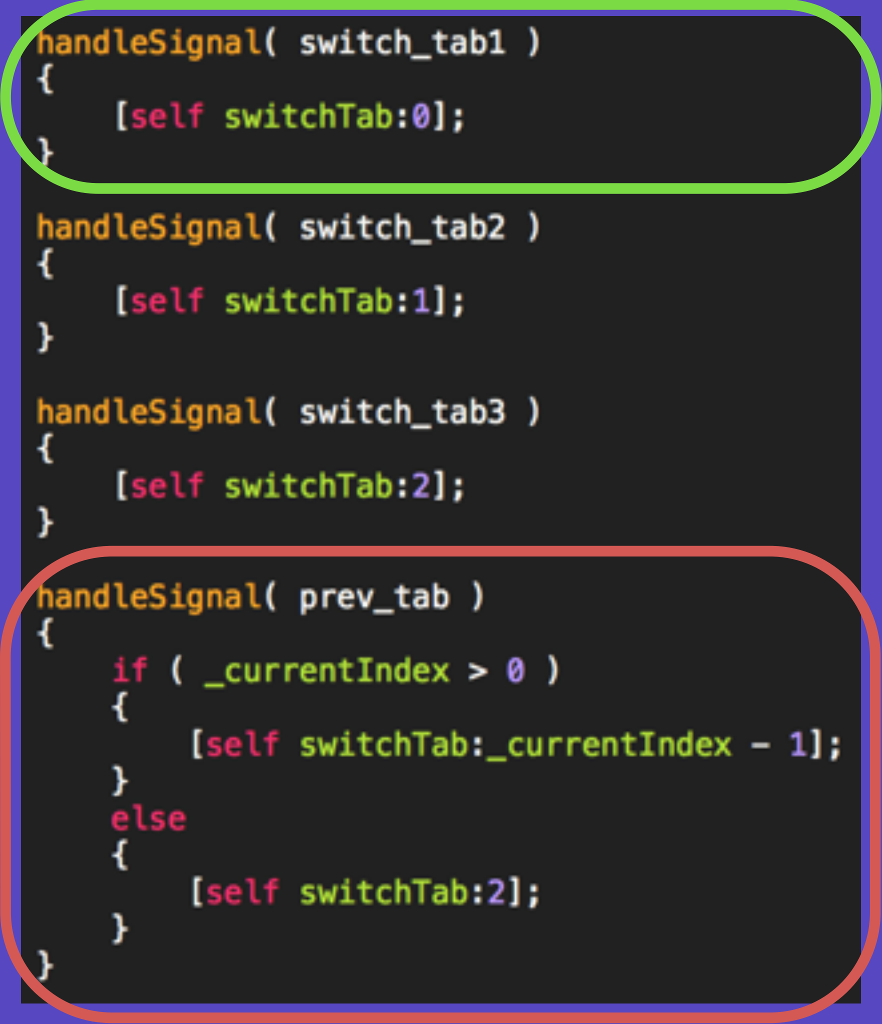
- Support native components
    - <UILabel> -> UILabel
    - <UIImageView> -> UIImageView
- Support container / reusable components
    - <UICollectionView> -> UICollectionView
    - <UICollectionViewCell> -> UICollectionViewCell
    - and more …

HTML (3)

- Support link style
    - <link rel="stylesheet" type="text/css" href="../css/normalize.css"/>
    - <link rel="stylesheet" type="text/css" href=“../css/main.css”/>
- Support <style media=“all”></style>
- Support inline style
    - <p style=“color:red”/>

HTML (4)

- Support gesture events
  - <div onclick="signal('switch-tab1')"/>
  - <div onswipe="signal('test')"/>
    - <div onswipe-left="signal('prev-tab')"/>
    - <div onswipe-right="signal('test')"/>
    - <div onswipe-up="signal('test')"/>
    - <div onswipe-down="signal('test')"/>
  - <div onpinch="signal('test')"/>
  - <div onpan="signal('test')"/>



```
handleSignal( switch_tab1 )
{
    [self switchTab:0];
}

handleSignal( switch_tab2 )
{
    [self switchTab:1];
}

handleSignal( switch_tab3 )
{
    [self switchTab:2];
}

handleSignal( prev_tab )
{
    if ( _currentIndex > 0 )
    {
        [self switchTab:_currentIndex - 1];
    }
    else
    {
        [self switchTab:2];
    }
}
```

HTML (5)

- Support inline text
    - <p>Hello, <span>Samurai</span></p>
- Support inline DOM
    - <b><p><i><span>Hello</span></i></p></b>
- Support quirks mode
    - Hello<br>, Samurai

CSS (1)

- Support CSS 2.0 / 3.0 syntax
  - tag { color: red; }
  - #id { color: red; }
  - .class { color: red; }
- Support CSS 2.0 / 3.0 selector
  - <UIScrollView id="list"/></UIScrollView>
  - <UIScrollView class=“style1 style2"/></UIScrollView>

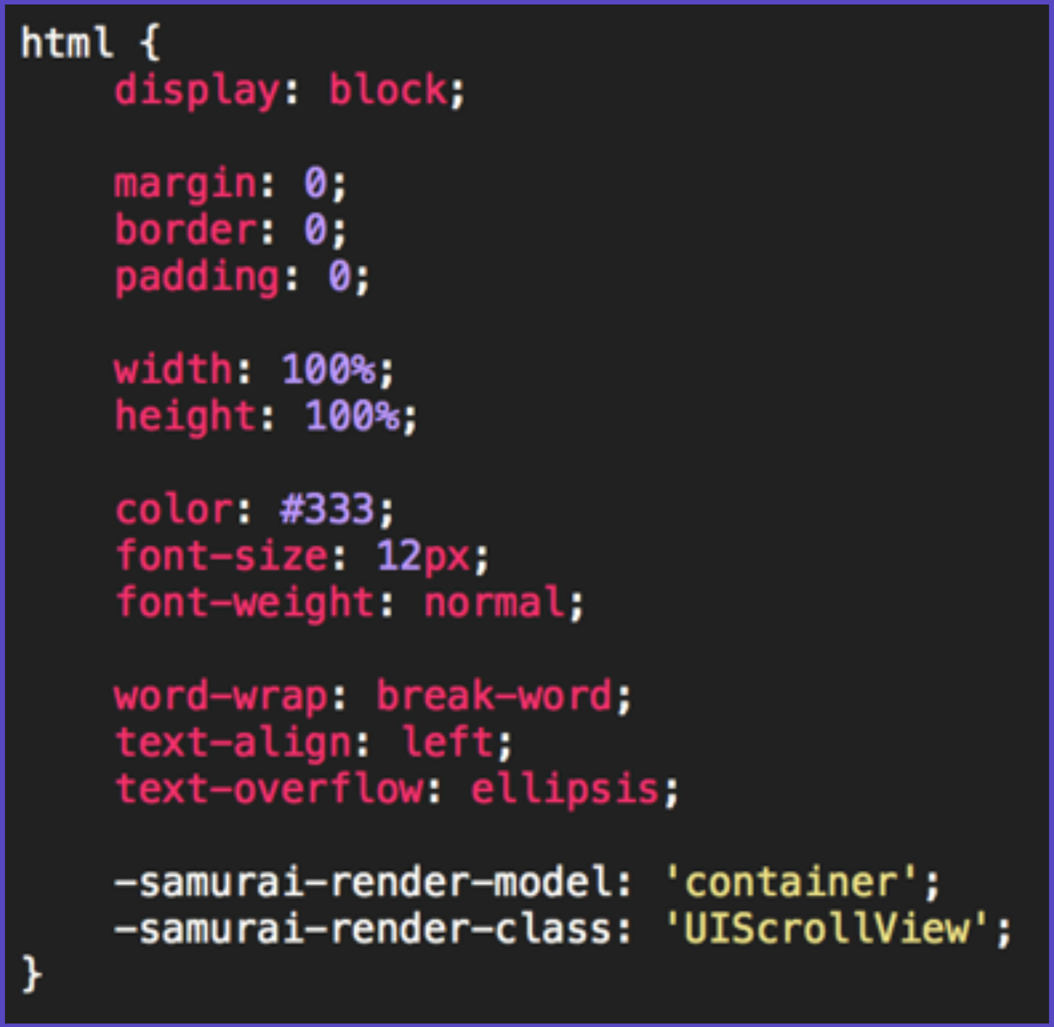
CSS (2)

- Support custom function
    - { width: equals(height); }
    - { height: equals(width); }
- Support media query
    - @media (device-width: 320px) { }
- Katana powered

```
device-width:
320px
min-device-width:
320px
max-device-width:
320px
device-height:
320px
min-device-height:
320px
max-device-height:
320px
scale:                 1.0
min-scale:             1.0
max-scale:             2.0
orientation:
landscape
```

CSS (3)

- User agent stylesheet **html.css**
    - -samurai-render-model:
        - element
        - container
        - hidden
        - inline
    - -samurai-render-class:



```
html {
    display: block;

    margin: 0;
    border: 0;
    padding: 0;

    width: 100%;
    height: 100%;

    color: #333;
    font-size: 12px;
    font-weight: normal;

    word-wrap: break-word;
    text-align: left;
    text-overflow: ellipsis;

    -samurai-render-model: 'container';
    -samurai-render-class: 'UIScrollView';
}
```

Native component (1)

- Only 3 steps
    - MyView.h
    - MyView.m
    - <MyView/>

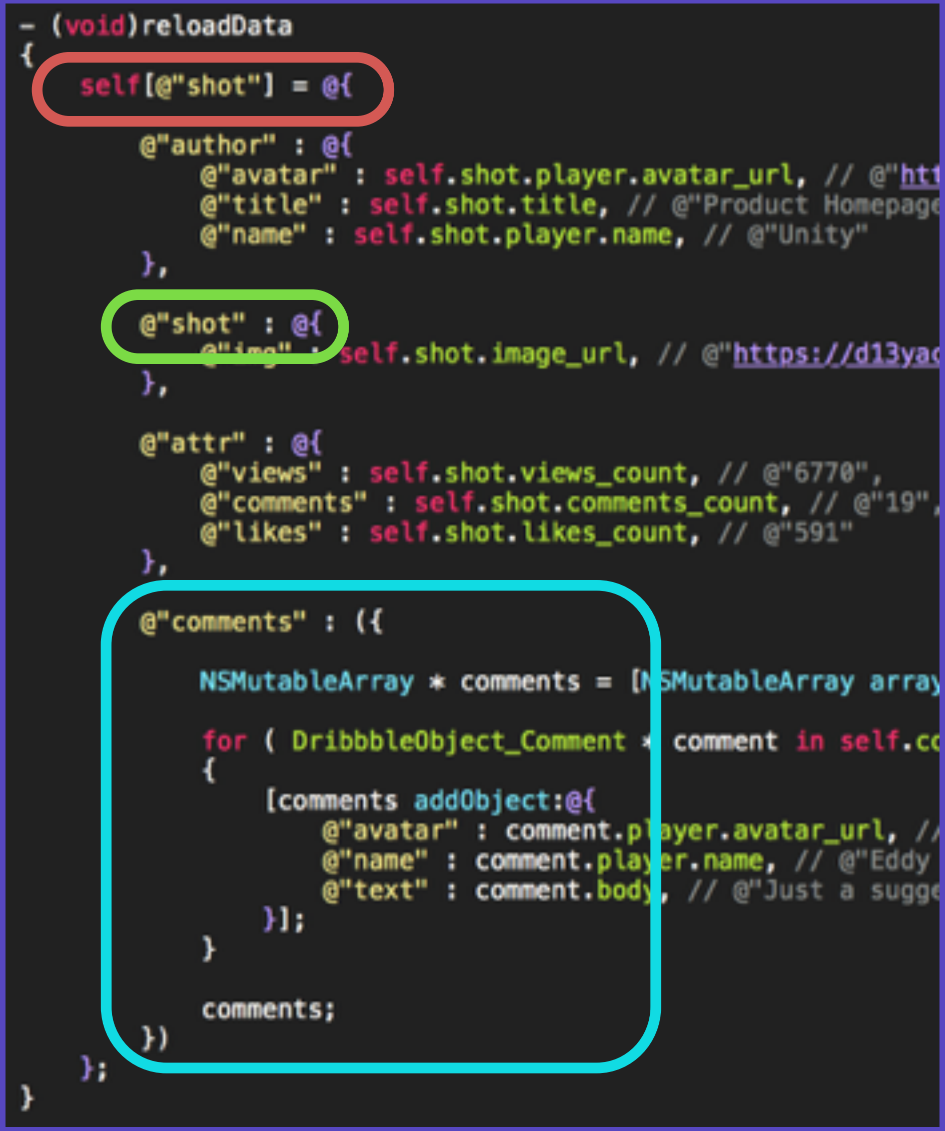
Native component (2)

- Implement native view (MyView.m)
  - @implementation MyView
    - - (void) html_applyDom :(SamuraiHtmlDomNode *)dom {}
    - - (void) html_applyStyle :(SamuraiHtmlRenderStyle *)style {}
  - @end

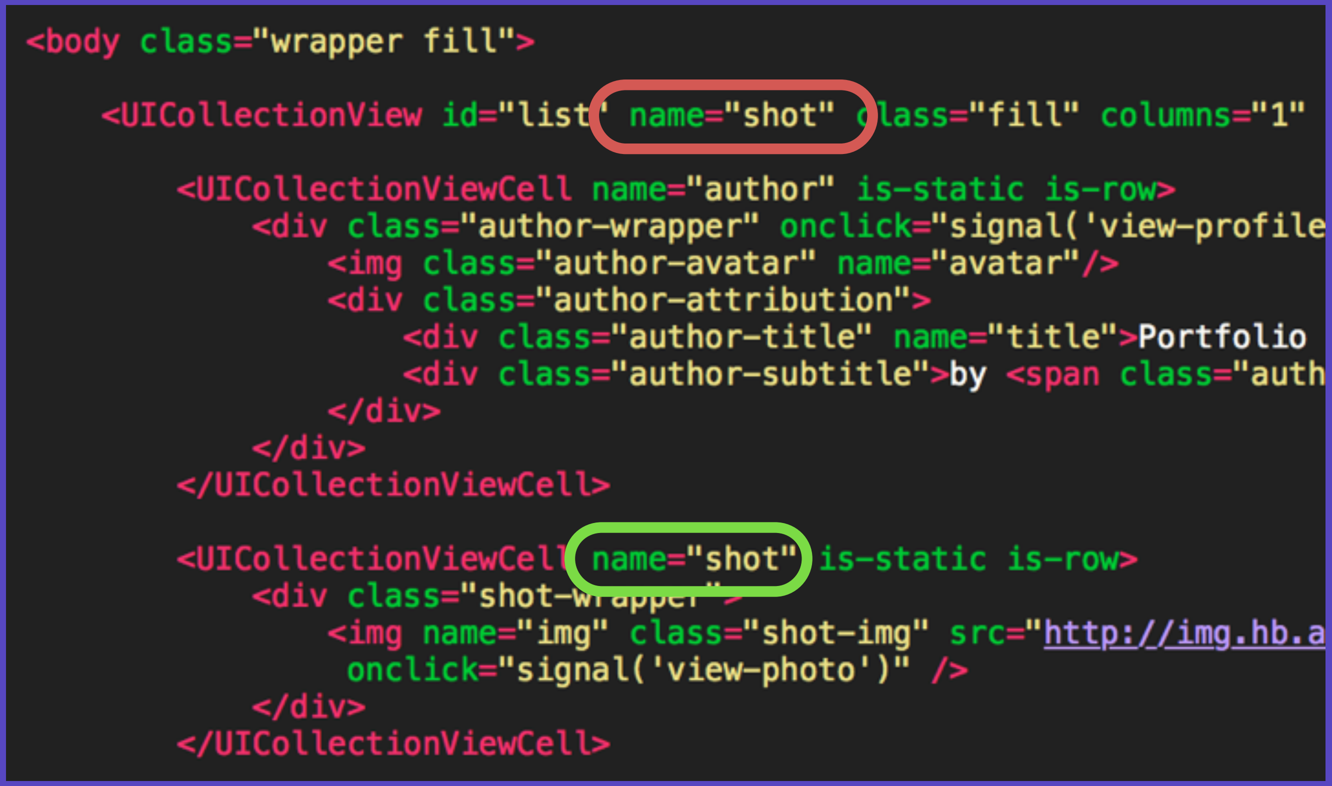
Native component (3)

- Import native component in web page (XXX.html)
  - <body>
    - <MyView id=“id” class=“style1 style2”/>
  - </body>

Data binding/query (1)



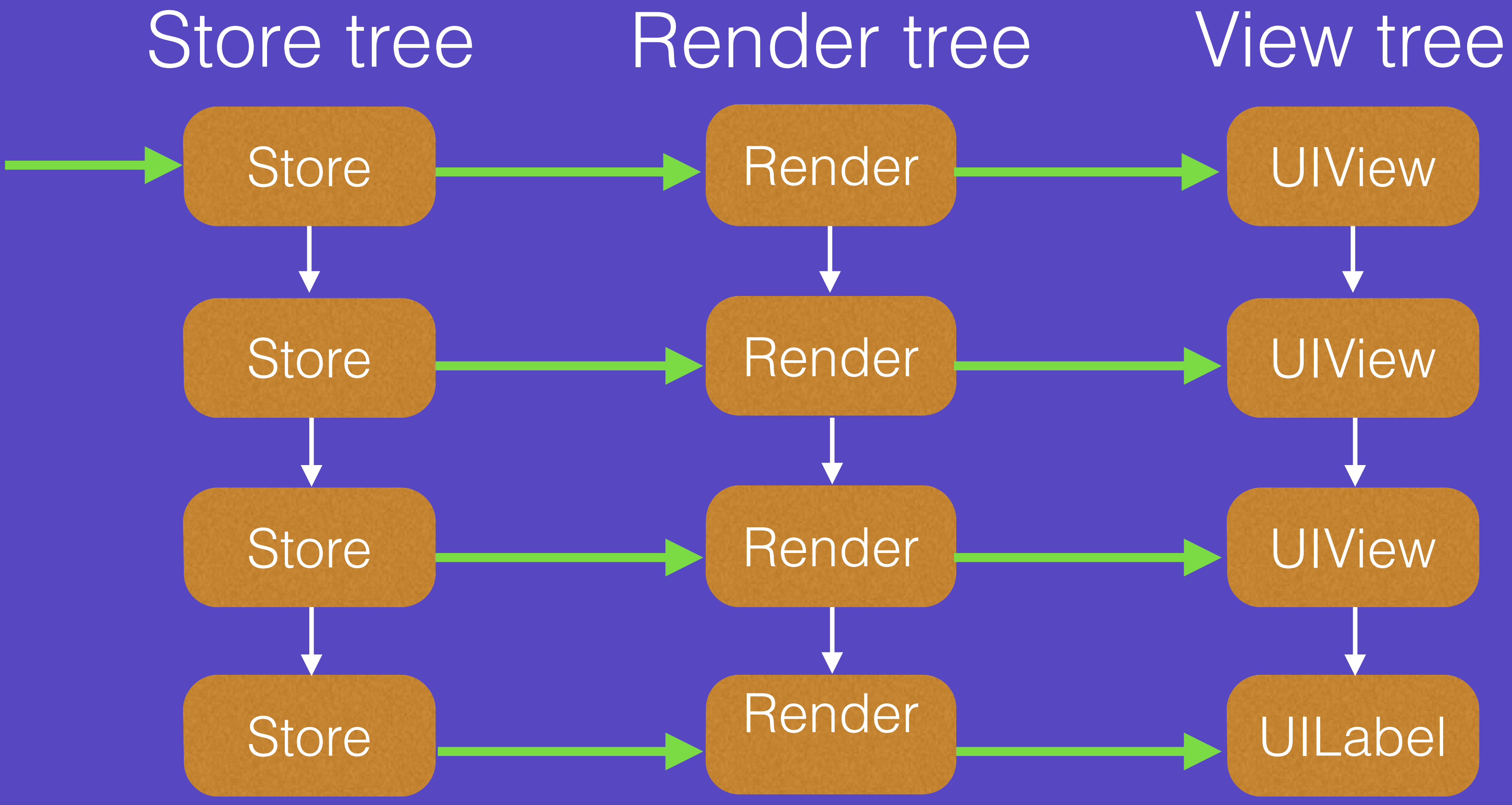
```
- (void)reloadData
{
    self[@"shot"] = @{

        @"author" : @{
            @"avatar" : self.shot.player.avatar_url, // @"htt
            @"title" : self.shot.title, // @"Product Homepage
            @"name" : self.shot.player.name, // @"Unity"
        },

        @"shot" : @{
            @"img" : self.shot.image_url, // @"https://d13yac
        },

        @"attr" : @{
            @"views" : self.shot.views_count, // @"6770",
            @"comments" : self.shot.comments_count, // @"19",
            @"likes" : self.shot.likes_count, // @"591"
        },

        @"comments" : ({

            NSMutableArray * comments = [NSMutableArray array

            for ( DribbbleObject_Comment * comment in self.co
            {
                [comments addObject:@{
                    @"avatar" : comment.player.avatar_url, //
                    @"name" : comment.player.name, // @"Eddy
                    @"text" : comment.body, // @"Just a sugge
                }];
            }

            comments;
        })
    };
}
```



```
<body class="wrapper fill">

    <UICollectionView id="list" name="shot" class="fill" columns="1"

        <UICollectionViewCell name="author" is-static is-row>
            <div class="author-wrapper" onclick="signal('view-profile
                <img class="author-avatar" name="avatar"/>
                <div class="author-attribution">
                    <div class="author-title" name="title">Portfolio
                    <div class="author-subtitle">by <span class="auth
                </div>
            </div>
        </UICollectionViewCell>

        <UICollectionViewCell name="shot" is-static is-row>
            <div class="shot-wrapper">
                <img name="img" class="shot-img" src="http://img.hb.a
                 onclick="signal('view-photo')" />
            </div>
        </UICollectionViewCell>
```

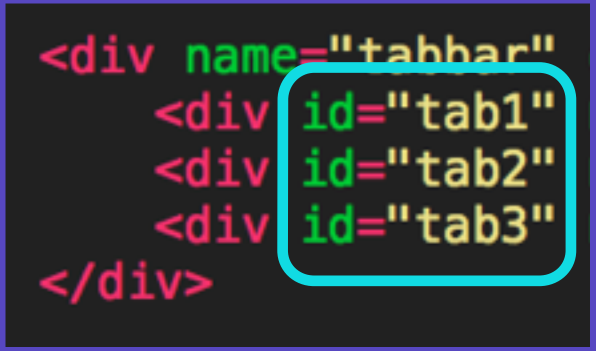
Data binding/query (2)

```
{
    “key” : “value”,
    “key” : “value”,
    “key” : “value”,
    “key” : “value”
}
```

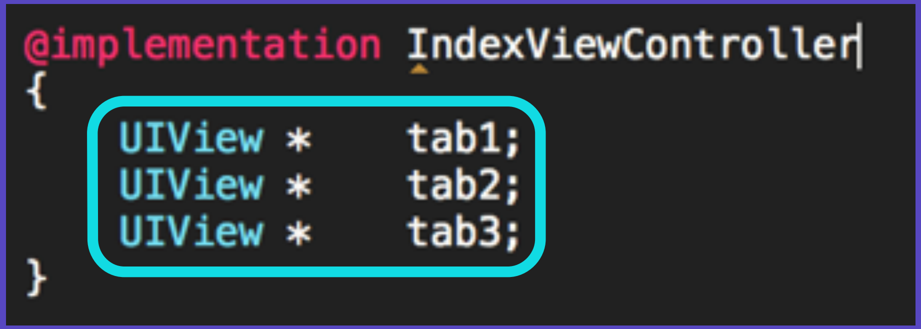


View binding/query (1)

- Property auto binding
- IVAR auto binding



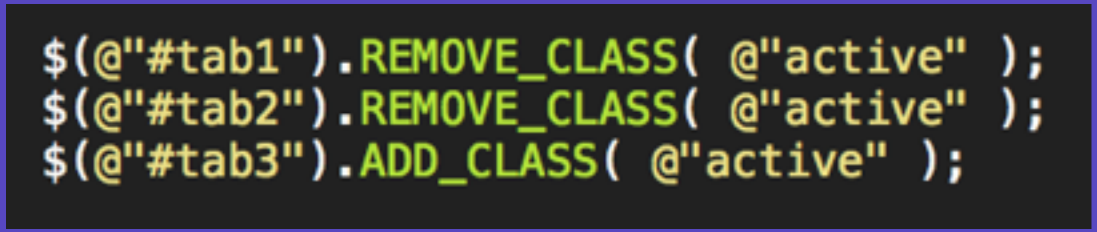
```
<div name="tabbar"
    <div id="tab1"
    <div id="tab2"
    <div id="tab3"
</div>
```



```
@implementation IndexViewController
{
    UIView *    tab1;
    UIView *    tab2;
    UIView *    tab3;
}
```

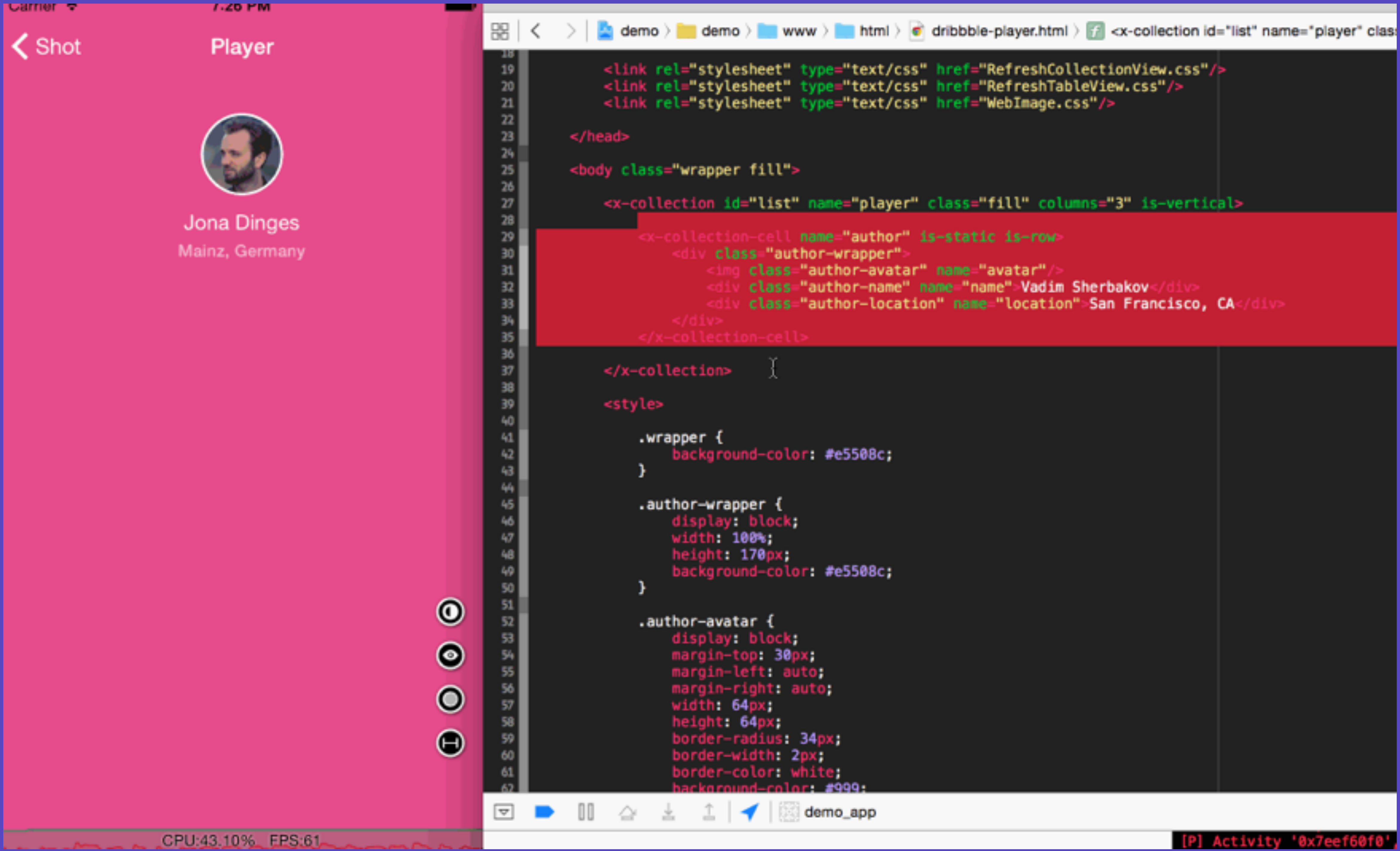
View binding/query (2)

- NativeQuery
- OC version of jQuery



```
$(@"#tab1").REMOVE_CLASS( @"active" );
$(@"#tab2").REMOVE_CLASS( @"active" );
$(@"#tab3").ADD_CLASS( @"active" );
```

Live load



Carrier
7:26 PM
Shot
Player
Jona Dinges
Mainz, Germany
CPU:43.10% FPS:61
demo 〉 demo 〉 www 〉 html 〉 dribbble-player.html 〉 <x-collection id="list" name="player" clas
demo_app

```
        <link rel="stylesheet" type="text/css" href="RefreshCollectionView.css"/>
        <link rel="stylesheet" type="text/css" href="RefreshTableView.css"/>
        <link rel="stylesheet" type="text/css" href="WebImage.css"/>

    </head>

    <body class="wrapper fill">

        <x-collection id="list" name="player" class="fill" columns="3" is-vertical>

            <x-collection-cell name="author" is-static is-row>
                <div class="author-wrapper">
                    <img class="author-avatar" name="avatar"/>
                    <div class="author-name" name="name">Vadim Sherbakov</div>
                    <div class="author-location" name="location">San Francisco, CA</div>
                </div>
            </x-collection-cell>

        </x-collection>

        <style>

            .wrapper {
                background-color: #e5508c;
            }

            .author-wrapper {
                display: block;
                width: 100%;
                height: 170px;
                background-color: #e5508c;
            }

            .author-avatar {
                display: block;
                margin-top: 30px;
                margin-left: auto;
                margin-right: auto;
                width: 64px;
                height: 64px;
                border-radius: 34px;
                border-width: 2px;
                border-color: white;
                background-color: #999;
```

```
[P] Activity '0x7eef60f0'
```

Plan

- 2015-Jun
    - Write more test-cases, give out good enough compatibility.
- 2015-Sep
    - AppStore top 100 UI template, all free, and easy to use.
- 2015-Dec
    - Android version, JS support.
- 2016-Mar
    - Support chrome/safari.

github.com search `samurai-native`

Finally

- Positioning of your team
  - Transition from a web-app developer
  - Transition from a native-app developer
- Architecture of your app

Fin.

Presented by Geek-Zoo Studio 2015 @ QCON

Author

@老郭为人民服务

@Qfish为人民服务

Special thanks

www.geek-zoo.com

Material provider

https://www.thenounproject.com/