QINIU

大道至简

React Native在直播应用中的实践

卜赫

buhe@qiniu.com | 2016.5.21

React Native 简介

什么是直播

一个直播应用需要多大开发量?

简单代码下的细节

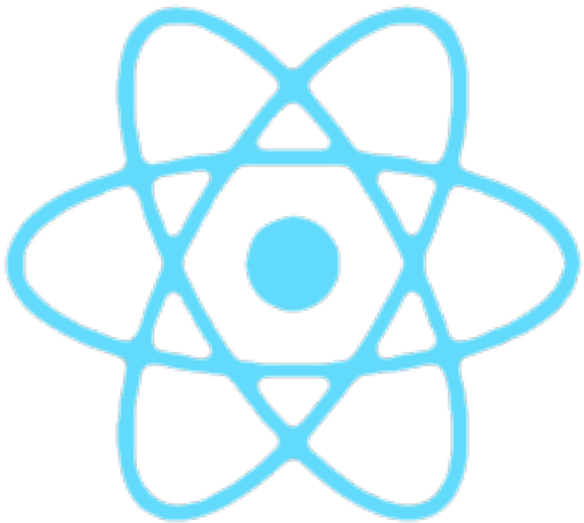
API 设计

统一 Layout

如何优雅的获取和释放硬件资源

React Native 版本升级辄止

ReactJS 语法



React

NPM Base

和PhoneGap等框架的区别

RN的优势：

用原生组件渲染，而不是webview

响应速度不像是webapp，而是native app

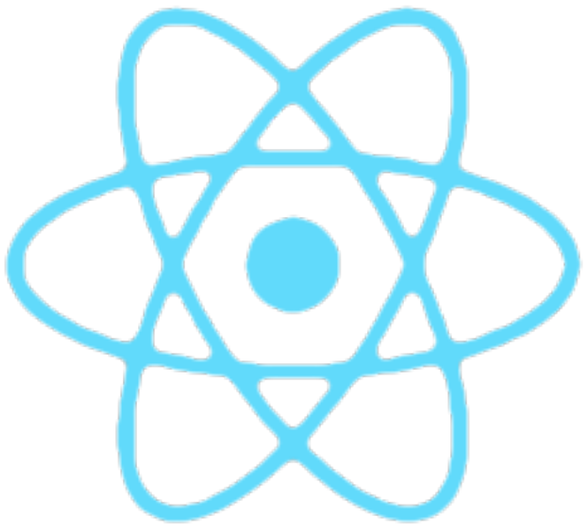
动画不是css模拟而是原生动画

PhoneGap的优势：

用你想用的所有前端库

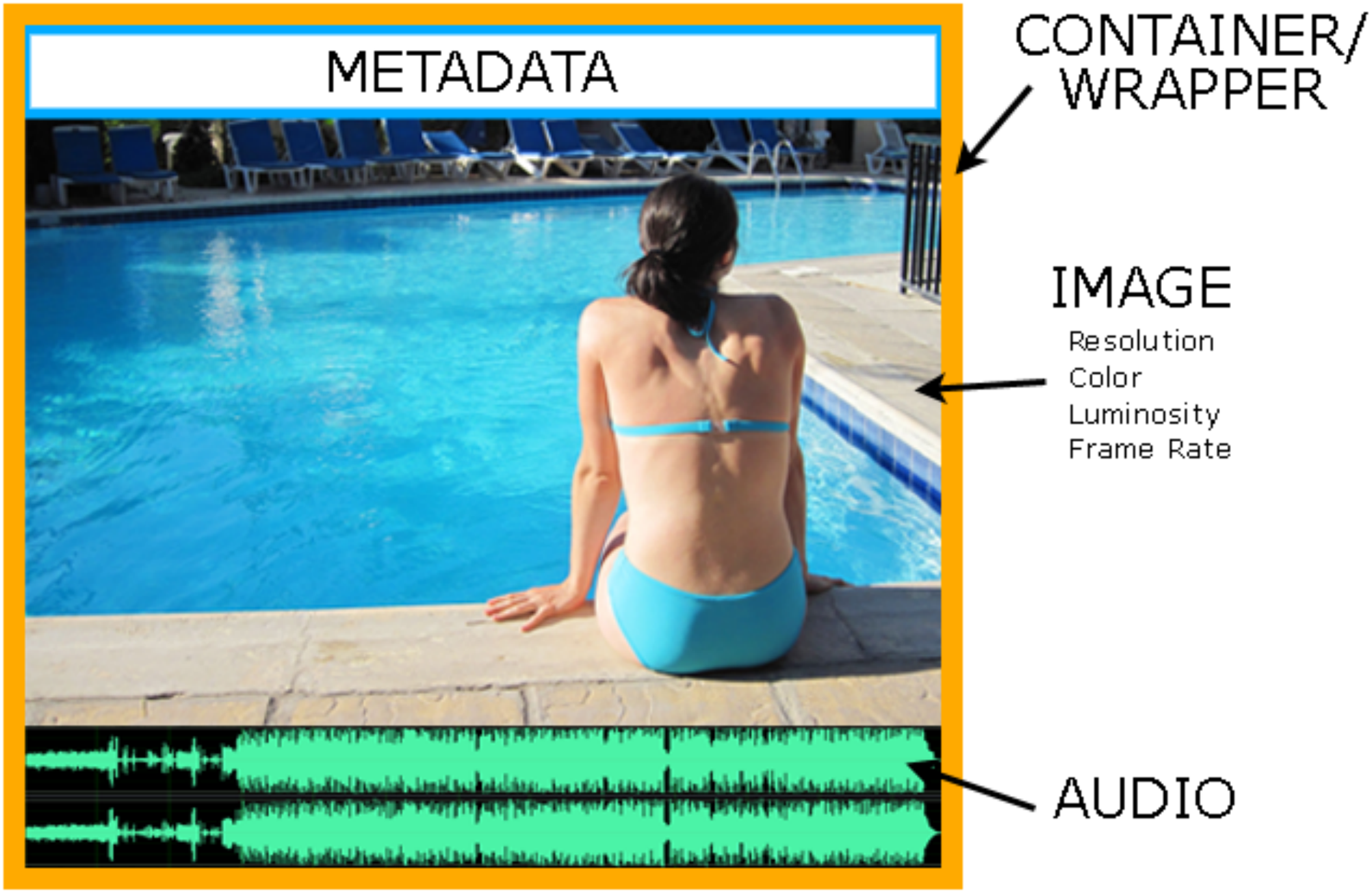
跨平台优势明显 *

View 为中心



React

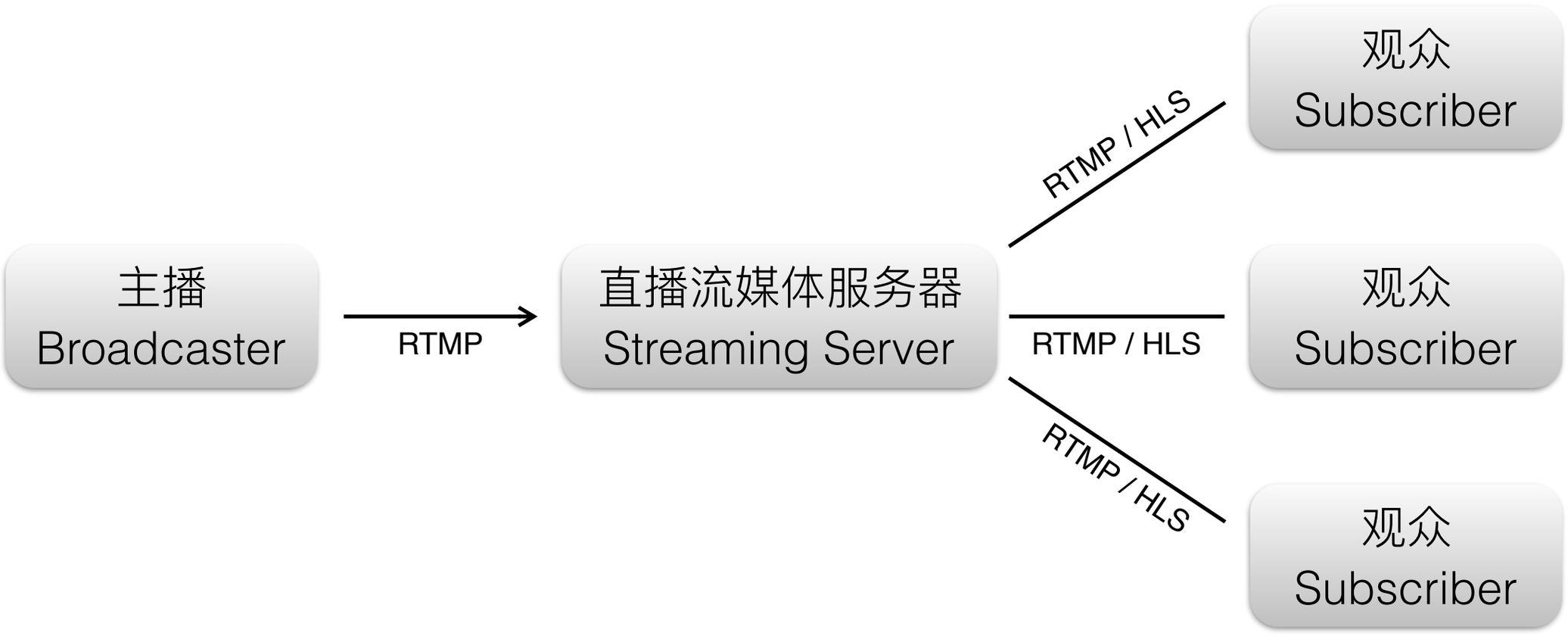
理性认知视频


METADATA
CONTAINER/
WRAPPER
IMAGE
Resolution
Color
Luminosity
Frame Rate
AUDIO

直播模型


主播
Broadcaster
直播流媒体服务器
Streaming Server
观众
Subscriber
观众
Subscriber
观众
Subscriber

直播协议


主播
Broadcaster
RTMP
直播流媒体服务器
Streaming Server
RTMP / HLS
RTMP / HLS
RTMP / HLS
观众
Subscriber
观众
Subscriber
观众
Subscriber

协议差异

	全称	协议	原理	延时
RTMP	Real Time Messaging Protocol	长连接 TCP	每个时刻的数据，收到后立刻转发	1~3 秒
HLS	HTTP Live Streaming	短连接 HTTP	集合一段时间数据，生成 ts 切片文件，更新m3u8	> 10 秒
HTTP-FLV	RTMP over HTTP	长连接 HTTP	同RTMP，使用HTTP协议	1~3 秒

RTMP vs. HLS

	优点	缺点	适用场景
RTMP HTTP-FLV	低延时	跨平台差 Flash Player 以外的平台都需要做移植	即时，有互动需求
HLS	跨平台 可点播回放	高延时 多次请求，网络质量影响大	单向广播

一个直播应用需要多大开发量?

现场看一下所有的代码

```
<Streaming
    stream={{
        id:"xxx", //pili id
        title:"title", //pili title
```

```
<Player
  source={{
    uri:"rtmp://pili-live-rtmp.pilitest.qiniucdn.com/pilitest/xxx",
    controller: true, //Controller ui  Android only
    timeout: 10 * 1000, //live streaming timeout (ms) Android only
    live:true, //live streaming ? Android only
    hardCodec:false, //hard codec [recommended false]  Android only
    }}
  started={true} //iOS only
  muted={false} //iOS only
  style={{
    height:200,
    width:200,
  }}
  onLoading={()=>{}} //loading from remote or local
  onPaused={()=>{}} //pause event
  onShutdown={()=>{}} //stopped event
  onError={()=>{}} //error event
  onPlaying={()=>{}} //play event
  />
```

```
onShutdown={()->{}} //onShutdown event
onIOError={()->{}} //onIOError event
onDisconnected={()->{}} //onDisconnected event
/>
```

第三行

没有了..

现场看一下所有的代码

Streaming

```
<Streaming
    stream={{
        id:"xxx", //pili id
        title:"title", //pili title
        hub:"hubname", //pili hub name
        publishKey:"<PK>", //pili key
        publishSecurity:"static", //pili secrity policy (static or dynamic)
        hosts:{
          publish:{ //pili Streaming url (support rtmp)
            rtmp:"pili-publish.pilitest.qiniucdn.com"
          }
        }
        }}
    style={{
        height:400,
        width:400,
    }}
    zoom={1} //zoom
    muted={true} //muted
    focus={false} //focus
    profile={{  //video and audio profile
       video:{
         fps:30,
         bps:1000 * 1024,
         maxFrameInterval:48
       },
       audio:{
         rate:44100,
         bitrate:96 * 1024
       },
    started={false} //streaming status
    onReady={()->{}} //onReady event
    onConnecting={()->{}} //onConnecting event
    onStreaming={()->{}} //onStreaming event
    onShutdown={()->{}} //onShutdown event
    onIOError={()->{}} //onIOError event
    onDisconnected={()->{}} //onDisconnected event
    />
```

Player

```
<Player
  source={{
  uri:"rtmp://pili-live-rtmp.pilitest.qiniucdn.com/pilitest/xxx",
  controller: true, //Controller ui  Android only
  timeout: 10 * 1000, //live streaming timeout (ms) Android only
  live:true, //live streaming ? Android only
  hardCodec:false, //hard codec [recommended false]  Android only
  }}
  started={true} //iOS only
  muted={false} //iOS only
  style={{
    height:200,
    width:200,
  }}
  onLoading={()=>{}} //loading from remote or local
  onPaused={()=>{}} //pause event
  onShutdown={()=>{}} //stopped event
  onError={()=>{}} //error event
  onPlaying={()=>{}} //play event
  />
```

上面的那些代码隐藏了哪些细节?

Native Component UI

没办法用Pure JS 做编解码

没办法用Pure JS 实现RTMP

没办法用Pure JS 操作硬件

关键性能问题

JS + Obj-c + Java

Streaming = **JS**

PreviewView + **Native Code**

StreamingManger + **Native Code**

Encode **Native Code**

JS

Player =

Native Code

PlayerView +

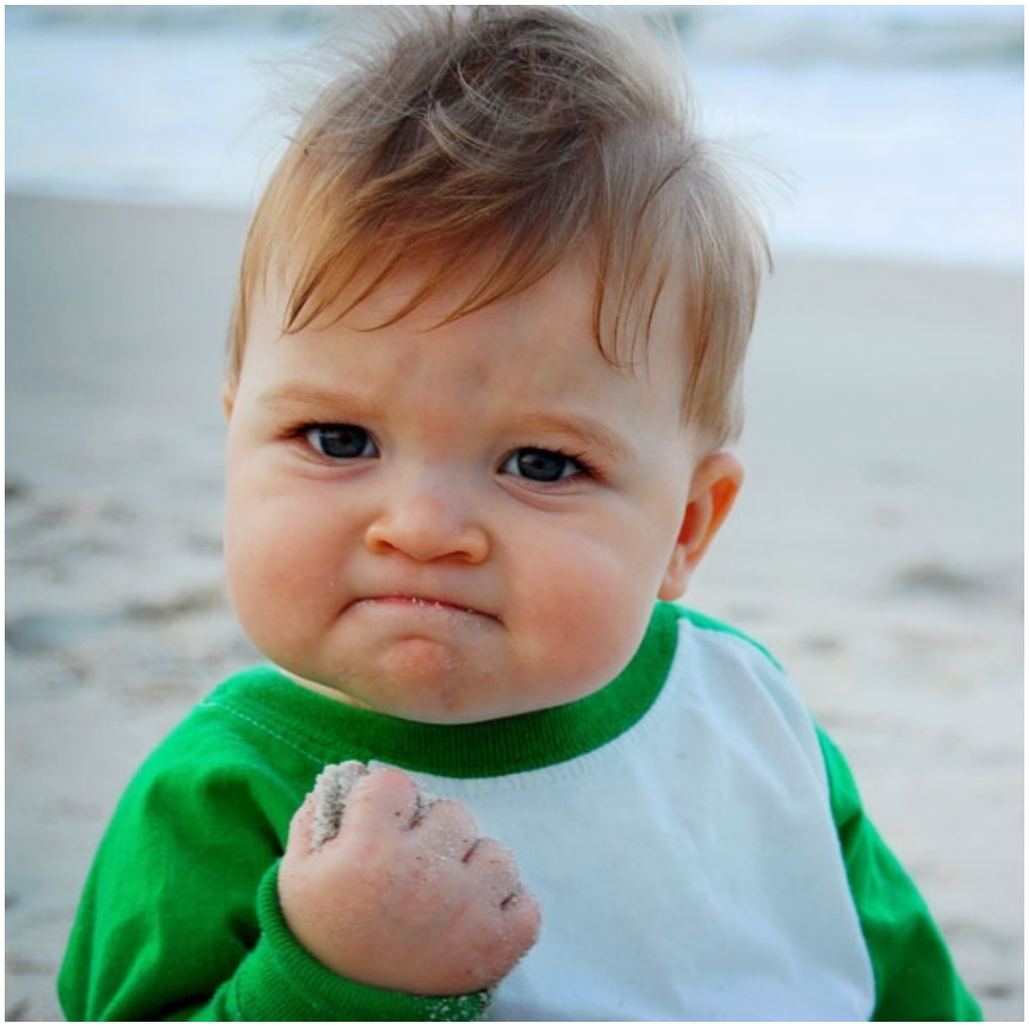
Native Code

PlayStreamingManger +

Native Code

Decode

QINIU



不管有多少层次页面，和状态管理，React Native 都把他们隐藏在细节里面了，最终返回给开发者的只是一个最终页面和够用的属性。

即：Preview View 和 Player View

API 设计

RN API 表达优势

API的设计风格统一

访问API的方式统一

对API设计功力更有挑战

Name	Value
width, height	positive number
minWidth, minHeight	positive number
maxWidth, maxHeight	positive number
left, right, top, bottom	number
margin, marginLeft, marginRight, marginTop, marginBottom	number
padding, paddingLeft, paddingRight, paddingTop, paddingBottom	positive number
borderWidth, borderLeftWidth, borderRightWidth, borderTopWidth, borderBottomWidth	positive number
flexDirection	'column', 'row'
justifyContent	'flex-start', 'center', 'flex-end', 'space-between', 'space-around'
alignItems, alignSelf	'flex-start', 'center', 'flex-end', 'stretch'
flex	positive number
flexWrap	'wrap', 'nowrap'
position	'relative', 'absolute'

配置管理

```
<Streaming
    stream={{
        id:"xxx", //pili id
        title:"title", //pili title
        hub:"hubname", //pili hub name
        publishKey:"<PK>", //pili key
        publishSecurity:"static", //pili secrity policy (static or dynamic)
        hosts:{
          publish:{ //pili Streaming url (support rtmp)
            rtmp:"pili-publish.pilitest.qiniucdn.com"
          }
        }
        }}
    style={{
        height:400,
        width:400,
    }}
    zoom={1} //zoom
    muted={true} //muted
    focus={false} //focus
    profile={{  //video and audio profile
       video:{
         fps:30,
         bps:1000 * 1024,
         maxFrameInterval:48
       },
       audio:{
         rate:44100,
         bitrate:96 * 1024
       },
    started={false} //streaming status
    onReady={()->{}} //onReady event
    onConnecting={()->{}} //onConnecting event
    onStreaming={()->{}} //onStreaming event
    onShutdown={()->{}} //onShutdown event
    onIOError={()->{}} //onIOError event
    onDisconnected={()->{}} //onDisconnected event
    />
```

状态管理

```
<Streaming
    stream={{
        id:"xxx", //pili id
        title:"title", //pili title
        hub:"hubname", //pili hub name
        publishKey:"<PK>", //pili key
        publishSecurity:"static", //pili secrity policy (static or dynamic)
        hosts:{
          publish:{ //pili Streaming url (support rtmp)
            rtmp:"pili-publish.pilitest.qiniucdn.com"
          }
        }
        }}
    style={{
        height:400,
        width:400,
    }}
    zoom={1} //zoom
    muted={true} //muted
    focus={false} //focus
    profile={{  //video and audio profile
       video:{
         fps:30,
         bps:1000 * 1024,
         maxFrameInterval:48
       },
       audio:{
         rate:44100,
         bitrate:96 * 1024
       },
    started={false} //streaming status
    onReady={()->{}} //onReady event
    onConnecting={()->{}} //onConnecting event
    onStreaming={()->{}} //onStreaming event
    onShutdown={()->{}} //onShutdown event
    onIOError={()->{}} //onIOError event
    onDisconnected={()->{}} //onDisconnected event
    />
```

动作管理

```
<Streaming
    stream={{
        id:"xxx", //pili id
        title:"title", //pili title
        hub:"hubname", //pili hub name
        publishKey:"<PK>", //pili key
        publishSecurity:"static", //pili secrity policy (static or dynamic)
        hosts:{
          publish:{ //pili Streaming url (support rtmp)
            rtmp:"pili-publish.pilitest.qiniucdn.com"
          }
        }
        }}
    style={{
        height:400,
        width:400,
    }}
    zoom={1} //zoom
    muted={true} //muted
    focus={false} //focus
    profile={{  //video and audio profile
       video:{
         fps:30,
         bps:1000 * 1024,
         maxFrameInterval:48
       },
       audio:{
         rate:44100,
         bitrate:96 * 1024
       },
    started={false} //streaming status
    onReady={()->{}} //onReady event
    onConnecting={()->{}} //onConnecting event
    onStreaming={()->{}} //onStreaming event
    onShutdown={()->{}} //onShutdown event
    onIOError={()->{}} //onIOError event
    onDisconnected={()->{}} //onDisconnected event
    />
```

事件管理

```
<Streaming
    stream={{
        id:"xxx", //pili id
        title:"title", //pili title
        hub:"hubname", //pili hub name
        publishKey:"<PK>", //pili key
        publishSecurity:"static", //pili secrity policy (static or dynamic)
        hosts:{
          publish:{ //pili Streaming url (support rtmp)
            rtmp:"pili-publish.pilitest.qiniucdn.com"
          }
        }
        }}
    style={{
        height:400,
        width:400,
    }}
    zoom={1} //zoom
    muted={true} //muted
    focus={false} //focus
    profile={{  //video and audio profile
       video:{
         fps:30,
         bps:1000 * 1024,
         maxFrameInterval:48
       },
       audio:{
         rate:44100,
         bitrate:96 * 1024
       },
    started={false} //streaming status
    onReady={()->{}} //onReady event
    onConnecting={()->{}} //onConnecting event
    onStreaming={()->{}} //onStreaming event
    onShutdown={()->{}} //onShutdown event
    onIOError={()->{}} //onIOError event
    onDisconnected={()->{}} //onDisconnected event
    />
```

this.state = {myMuted:true}

<Streaming
muted={this.state.myMuted}
/>

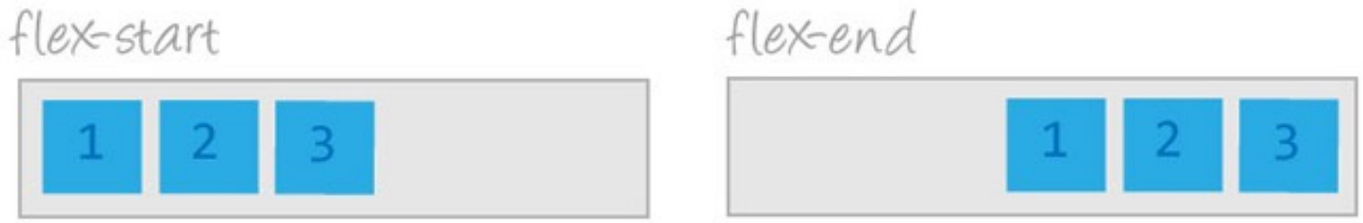
```
<Streaming
    stream={{
        id:"xxx", //pili id
        title:"title", //pili title
        hub:"hubname", //pili hub name
        publishKey:"<PK>", //pili key
        publishSecurity:"static", //pili security policy (static or dynamic)
        hosts:{
          publish:{ //pili Streaming u    upport rtmp)
            rtmp:"pili-publish.pilitest.qiniucdn.com"
          }
        }
        }}
    style={{
        height:400,
        width:400,
    }}
    zoom={1} //zoom
    muted={true} //muted
    focus={false} //focus
    profile={{  //video and audio profile
       video:{
         fps:30,
         bps:1000 * 1024,
         maxFrameInterval:48
       },
       audio:{
         rate:44100,
         bitrate:96 * 1024
       },
    started={false} //streaming status
    onReady={()->{}} //onReady event
    onConnecting={()->{}} //onConnecting event
    onStreaming={()->{}} //onStreaming event
    onShutdown={()->{}} //onShutdown event
    onIOError={()->{}} //onIOError event
    onDisconnected={()->{}} //onDisconnected event
    />
```

this.setState({myMuted:false})

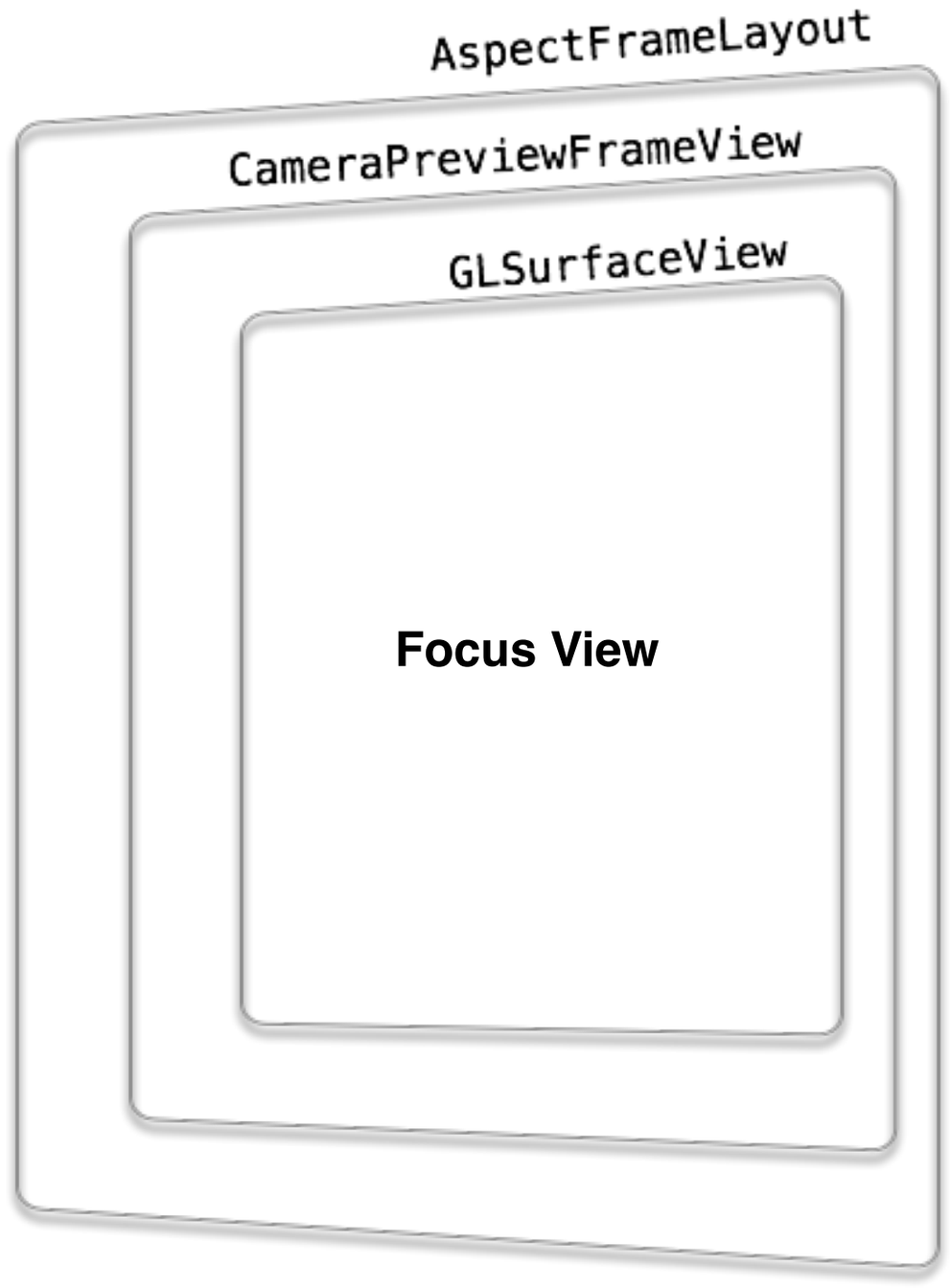
iOS,Android,RN 布局各有不同,在 React Native Component UI 中怎么适配?

RN Layout

Flex Box



Streaming View Hierarchy



Android Layout

Android 一般使用xml Layout

在RN中如果混入xml 让用户配置肯定不合适，可以内部用代码定义布局，最外层的View通过RN 的FlexBox来布局

RN 直接调用measure 和 layout对View 进行定位和布局

```objc
UIView *playerView = _plplayer.playerView;
[self addSubview:playerView];
 [playerView setTranslatesAutoresizingMaskIntoConstraints:NO];

NSLayoutConstraint *centerX = [NSLayoutConstraint constraintWithItem:playerView attribute:NSLayoutAttributeCenterX relatedBy:
    NSLayoutRelationEqual toItem:self attribute:NSLayoutAttributeCenterX multiplier:1.0 constant:0];
NSLayoutConstraint *centerY = [NSLayoutConstraint constraintWithItem:playerView attribute:NSLayoutAttributeCenterY relatedBy:
    NSLayoutRelationEqual toItem:self attribute:NSLayoutAttributeCenterY multiplier:1.0 constant:0];
NSLayoutConstraint *width = [NSLayoutConstraint constraintWithItem:playerView attribute:NSLayoutAttributeWidth relatedBy:
    NSLayoutRelationEqual toItem:self attribute:NSLayoutAttributeWidth multiplier:1.0 constant:0];
NSLayoutConstraint *height = [NSLayoutConstraint constraintWithItem:playerView attribute:NSLayoutAttributeHeight relatedBy:
    NSLayoutRelationEqual toItem:self attribute:NSLayoutAttributeHeight multiplier:1.0 constant:0];

NSArray *constraints = [NSArray arrayWithObjects:centerX, centerY,width,height, nil];
[self addConstraints: constraints];
```

iOS Layout

同理这里也是不推荐用 StoryBoard 的，一般来讲直接让内部的 View 和最外层暴露给 RN 的 View 保持大小一致就好了

这里也比较简单，添加约束让 SubView 居中，长宽等于 ParentView 就可以了

如何优雅的获取和释放硬件资源

QINIU

什么时候需要获取和释放硬件资源

PiliStreamingViewManager.java

```java
public class PiliStreamingViewManager extends SimpleViewManager<View>
        LifecycleEventListener {
    public View createViewInstance(ThemedReactContext context){
        context.addLifecycleEventListener(this);
        return ...
    }

        @Override
    public void onHostResume() {
        mCameraStreamingManager.resume();
    }

    @Override
    public void onHostPause() {
        mCameraStreamingManager.pause();
    }

    @Override
    public void onHostDestroy() {
        mCameraStreamingManager.destroy();
    }


}
```

React Native 频繁的版本升级怎么应对?

升级 or ~~不升级~~

RN 目前处在高速发展中，导航栏就发布了三个版本。

新的组件在不断的发布。

新的API在持续更新。

性能在变好，包括动画，更少的使用反射(https://github.com/facebook/react-native/pull/6466, https://github.com/facebook/react-native/commit/57f6cbb3dc12d2dcc3eedd9712c36bb3d39149a5)

支持硬件新特性，比如3D touch

升级带来的不适

0.14 开始支持更好的image加载方式，需要手动升级

0.19 移动了Android ReactProp annotation的位置

升级node版本后，删除node_modules 重新install

https://github.com/buhe/react-native-pili

https://github.com/pili-engineering

延伸阅读

https://demos.scotch.io/visual-guide-to-css3-flexbox-flexbox-playground/demos/

https://facebook.github.io/react-native/docs/native-components-android.html#content

https://facebook.github.io/react-native/docs/native-components-ios.html#content

https://github.com/facebook/css-layout

Author

buhe

github.com/buhe

wechat: 81128054

email: buhe@qiniu.com

focus: DCOS,Data,full-stack