

Samurai-Native架构设计与项目构建

Geek Zoo Studio @ GMTC 2016, Beijing



促进软件开发领域知识与创新的传播



关注InfoQ官方信息
及时获取移动大会演讲
视频信息



[深圳站] 2016年07月15-16日
咨询热线: 010-89880682



[上海站] 2016年10月20-22日
咨询热线: 010-64738142

@老郭为人民服务 | <https://github.com/gavinkwoe>

来自 Geek-Zoo Studio, BeeFramework & Samurai-Native 作者.

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

议程

- 历史回顾
- Demo
- 架构设计
- 实现细节
- QA

历史回顾

1994

Web 出现



- 21年前
 - Netscape navigator 浏览器发布
 - 使用 **HTML+CSS** 编写网页

2008

Native App 出现



- 8年前
 - Apple iOS 发布
 - 使用 OC 开发原生App

2011

Hybrid App 出现



- 5年前

- Adobe PhoneGap 发布
- 使用 HTML+CSS 开发App

2015

Semi-Hybrid App 出现



- 1年前
 - Facebook React-Native 发布
 - 使用 **Reactjs** 开发原生App

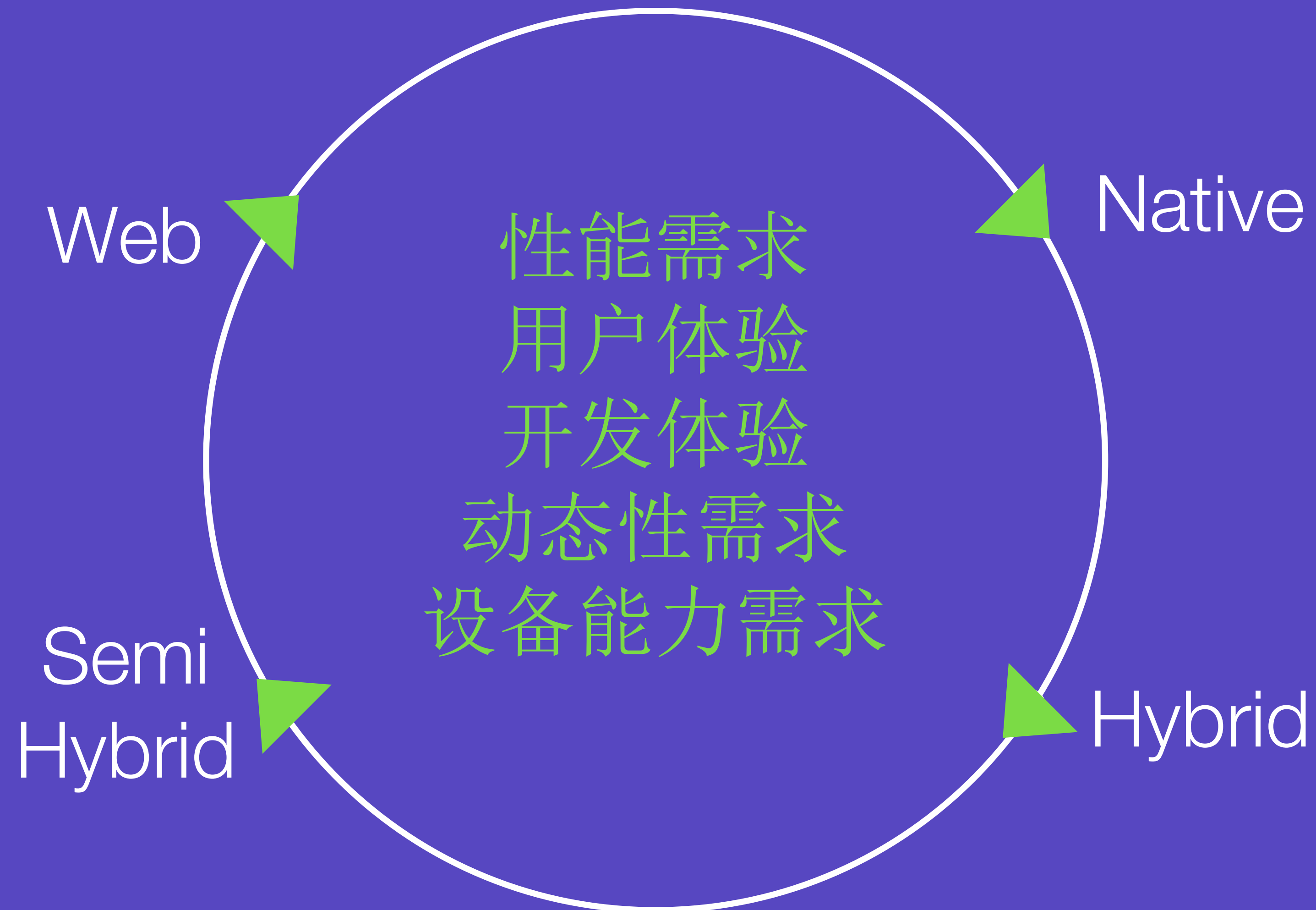
2015-2016

更多垂直框架出现



- 最近几个月
 - WEEX 发布
 - JSPatch 发布

技术在不断新陈代谢，Hybrid模式更适合移动场景



最近一年技术栈也已经发生变化，新的技术涌现



动态视图



动态逻辑



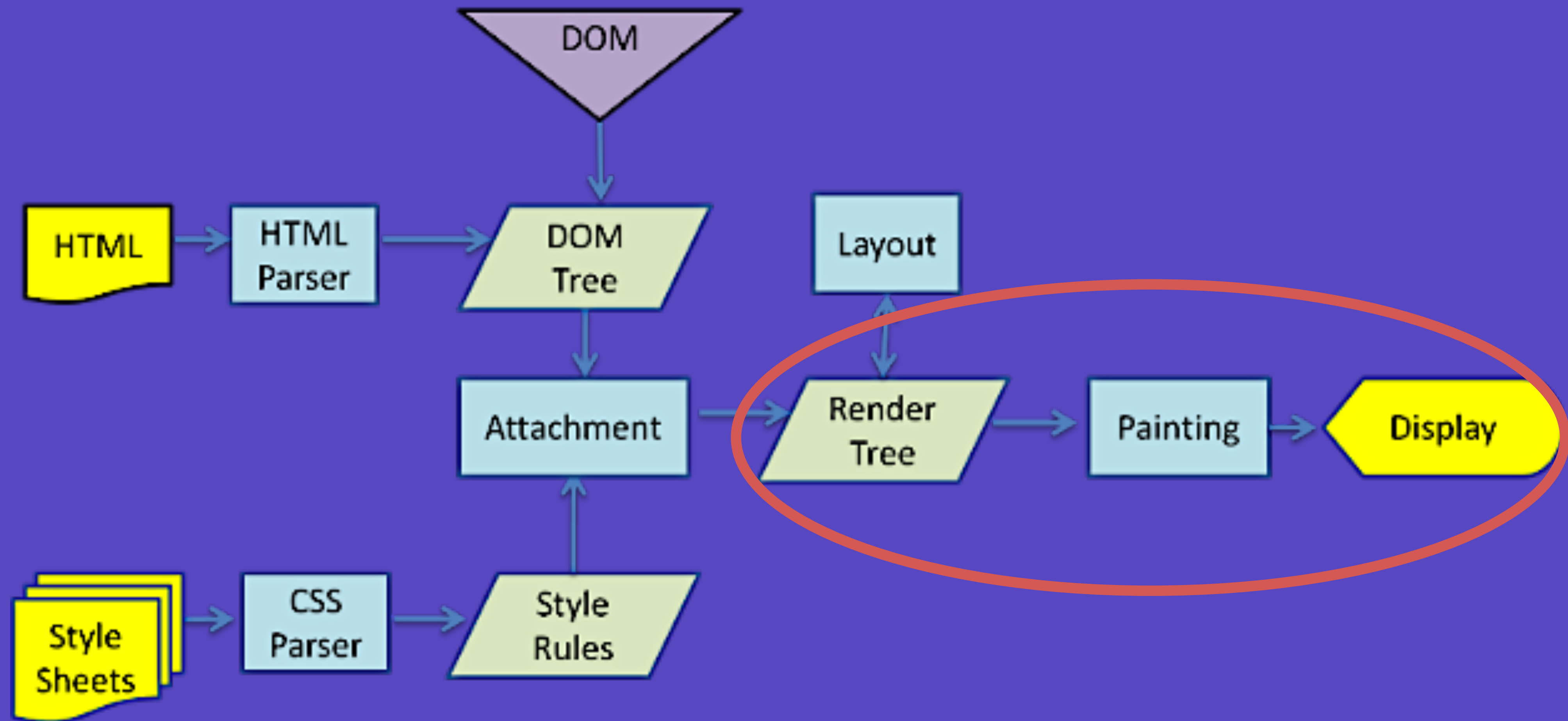
动态补丁



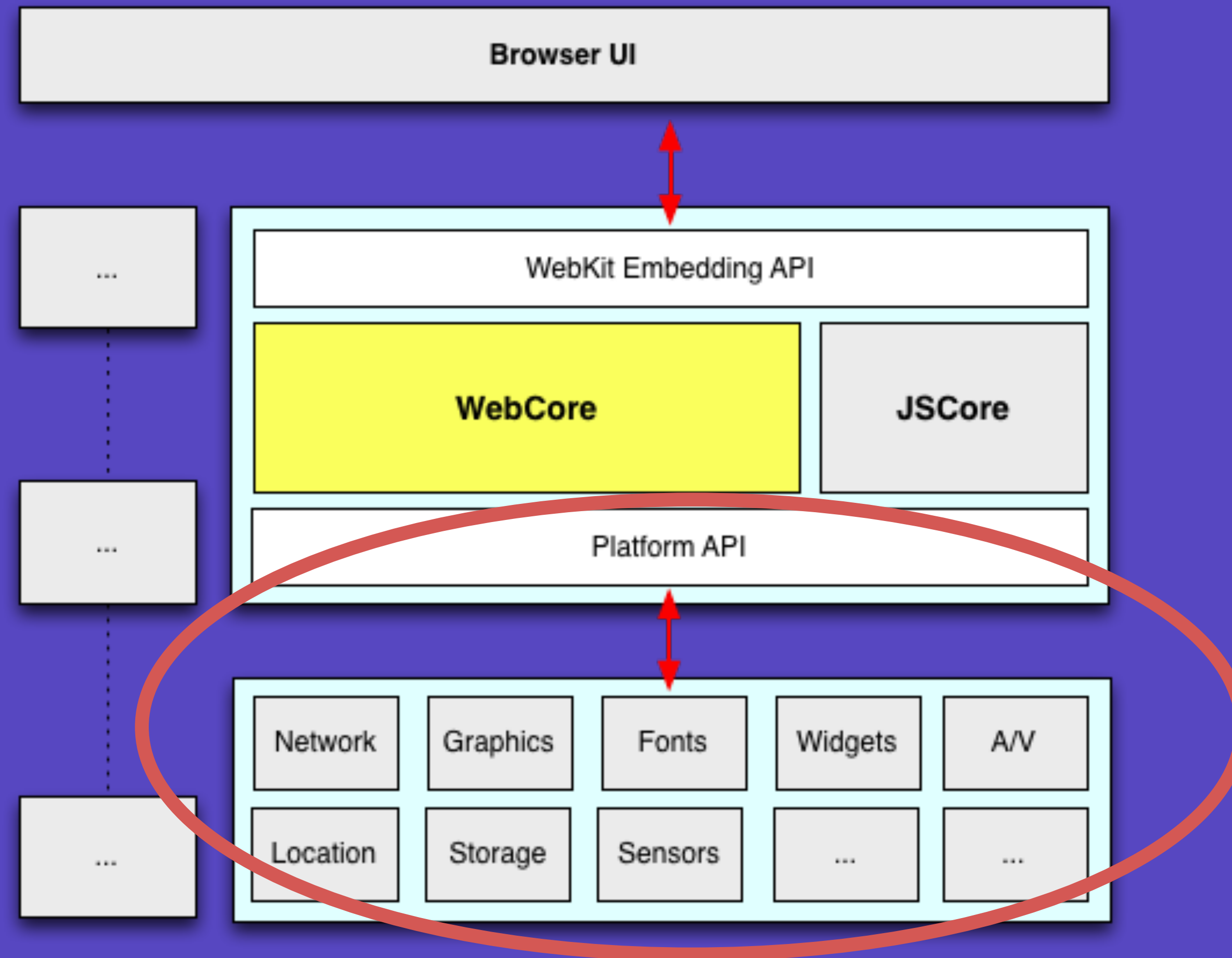
通过分析软件包资源

为什么 WebKit 不好满足这些技术要求？

Painter做为渲染后端，无法直接还原用户体验

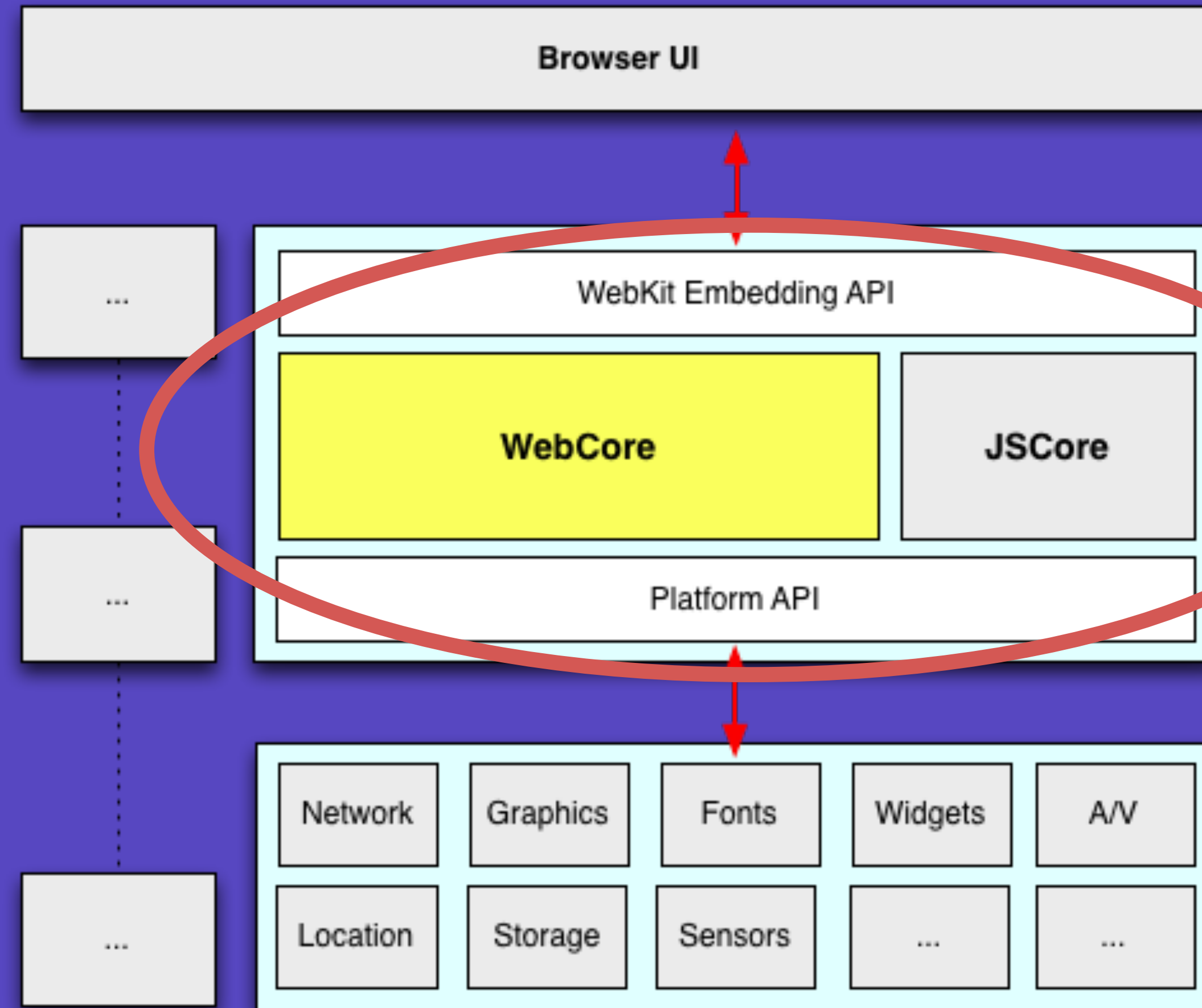


对开发者完全黑盒，缺失设备/运行时调用能力



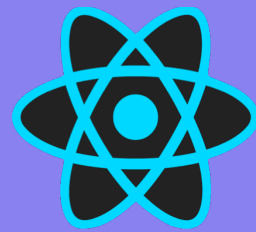
历史包袱过多，W3C规范定制过慢，落后于实际需求

CSS 1.0
CSS 2.0
CSS 3.0
HTML 2.0
HTML 3.2
HTML 4.0
HTML 5.0
XHTML



URL / URI
Protocols
Networks
Graphics
Audio / Video
Math
Mobile Web

从近年技术发展看，WebKit 正在被重新发明



App



View

JSPatch

Business

JSPatch

Foundation

Hybrid app

V.S.



Browser



WebCore



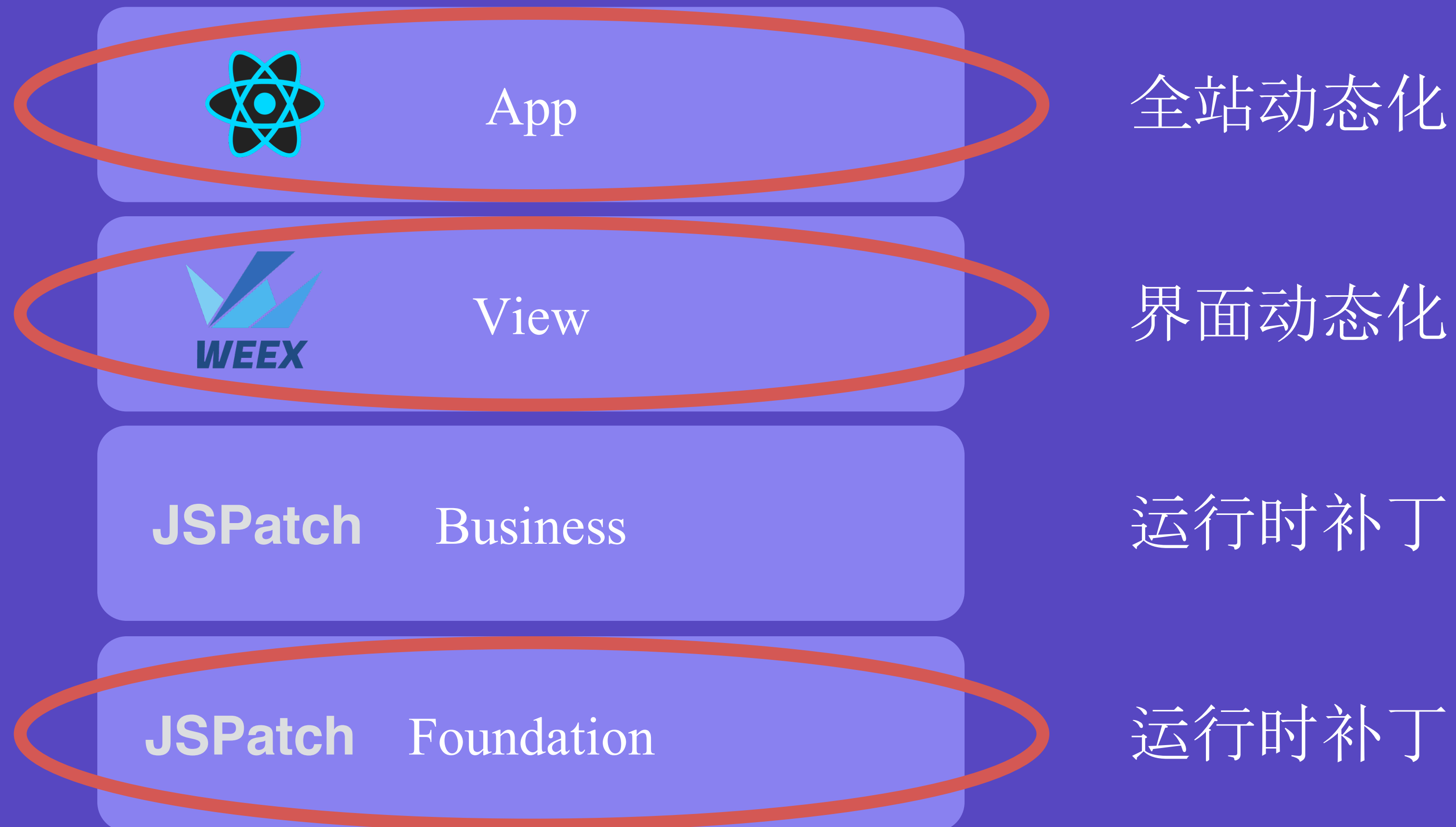
JSCore

**Platform
API**

Foundation

Web app

通过不同垂直场景的解决方案，解决不同角度问题



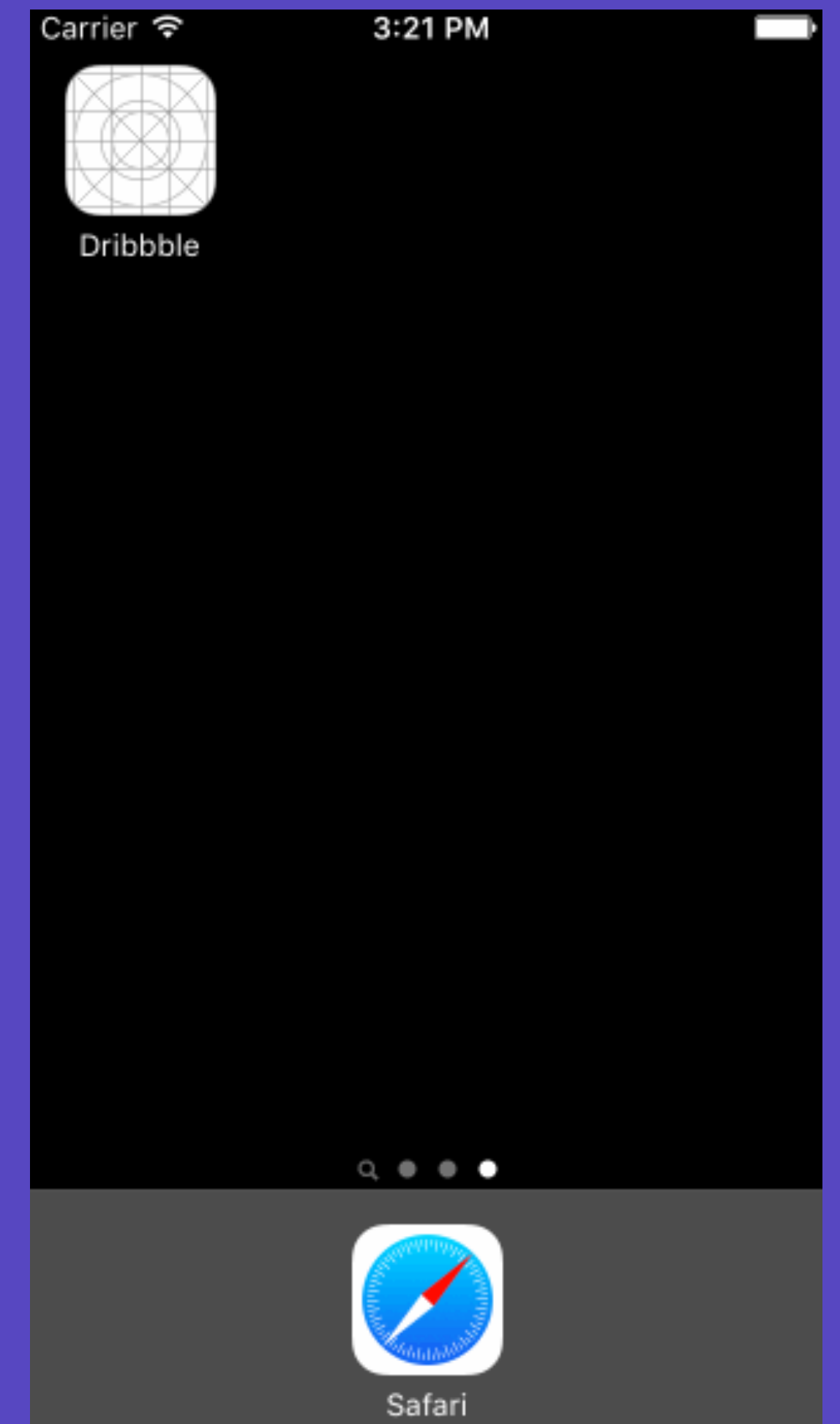
第三方软件正在解决实际需求



Bring web standards to native platform

Demo

- 通过 JS/HTML/CSS 编写iOS App
- 生成纯原生 UI 界面
- 可通过第三方扩展
 - SDWebImage
 - MBProgressHUD
 - AFNetworking



```

<body>
  <RefreshCollectionView class="list">
    <UICollectionViewCell class="cell">
      <SDWebImageView class="image"></SDWebImage
    </UICollectionViewCell>
  </RefreshCollectionView>
</body>

```

HTML + Native Component

```

<style>
  .list {
    display: block;
    width: 100%;
    height: 100%;
    margin: 0 3px;
  }

```

```

  .cell {
    width: 100%;
    height: auto;
  }

```

CSS + Property Mapping

```

  .image {
    width: 100%;
    height: 140px;
    background-color: #aaa;
    -uikit-contentMode: UIViewContentModeScaleAspe
    -uikit-layer-masksToBounds: YES;
  }
</style>

```

```

include( '/js/api/DribbbleAPI.js' );
include( '/js/model/DribbbleModel.js' );

```

Include file & Require class

```

require( 'MBProgressHUD' );

```

```

defineClass( 'DribbbleIndexPage', 'SamuraiActivity', ['MBProgressHUDDelegate'],

```

Define Native Class

```

{
  model: null,
  list: null,
  hud: null

```

```

},

```

```

{

```

```

  'onCreate' : function()

```

OC > JS Callback

```

{

```

```

  var activity = self;

```

```

  var model = new Dribbble.Model.ShotList( 'popular' );

```

```

  model.onLoading = function( reset ){

```

```

    if ( reset )

```

```

    {

```

```

      [activity showLoading];

```

```

    }

```

```

  };

```

```

  model.onLoaded = function(){

```

```

    [activity.props.list stopLoading];

```

```

    [activity reloadData];

```

```

    [activity hideLoading];

```

```

  };

```

```

  model.onError = function(){

```

```

    [activity.props.list stopLoading];

```

```

    [activity reloadData];

```

```

    [activity hideLoading];

```

```

  };

```

```

  self.props.model = model;

```

```

  [self setNavigationBarTitle:'Dribbble'];

```

```

  [self loadTemplateFile:'/html/DribbbleIndexPage.html'];

```

```

},

```

Using Obj-C Syntax

JS > OC Invocation

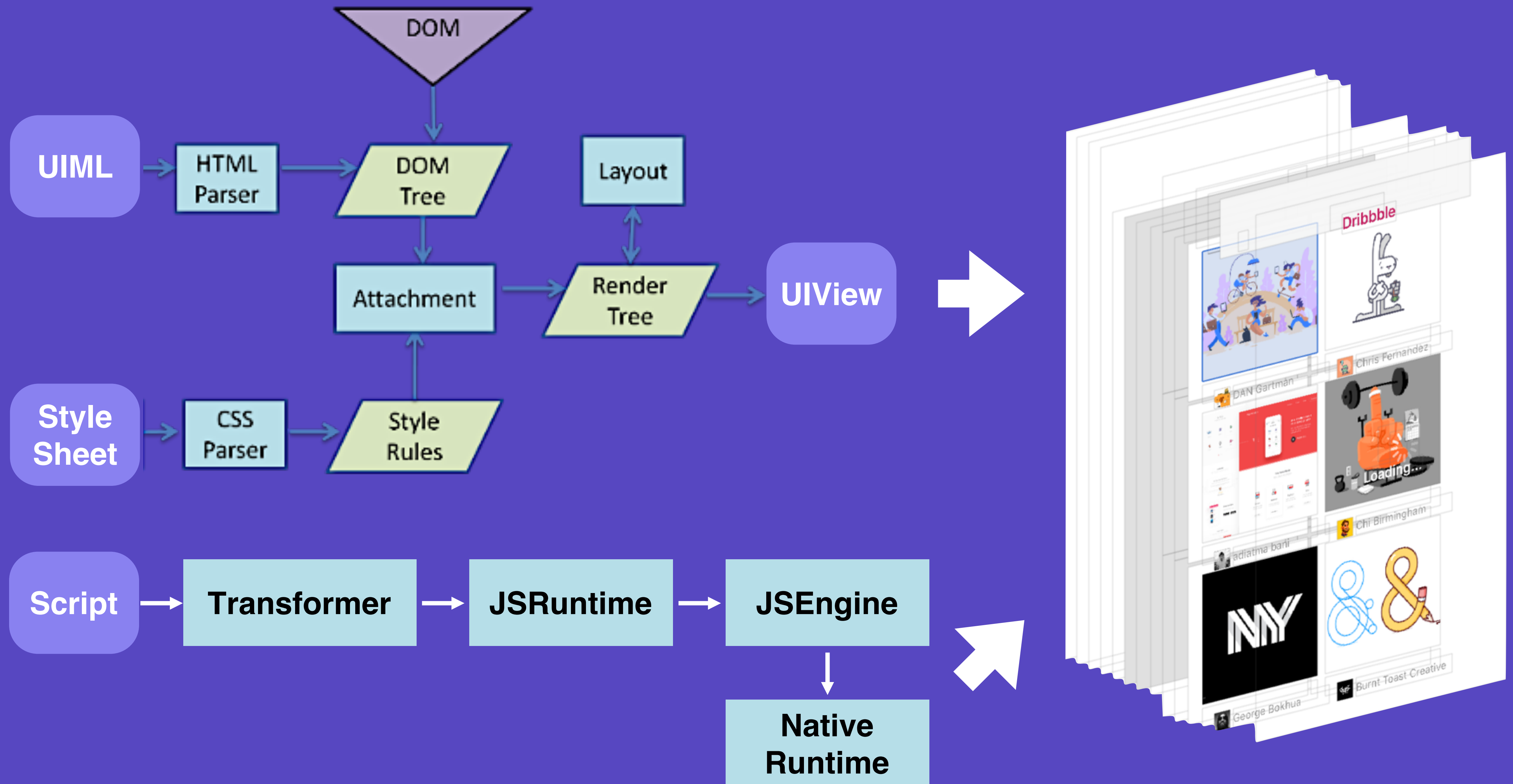
HTML
Layout

JS
Logic



CSS
Style

Native
Ability



架构概览

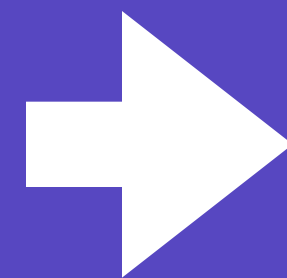


WebCore 实现

WebCore 作用

```
<UIImageView/>  
  
UIImageView {  
  width: 100%;  
  color: red;  
}
```

DSL Code



UIView Tree

Native UI

WebCore 架构

Render workflow

Template

HTML
UserAgent

HTML Render

HTML Layout

Document workflow

HTML Doc

HTML DOM

HTML Element

Layout algorithm

CSS Resolver

CSS Value

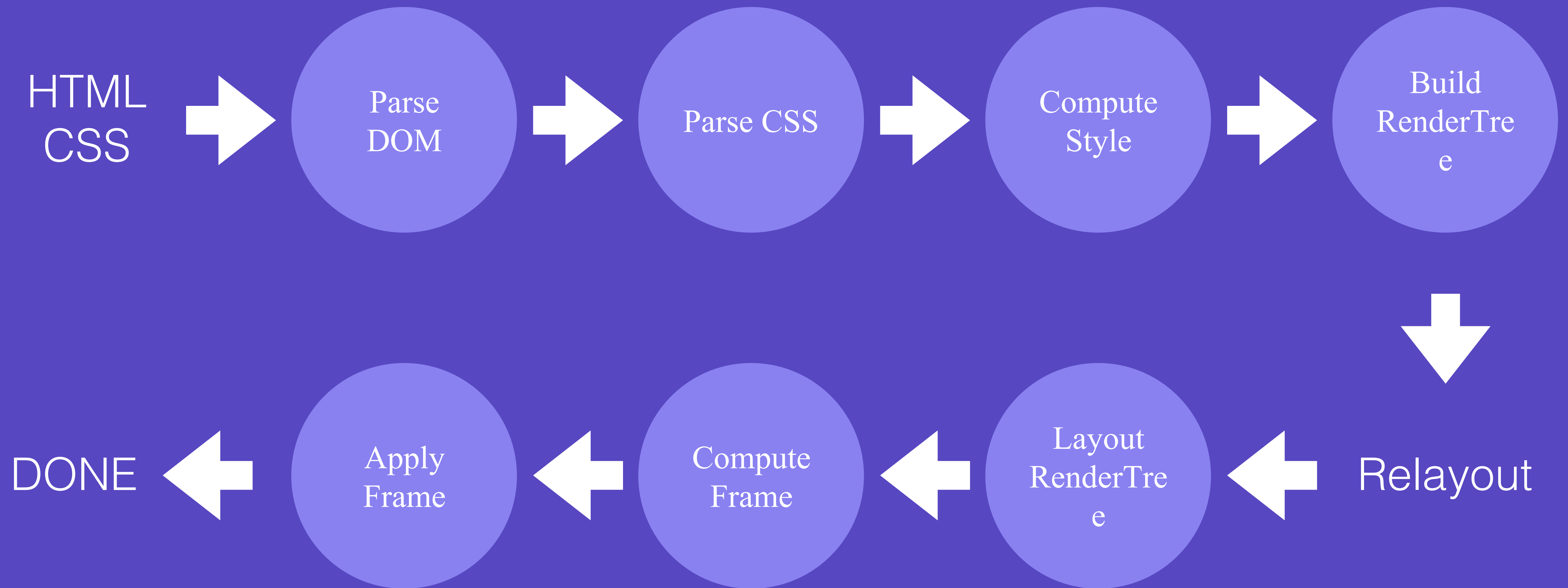
CSS StyleSheet

UserAgent Config

gumbo parser

katana parser

WebCore workflow



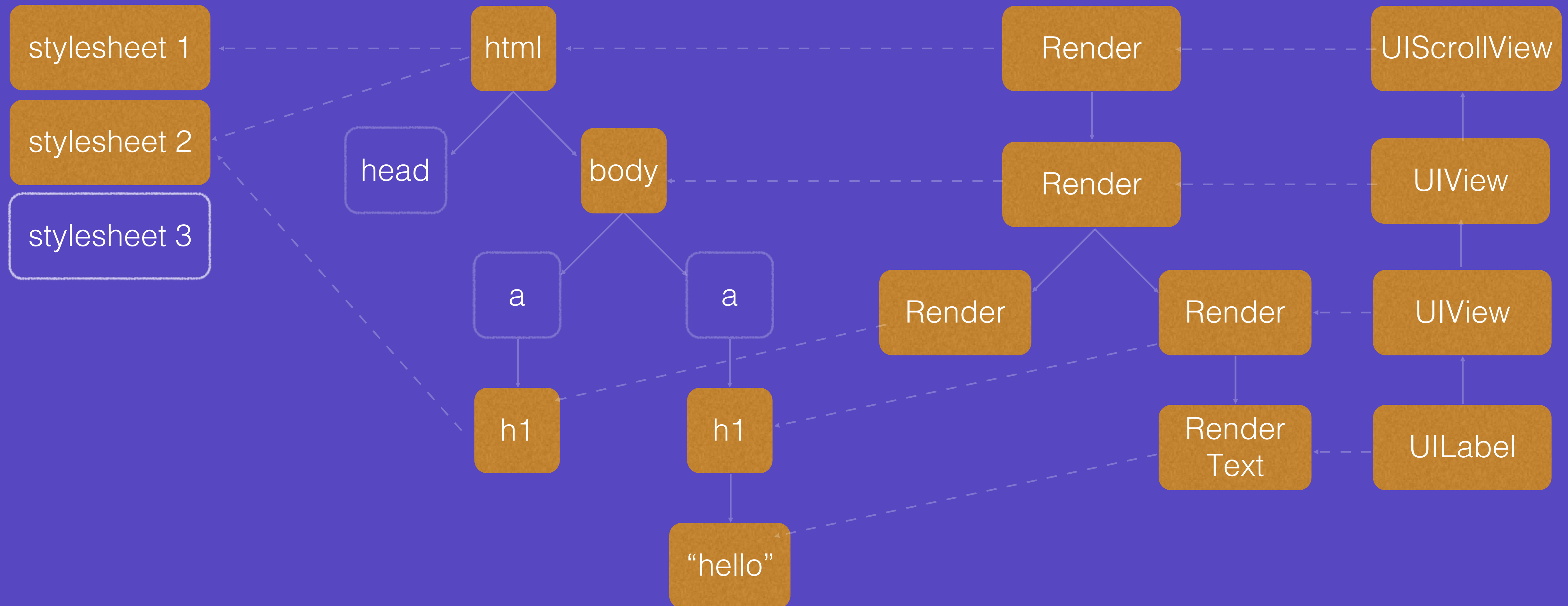
WebCore 树结构

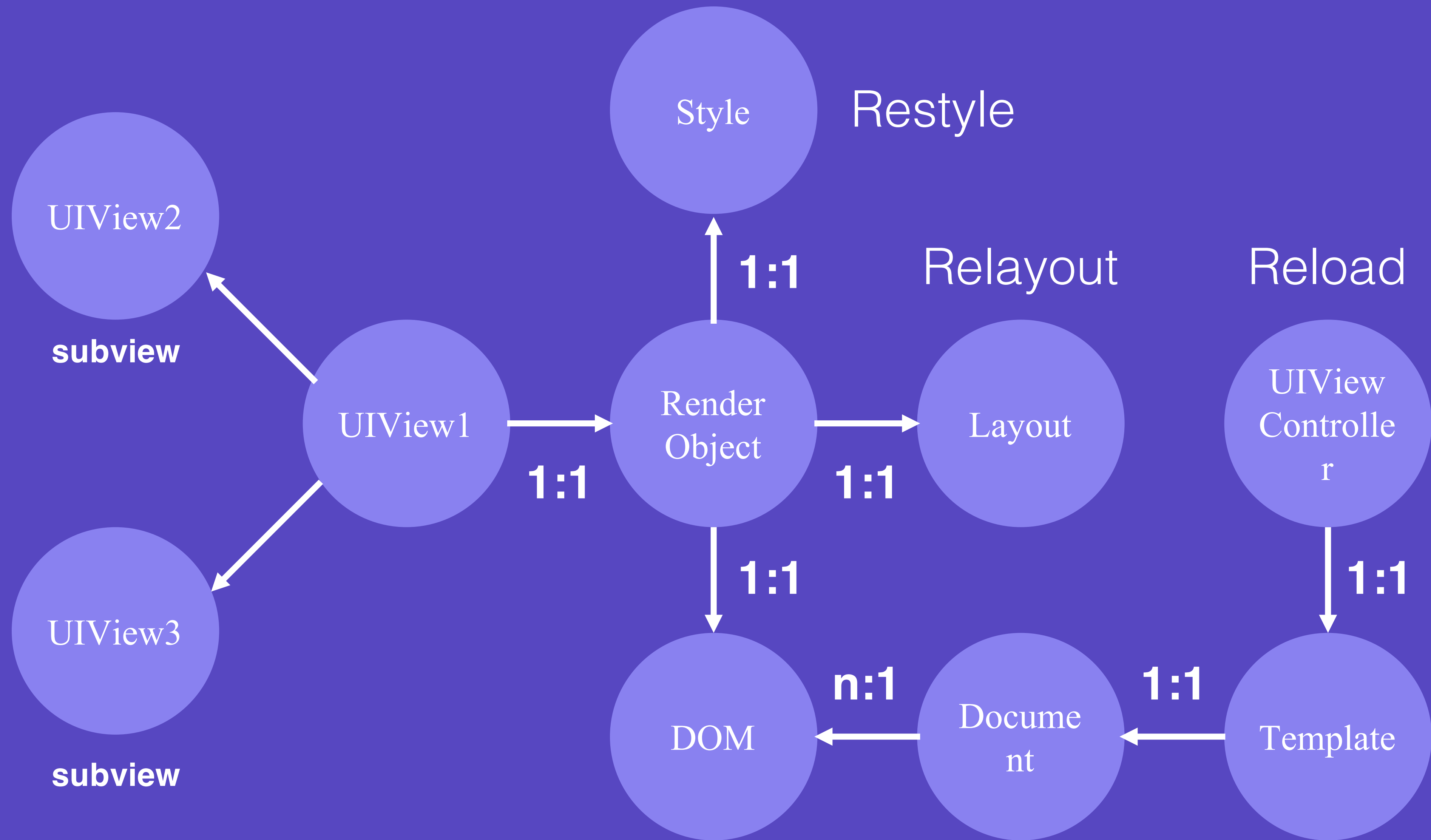
Stylesheets

Dom tree

Render tree

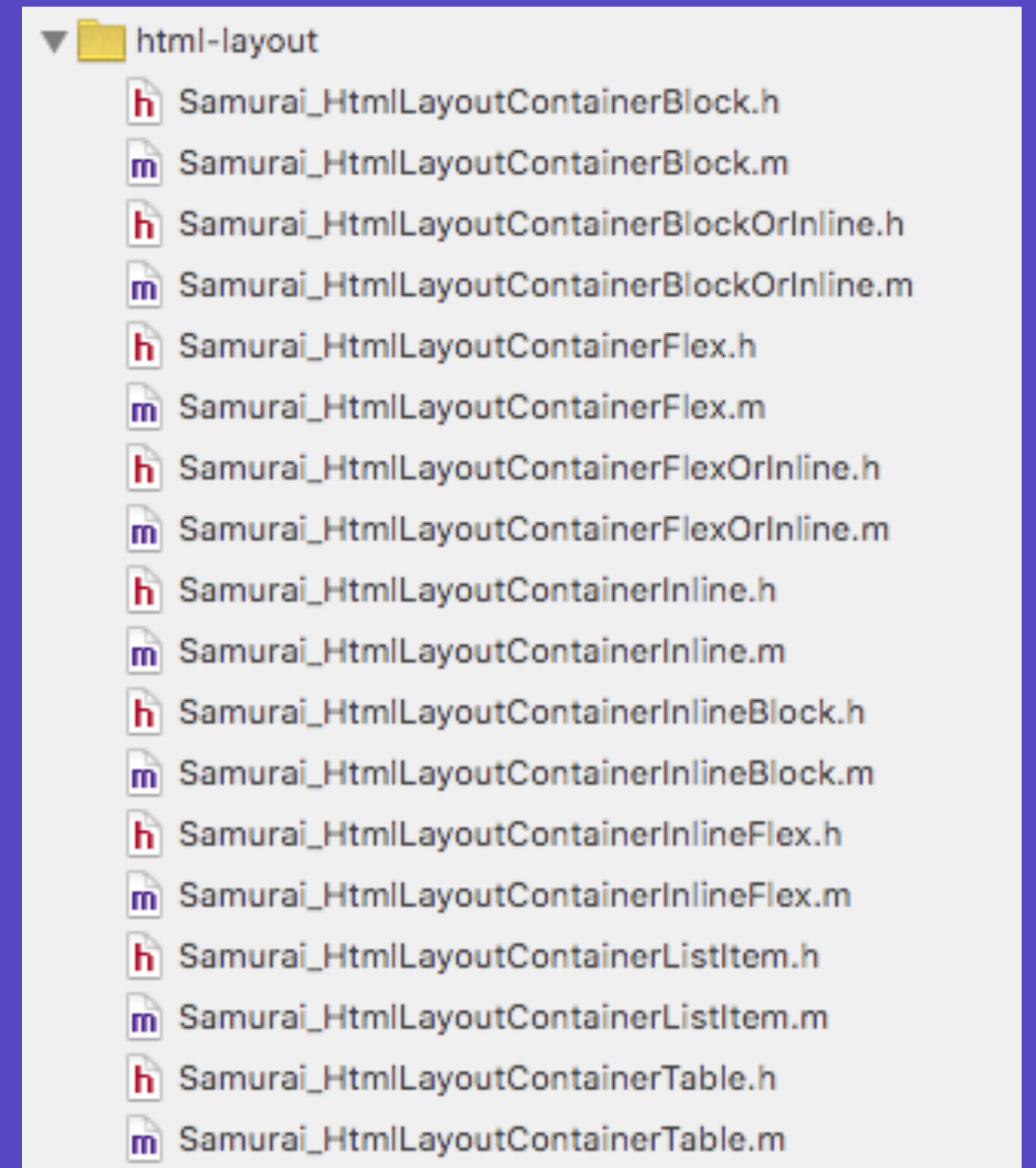
View tree



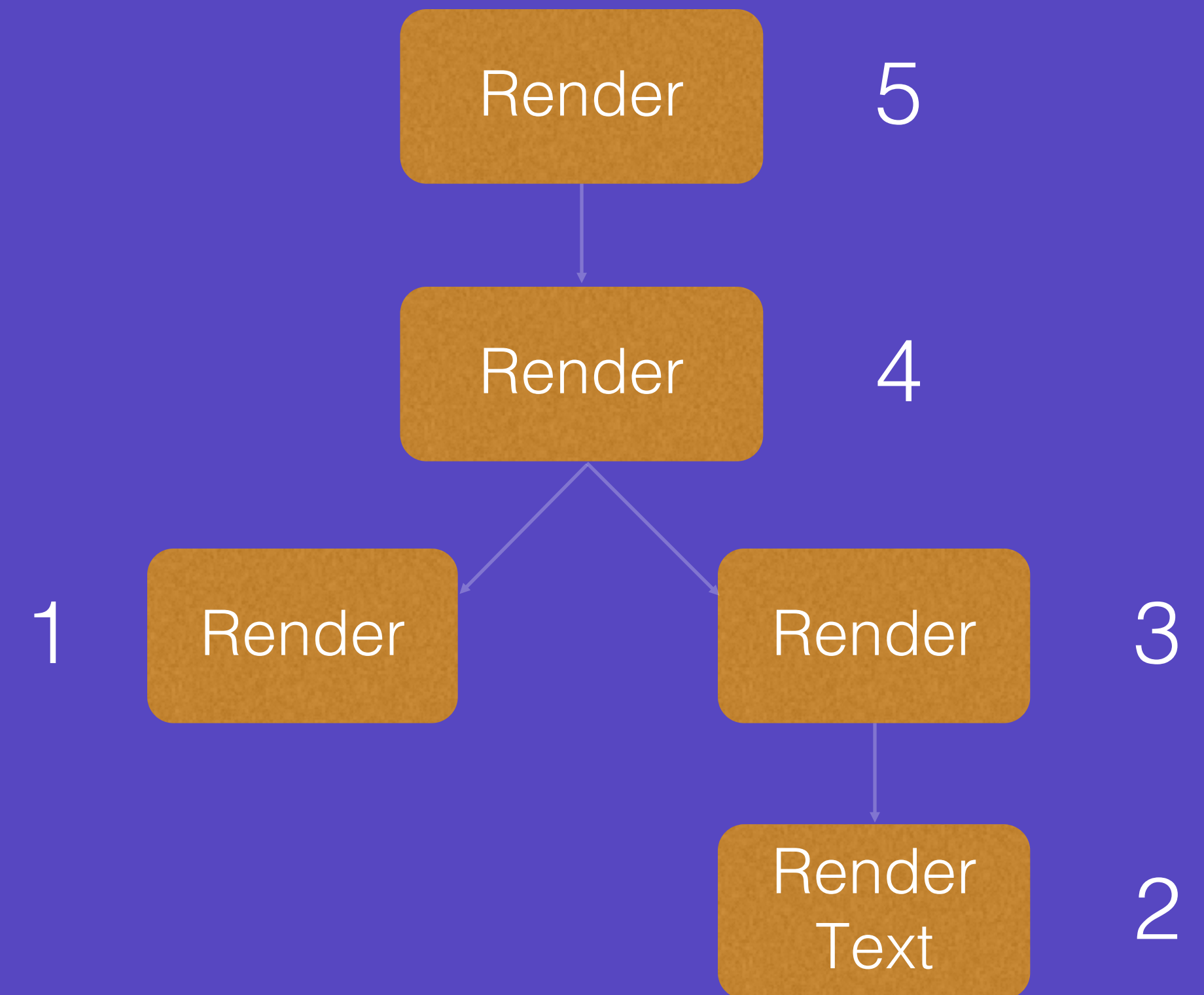
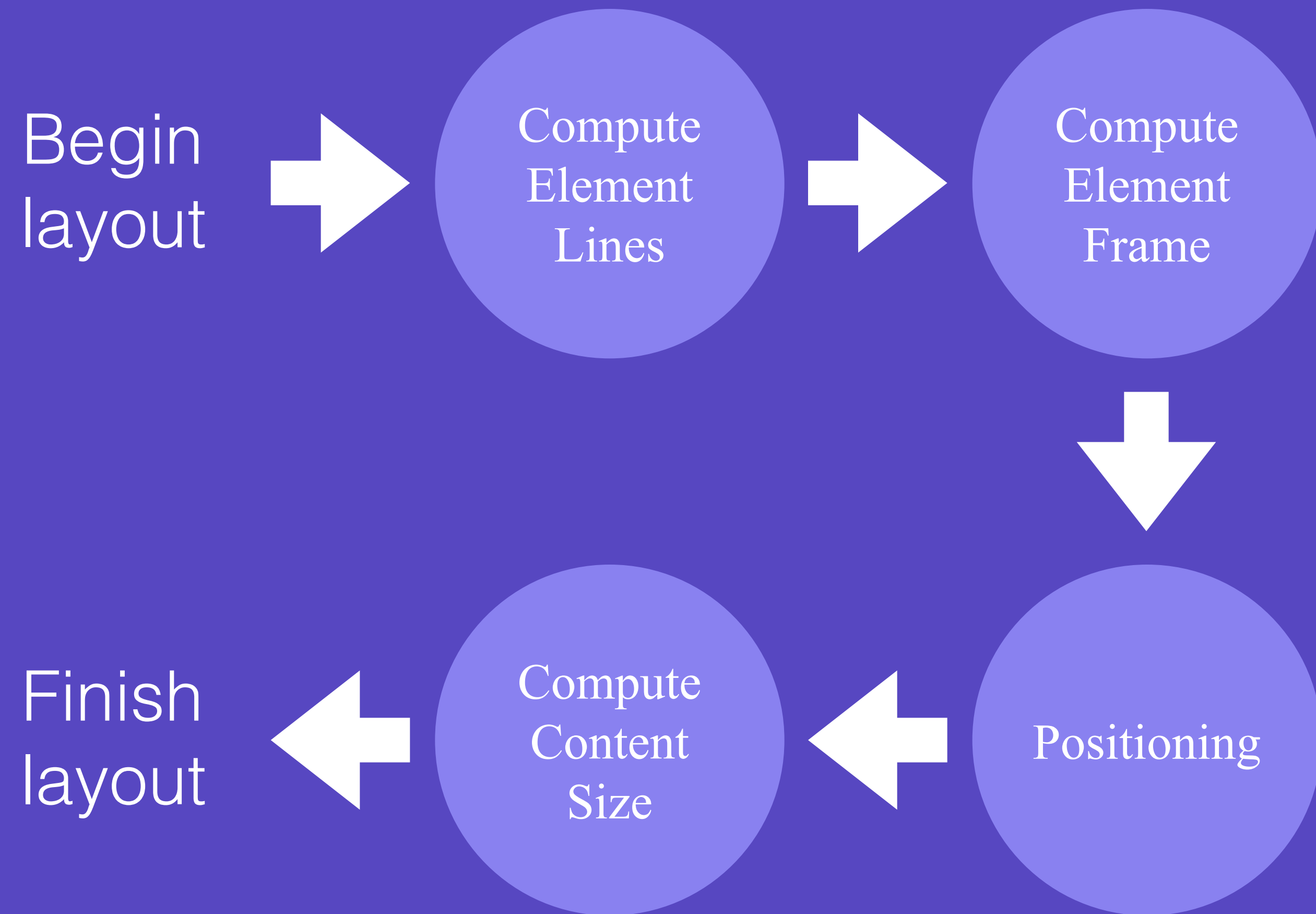


布局算法 (1)

- display: inline/block/flex/table
 - Inline layout
 - Block layout
 - Flex layout
 - Table layout



布局算法 (2)



用户代理

- UserAgent Setting
 - 默认 font / color / metrics
- UserAgent StyleSheet
 - 默认样式及元素与组件关系
 - samurai-html-hierarchy.css
 - samurai-html-useragent.css

```
body {  
    -samurai-view-hierarchy: 'tree';  
    -samurai-view-class: 'SamuraiHtmlElementBody';  
}  
  
article {  
    -samurai-view-hierarchy: 'tree';  
    -samurai-view-class: 'SamuraiHtmlElementArticle';  
}  
  
aside {  
    -samurai-view-hierarchy: 'tree';  
    -samurai-view-class: 'SamuraiHtmlElementAside';  
}  
  
div {  
    -samurai-view-hierarchy: 'tree';  
    -samurai-view-class: 'SamuraiHtmlElementDiv';  
}
```

对原生组件支持

<UIActivityIndicatorView/>

<UITextField/>

<UIScrollView/>

<UITableView/>

<UIButton/>

<UITextView/>

<UIProgressView/>

<UITableViewCell/>

<UIImageView/>

<UISlider/>

<UISlider/>

<UICollectionView/>

<UIPageControl/>

<UISwitch/>

<UIStepper/>

<UICollectionViewController/>

<UILabel/>

<UIView/>

<UIWebView/>

...

Support all of UIKit components

对原生组件扩展

- <UIButton/>
- 两步搞定
 - html_applyDom:
 - 解析DOM数据
 - html_applyRender:
 - 应用计算好的样式

```
@implementation UIButton(Html)

- (void)html_applyDom:(SamuraiHtmlDomNode *)dom
{
    [super html_applyDom:dom];

    self.titleLabel.text = [[dom computeInnerText] normalize];
    self.titleLabel.hidden = NO;

    [self setTitle:self.titleLabel.text forState:UIControlStateNormal];
}

- (void)html_applyRender:(SamuraiHtmlRenderObject *)render
{
    [super html_applyRender:render];

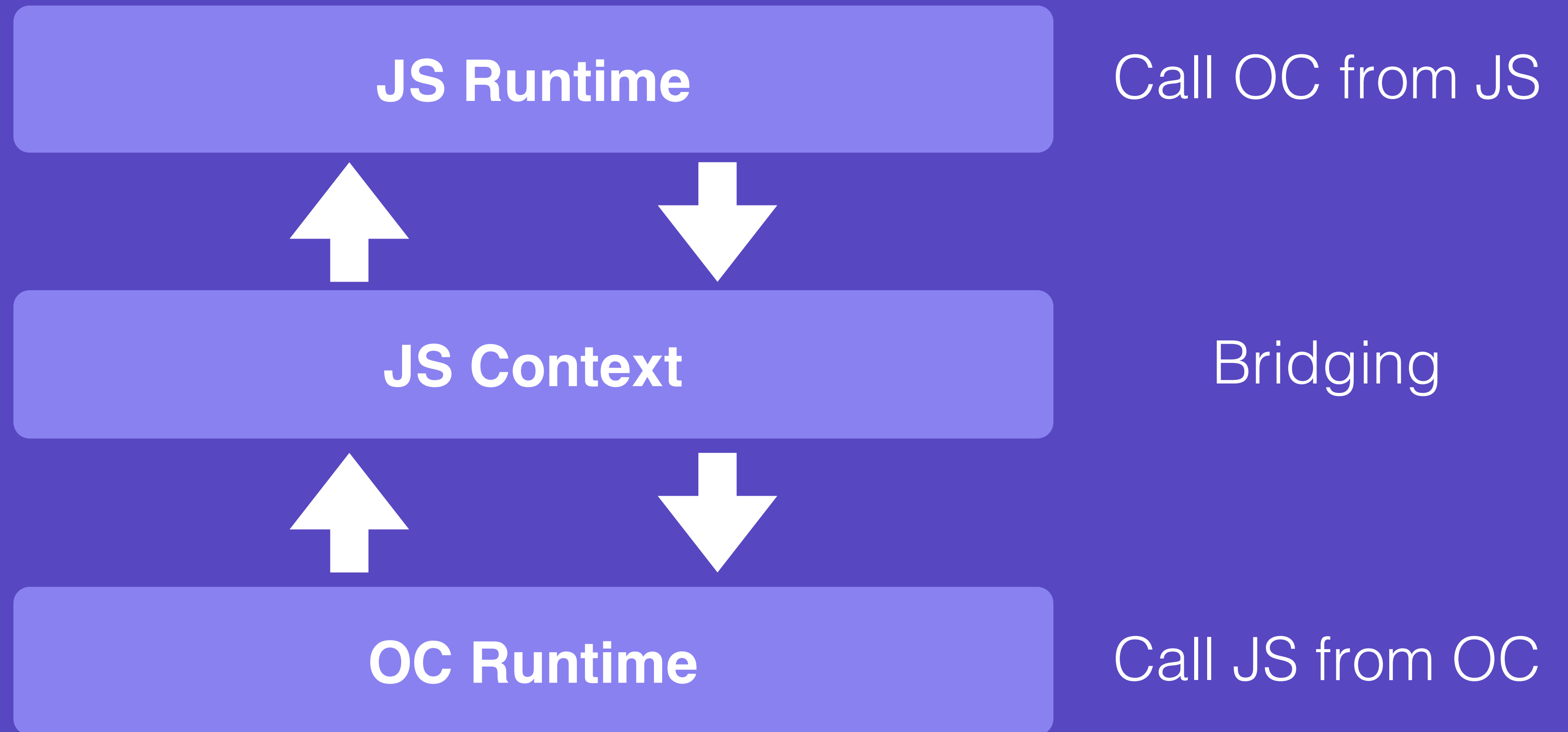
    self.titleLabel.hidden = NO;
    self.titleLabel.frame = CGRectMake( 0.0f, 0.0f, render.computedFrame.size.width, rende
    self.titleLabel.font = render.computedFont;

    switch ( render.computedTextAlign ) {
        case CSSTextAlign_Left:      self.titleLabel.textAlignment = NSTextAlignmentLef
        case CSSTextAlign_Right:     self.titleLabel.textAlignment = NSTextAlignmentRig
        case CSSTextAlign_Center:    self.titleLabel.textAlignment = NSTextAlignmentCen
        case CSSTextAlign_Justify:   self.titleLabel.textAlignment = NSTextAlignmentJus
        default:                     self.titleLabel.textAlignment = NSTextAlignmentLef
    }

    switch ( render.computedWordWrap ) {
        case CSSWordWrap_BreakWord:  self.titleLabel.lineBreakMode = NSLineBreakByWordW
        case CSSWordWrap_Normal:
        default:
        {
            switch ( render.computedTextOverflow ) {
                case CSSTextOverflow_Clip:      self.titleLabel.lineBreakMode = NSLineBrea
                case CSSTextOverflow_Ellipsis:  self.titleLabel.lineBreakMode = NSLineBrea
                default:                        self.titleLabel.lineBreakMode = NSLineBrea
            }
        }
    }
}
```

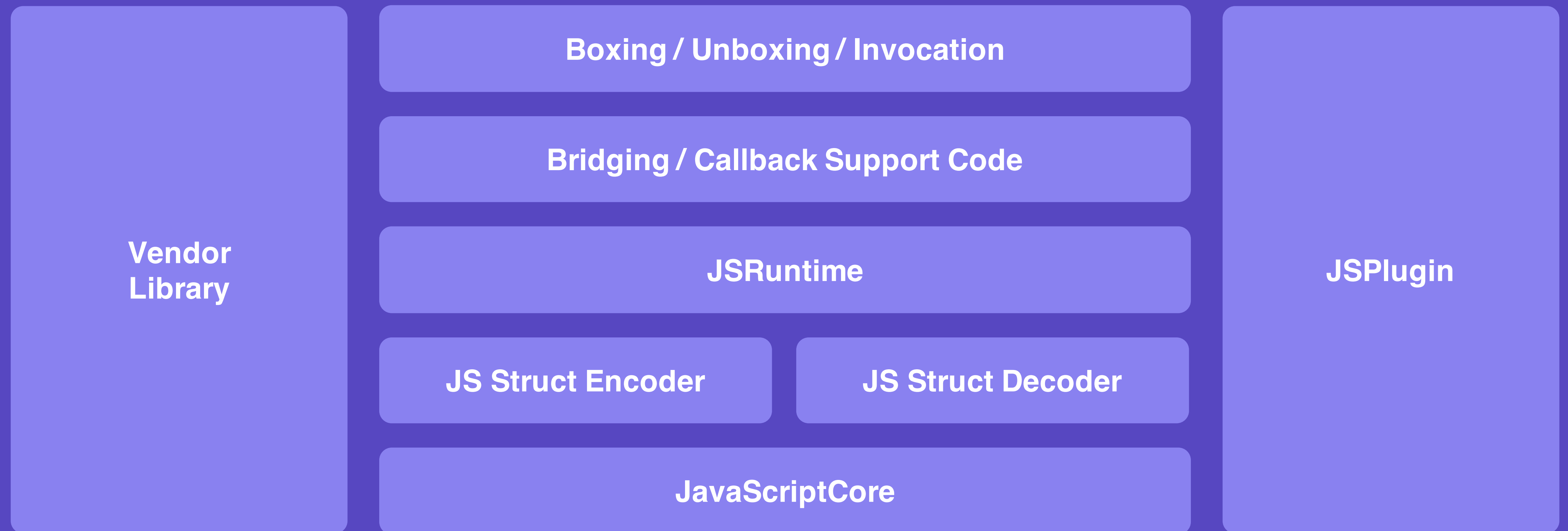
JSCore 实现

JSCore 作用



将底层Runtime所有能力导出给JS Context

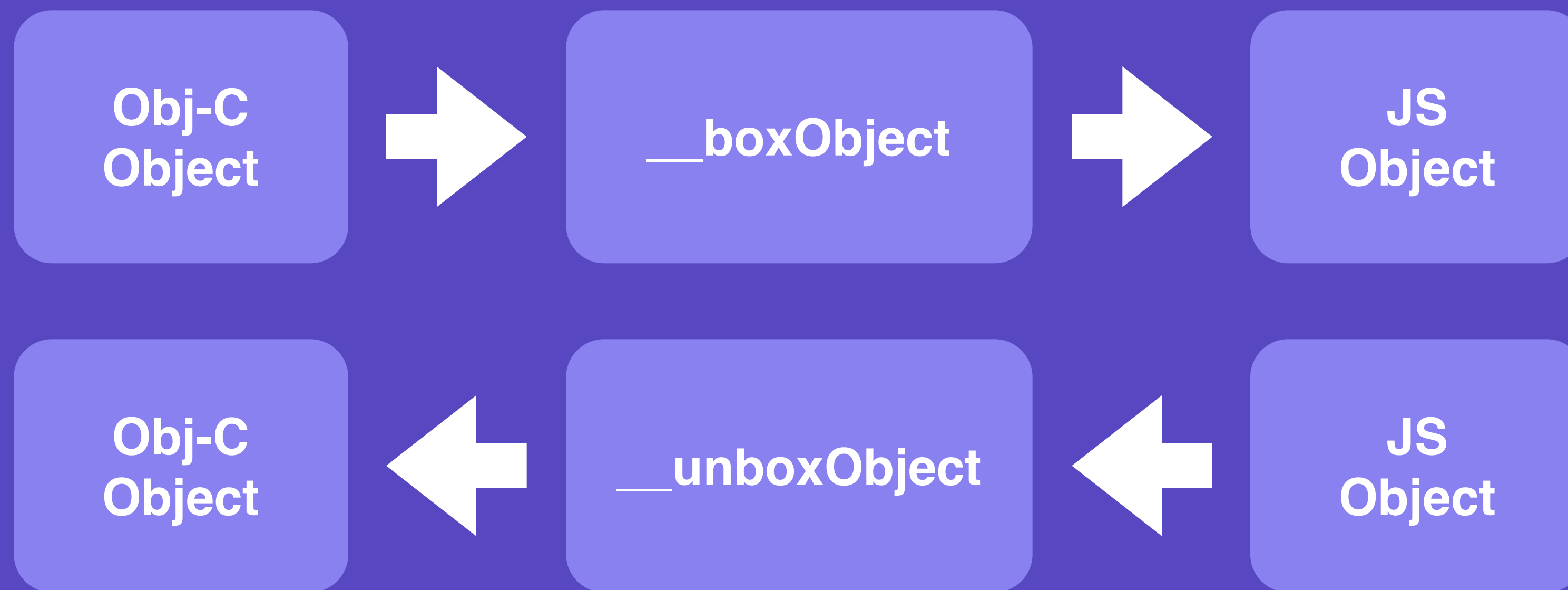
JSCore 架构



JSCore 实现

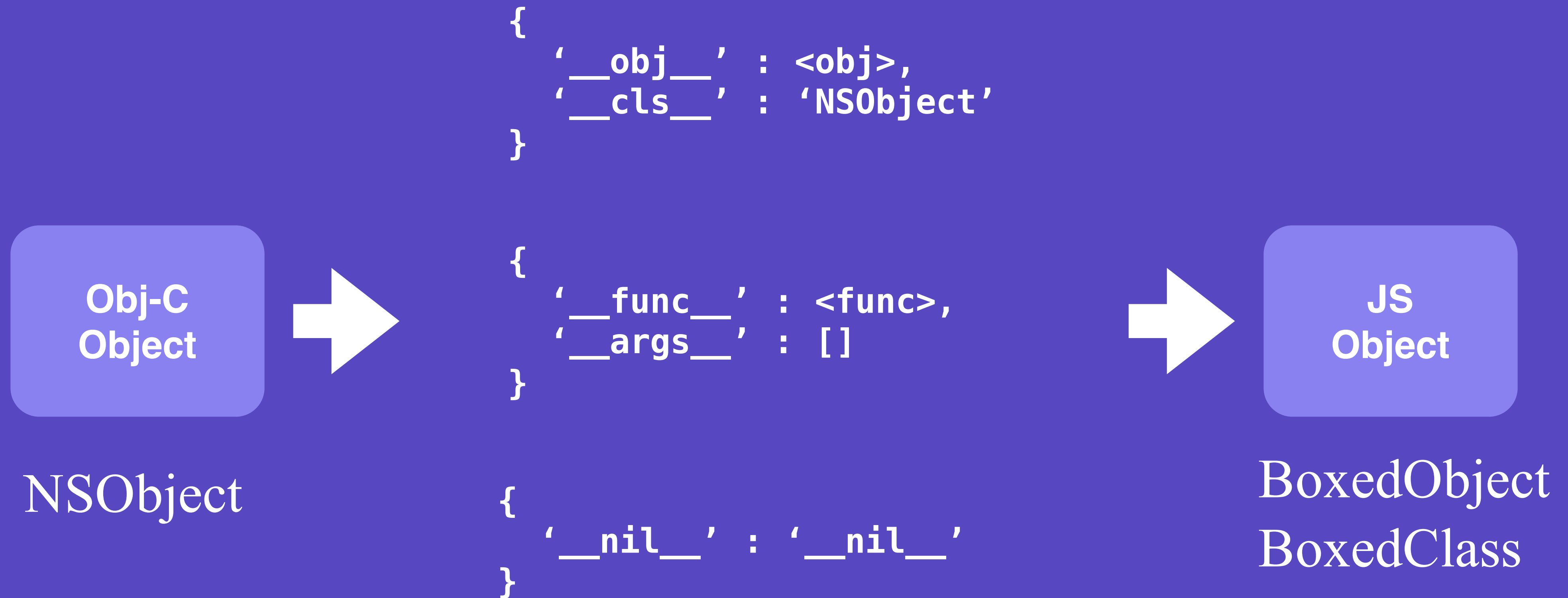
- Native 部分
 - 获取类
 - 创建类
 - 调用方法
 - 读写属性
 - 读写变量
 - 加载文件
- Runtime 部分
 - 方法Hook
 - 方法重定向
 - 对象类型转换
 - 返回值处理
 - Block支持
 - Context
- JS 部分
 - BoxedObject
 - BoxedClass
 - 打包/解包对象
 - 分发/处理回调
 - 引用原生类
 - 引用文件

对象传递 Boxing/Unboxing

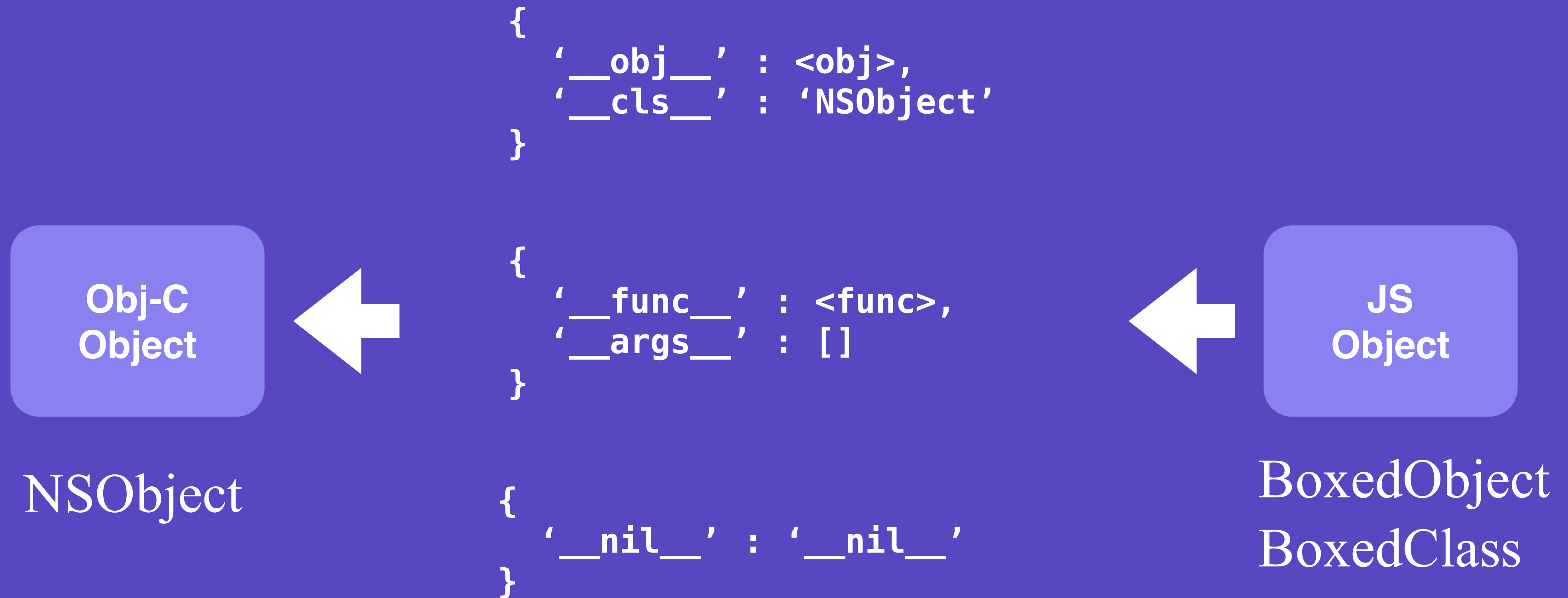


Different from JSPatch, boxing/unboxing in JS side

对象打包 Boxing



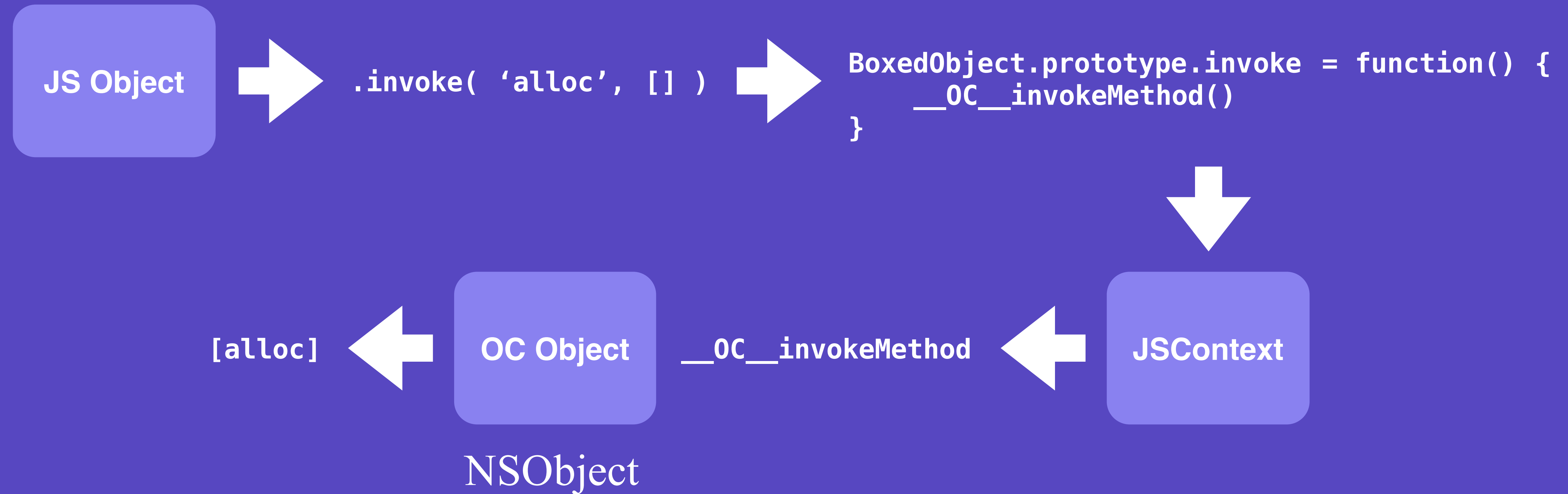
对象解包 Unboxing



从 JS 到 OC 调用 (1)

BoxedObject

BoxedClass



从 JS 到 OC 调用 (2)

```
BoxedObject.prototype.invoke = function( methodName, methodArgs )
{
    var classInfo = global.importedClasses[this._className];
    if ( false == __isNone( classInfo ) )
    {
        var methodFunction = classInfo.instanceMethods[methodName];
        if ( false == __isNone( methodFunction ) )
        {
            var _self = global.self;

            global.self = this;

            var returnValue = methodFunction.apply( this, methodArgs );

            global.self = _self;

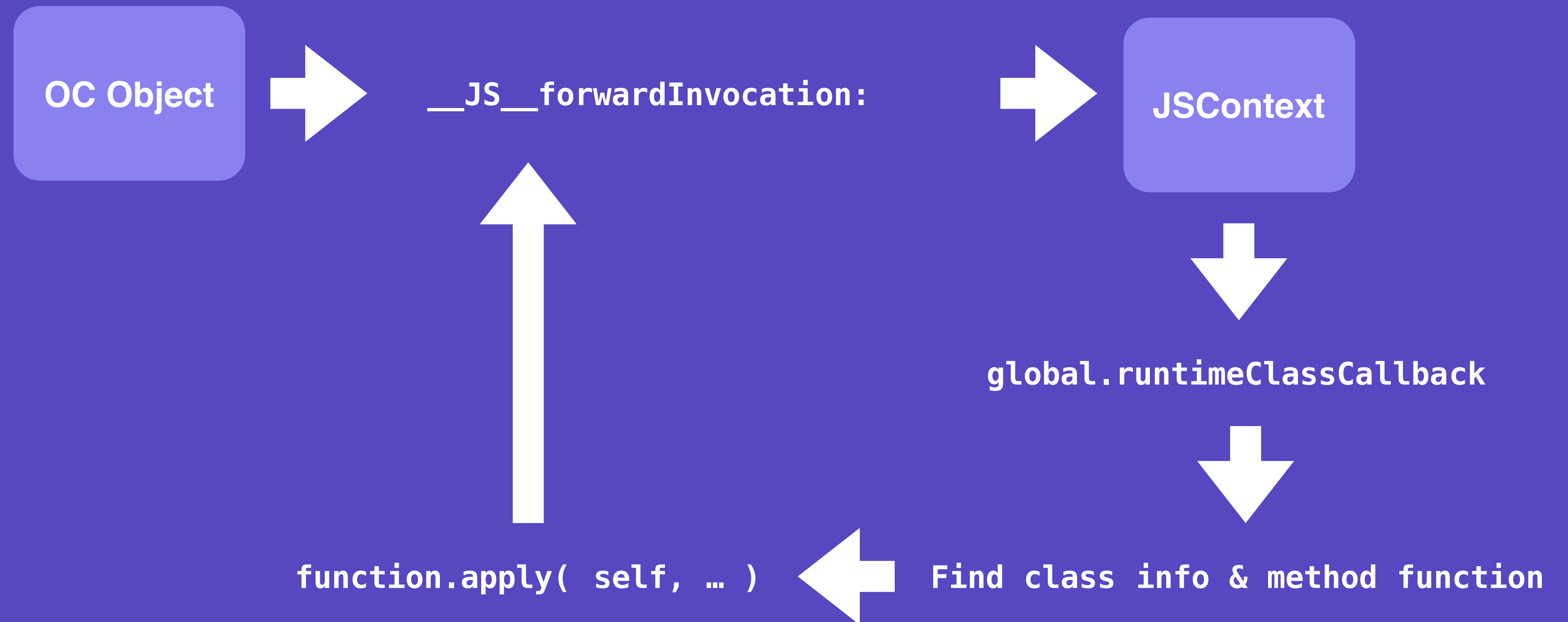
            return returnValue;
        }
    }

    var unboxedArgs = __isNone( methodArgs ) ? [] : __unboxObject( methodArgs );
    var returnValue = __OC__invokeMethod( this._object, methodName, unboxedArgs );
    if ( __isNone( returnValue ) )
    {
        return null;
    }

    return __boxObject( returnValue );
};
```

从 OC 到 JS 调用 (1)

NSObject



从 OC 到 JS 调用 (2)

```
global.runtimeClassCallback = function( className, method, object, args )
{
    var classInfo = global.importedClasses[className];
    if ( __isNone( classInfo ) )
    {
        console.log( "Class '" + className + "' not found" );
        return null;
    }

    var methodForward = classInfo.methodForwards[method];
    if ( methodForward )
    {
        method = methodForward;
    }

    var methodFunction = classInfo.instanceMethods[method];
    if ( __isNone( methodFunction ) )
    {
        console.log( "Method '" + method + "' not found" );
        return null;
    }

    var boxedThis = __boxObject( object );
    var boxedArgs = __boxObject( args );

    var _self = global.self;
    global.self = boxedThis;

    var returnValue = methodFunction.apply( boxedThis, boxedArgs );
    global.self = _self;

    if ( __isNone( returnValue ) )
        return null;

    return __unboxObject( returnValue );
};
```


JSContext 实现

- Get/Create Class
- Invoke Method
- Get/Set Property
- Get/Set Variable
- (在OC中, Class也是Object)

```
self.context[@"__OC__getClass"] = ^( JSValue * className ) {};  
self.context[@"__OC__createClass"] = ^( JSValue * className, JSValue *  
self.context[@"__OC__invokeMethod"] = ^( JSValue * jsObject, JSValue *  
self.context[@"__OC__getProperty"] = ^( JSValue * object, JSValue *  
self.context[@"__OC__setProperty"] = ^( JSValue * object, JSValue *  
self.context[@"__OC__getVariable"] = ^( JSValue * object, JSValue *  
self.context[@"__OC__setVariable"] = ^( JSValue * object, JSValue *  
self.context[@"__OC__retain"] = ^( JSValue * object ) {};  
self.context[@"__OC__release"] = ^( JSValue * object ) {};  
self.context[@"__OC__include"] = ^( JSValue * jsPath ) {};  
self.context[@"__OC__assert"] = ^( JSValue * value, JSValue * desc )  
self.context[@"__OC__log"] = ^( JSValue * text ) {};
```


JSRuntime 实现

- BoxedObject
- BoxedClass
- __boxObject(any)
- __unboxObject(any)
- global.runtimeClassCallback
- global.runtimeFunctionCallback

```
function BoxedObject( object, className ) {};  
  
BoxedObject.prototype.invoke = function( methodName, methodArgs ) {};  
BoxedObject.prototype.retain = function() {}  
BoxedObject.prototype.release = function() {}  
  
function BoxedClass( object, className )  
  
BoxedClass.prototype.invoke = function( methodName, methodArgs ) {};
```

从JS中创建OC类

- OC
 - `__OC__createClass`
- JS
 - `BoxedClass`

```
require( 'MBProgressHUD' );

defineClass( 'MyClass', 'SuperClass', ['Protocol'],
{
    hello : null
},
{
    'method' : function()
    {
        self.props.hello = 'world';
        self.invoke( 'method2:', ["abc"] );
    },

    'method2' : function()
    {
    },
},
));
```

CocoaKit

CocoaKit 作用

App Bundle (/www)

CocoaScript

UIComponents

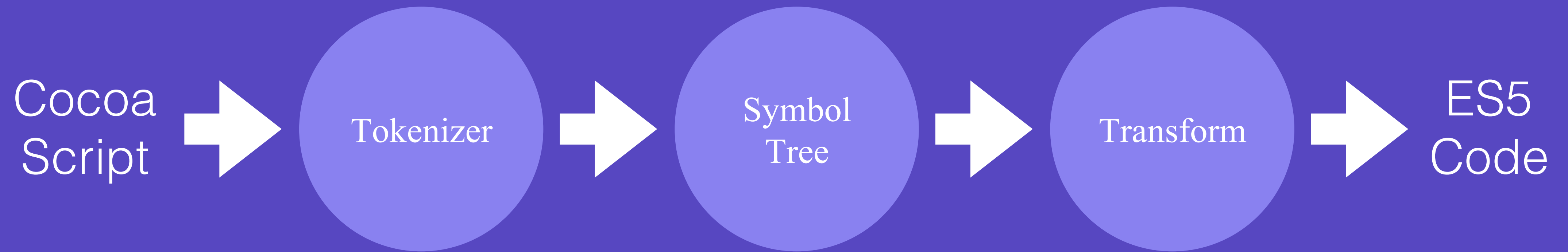
提供高阶的App Bundle支持，及CocoaScript编程语言支持

CocoaKit

- CocoaScript Transformer
 - 运行时将 Obj-C 代码转换为 可运行的JS 代码
 - Sketch插件官方御用语言
 - 原作者是 ccgus (FMDB) ， 对原版进行升级改造
- 支持数据绑定
- 支持更多UIKit组件



CocoaScript Transformer (1)



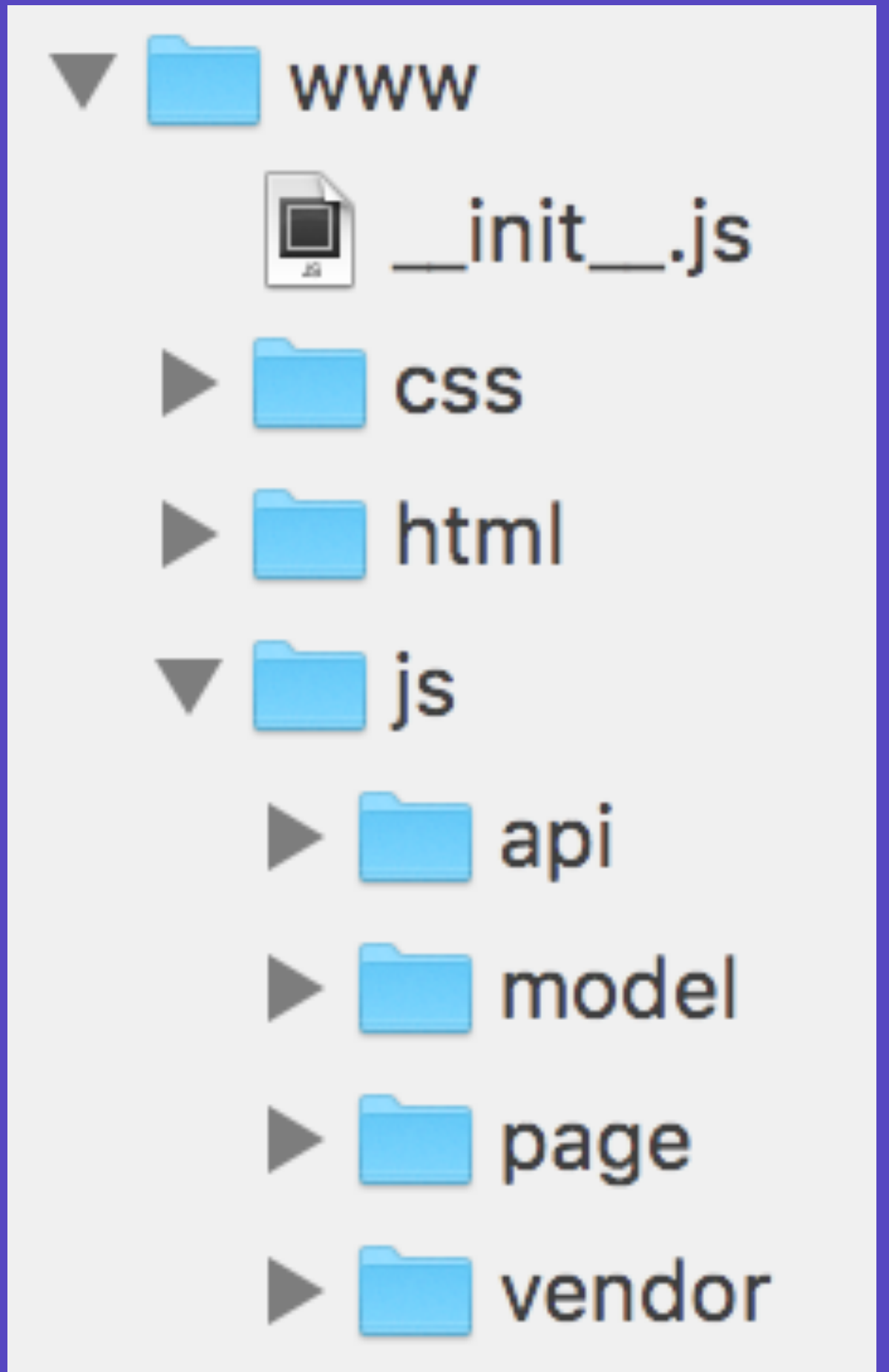
`[obj doSth:arg]` ➔ `obj.invoke('doSth', [arg])`

CocoaScript Transformer (2)

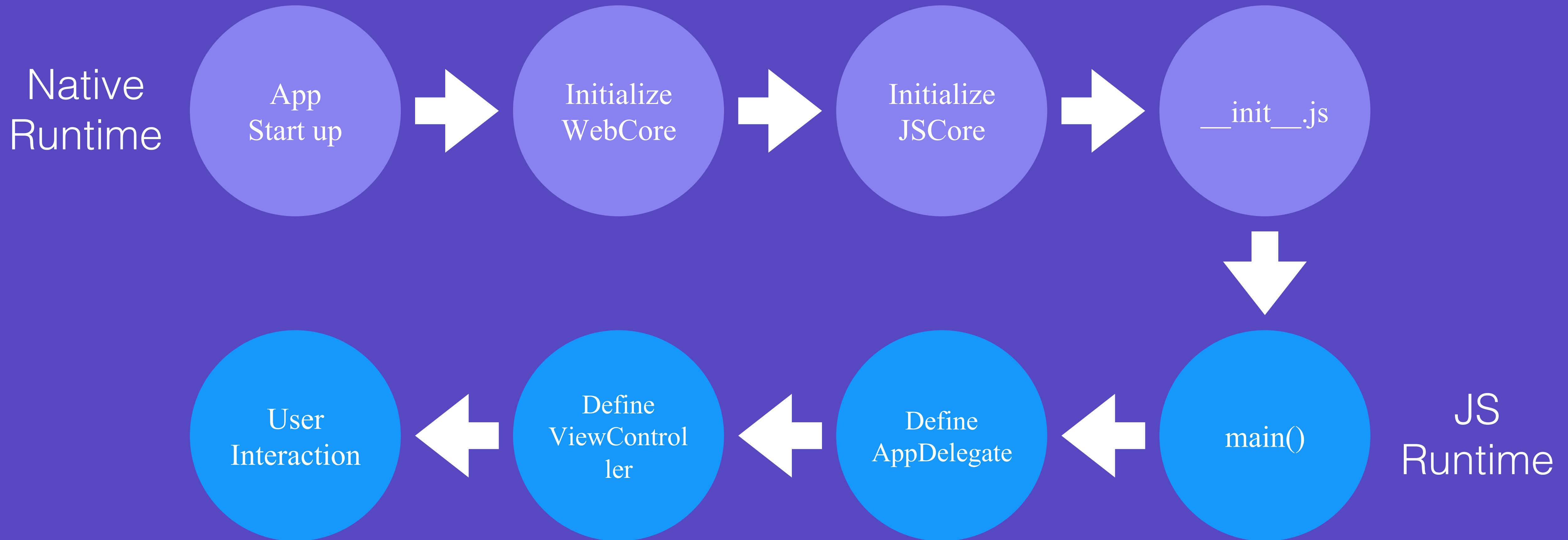
- Transform 规则
 - `[a b] -> a.invoke('b')`
 - `[a b:c] -> a.invoke('b:', c)`
 - `a[i] -> a[i]`
 - `[[a.b.c() d:[e f] g:h i:j.k().l([m.n().o p])] q:([r.s]) t:u[1] v:w(1+2)]`
- `TODParseKit`

App Bundle (1)

- \www
 - `__init__.js` main entry
- \css style files here
- \html template files here
- \js javascript files here



App Bundle (2)



App Bundle (2)

```
int main( int argc, char * argv[] )
{
    SAMURAI_SET_BUNDLE_PATH( "/www" );

    [[SamuraiCocoaKit sharedInstance] startup];

    @autoreleasepool
    {
        return UIApplicationMain( argc, argv, nil, @"DribbbleApp" );
    }
}
```

数据绑定

```
'reloadData' : function()
{
    var model = self.props.model;
    var shots = [];

    for ( var i = 0; i < model.shots.length; ++i )
    {
        var shot = model.shots[i];

        shots.push({
            photo : shot.images.normal,
            avatar : shot.user.avatar_url,
            author : shot.user.name,
            attr : {
                views : shot.views_count,
                likes : shot.likes_count,
                comments : shot.comments_count
            },
        });
    }

    [self.props.list cocoa_setData:{ "shots" : shots }];
},
```

```
<UICollectionViewCell model="shots" onclick="@shotDidClic
width: 100%;
height: auto;
">
<div style="
width: 100%;
height: auto;
margin: 2px;
padding: 2px;
border: solid 1px #d0d0d0;
background-color: #ffffff;
">
<SDWebImageView model="photo" style="
width: 100%;
height: 140px;
background-color: #aaa;
-uikit-contentMode: UIViewContentModeScaleAsP
-uikit-layer-masksToBounds: YES;
">
</SDWebImageView>
```

[view cocoa_setData:{model: value}];

事件转发、处理

Signal handling (1)

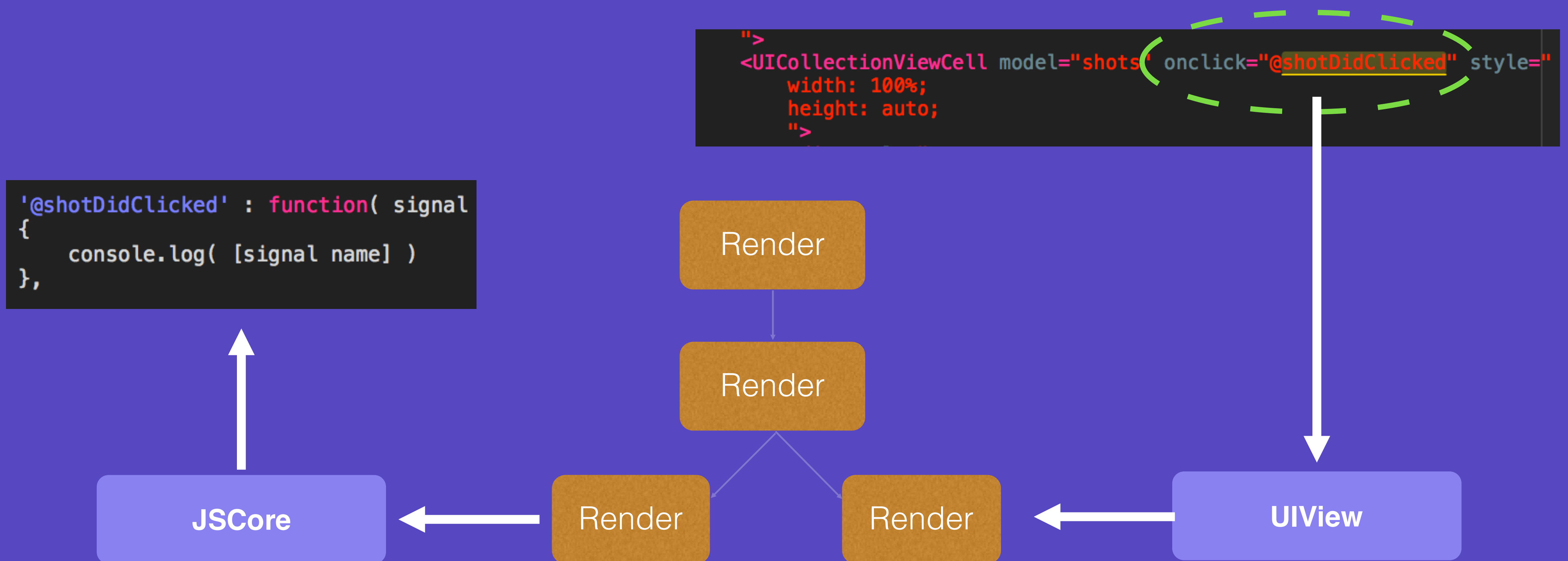
```
'@shotDidClicked' : function( signal )
{
    console.log( [signal name] )
},

'@RefreshCollectionView.eventPullToRefresh' : function( signal )
{
    [self refresh];
},

'@RefreshCollectionView.eventLoadMore' : function( signal )
{
    [self loadMore];
}
```

@ {ClassName} . {EventName}

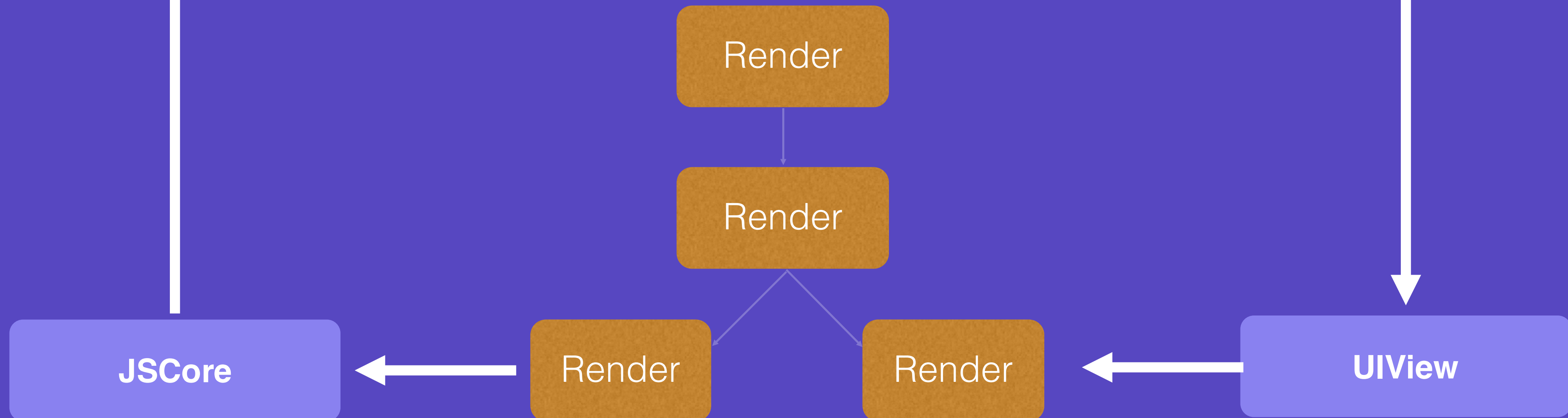
Signal handling (2)



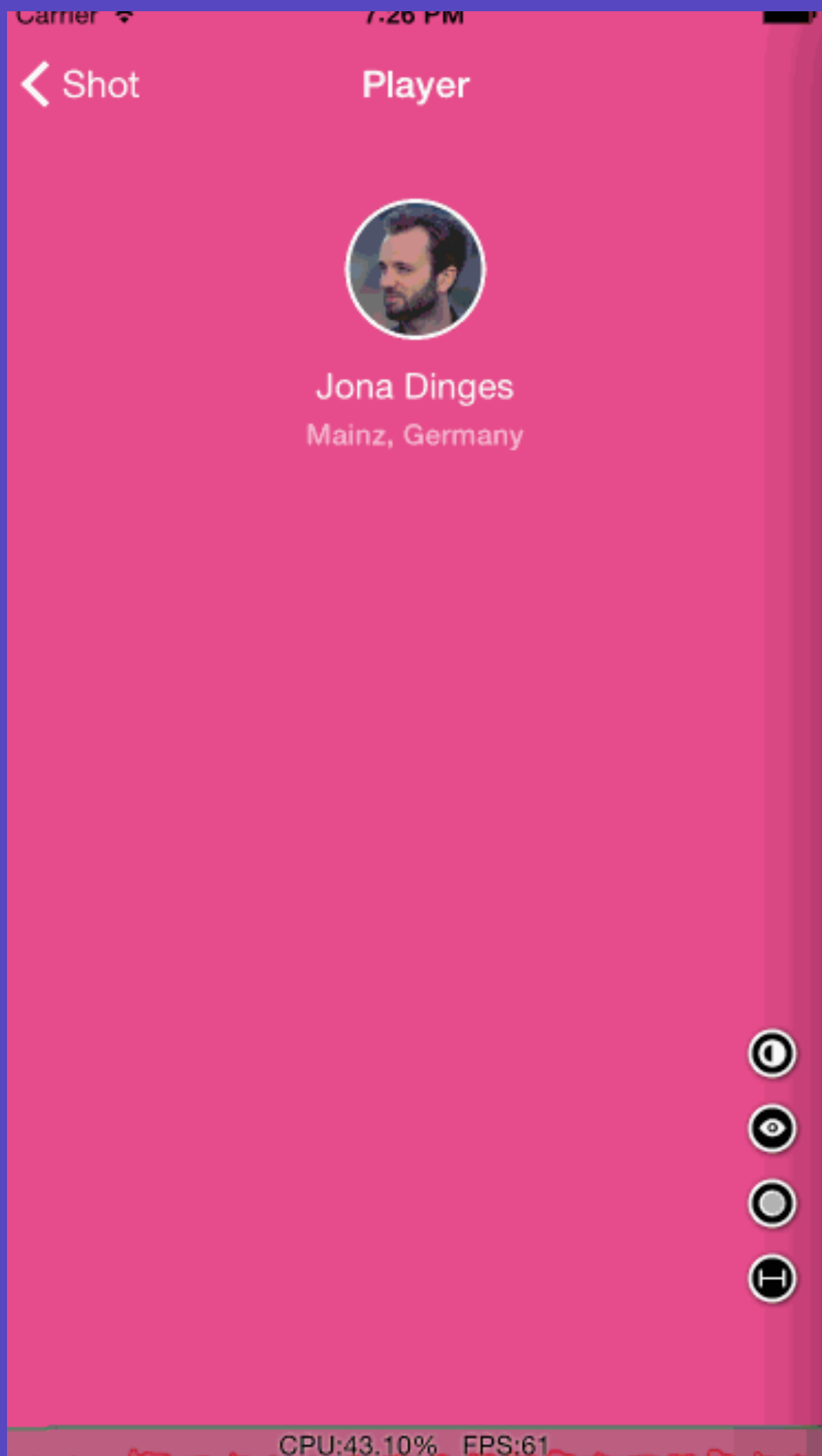
Signal handling (3)

```
@RefreshCollectionView.eventPullToRefresh' : function( signal )  
[  
  [self refresh];  
],
```

```
[self ins_addPullToRefreshWithHeight:60.0 handler:^( UIScrollView  
@strongify(self):  
[self sendSignal:self.eventPullToRefresh];  
}];
```



Liveload



```
demo > demo > www > html > dribbble-player.html > <x-collection id="list" name="player" clas:
18
19 <link rel="stylesheet" type="text/css" href="RefreshCollectionView.css"/>
20 <link rel="stylesheet" type="text/css" href="RefreshTableView.css"/>
21 <link rel="stylesheet" type="text/css" href="WebImage.css"/>
22
23 </head>
24
25 <body class="wrapper fill">
26
27 <x-collection id="list" name="player" class="fill" columns="3" is-vertical>
28
29 <x-collection-cell name="author" is-static is-row>
30 <div class="author-wrapper">
31 <img class="author-avatar" name="avatar"/>
32 <div class="author-name" name="name">Vadim Sherbakov</div>
33 <div class="author-location" name="location">San Francisco, CA</div>
34 </div>
35 </x-collection-cell>
36
37 </x-collection>
38
39 <style>
40
41 .wrapper {
42 background-color: #e5508c;
43 }
44
45 .author-wrapper {
46 display: block;
47 width: 100%;
48 height: 170px;
49 background-color: #e5508c;
50 }
51
52 .author-avatar {
53 display: block;
54 margin-top: 30px;
55 margin-left: auto;
56 margin-right: auto;
57 width: 64px;
58 height: 64px;
59 border-radius: 34px;
60 border-width: 2px;
61 border-color: white;
62 background-color: #999;
```

Samurai-Native 0.2

Open source timeline

- 0.1) Apr, 2015
 - First commit, Open source on github.com
 - <https://github.com/hackers-painters/samurai-native>
- 0.1) Jul, 2015 (Internal)
 - ACID1 pass
 - ACID2 50%
- 0.2) Aug, 2016 (External)
 - Open Source

Thanks

Geek Zoo Studio @ GMTC 2016, Beijing

Be Hungry, Be Foolish

Geek Zoo Studio @ GMTC 2016, Beijing