TLC 2019 大前端 信息流

腾讯LIVE开发者大会
TENCENT LIVE CONFERENCE

多端一体方案 Hippy 的架构和实战

腾讯 Hippy 开发组 / xqkuang (旷旭卿)

2019.8.16

目录

需求和解决方案


热更新能力
灰度和分阶段更新
上线快
投入少
跨平台运行
一套代码三端运行
需求
低学习成本
支持主流前端框架：React 和 Vue
易上手
体验好
高性能
前终端高性能优化

开发模式

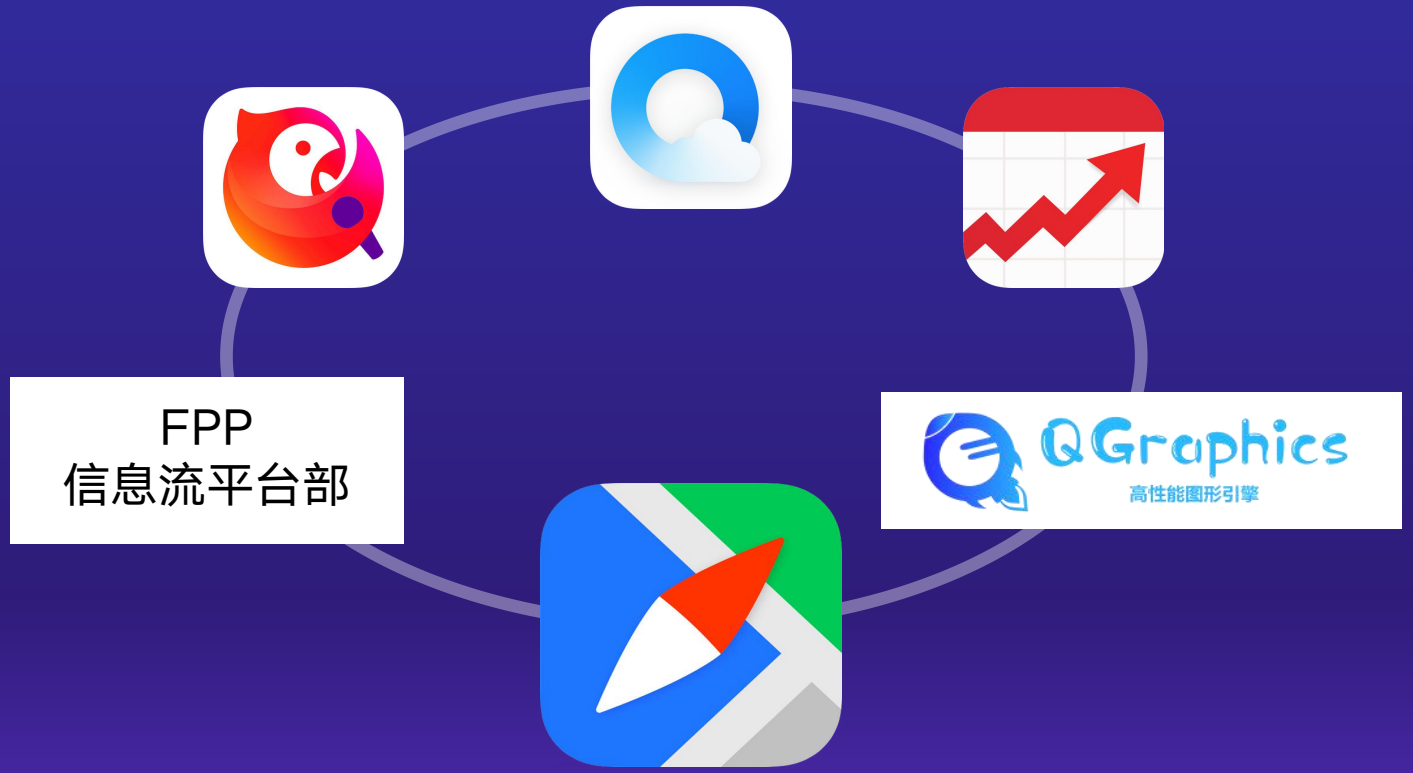
前端开发

- 开发效率高
- 双平台支持
- 动态发布
- 体验相对交差
- 原生交互能力弱

终端原生开发

- 原生支持
- 体验较好
- 多平台开发
- 开发效率低
- 无法动态发布

Hippy 更接近传统 Web 开发



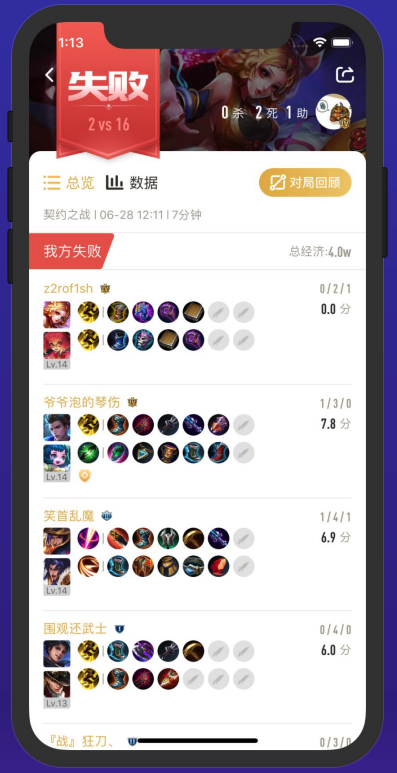
QQ浏览器发起

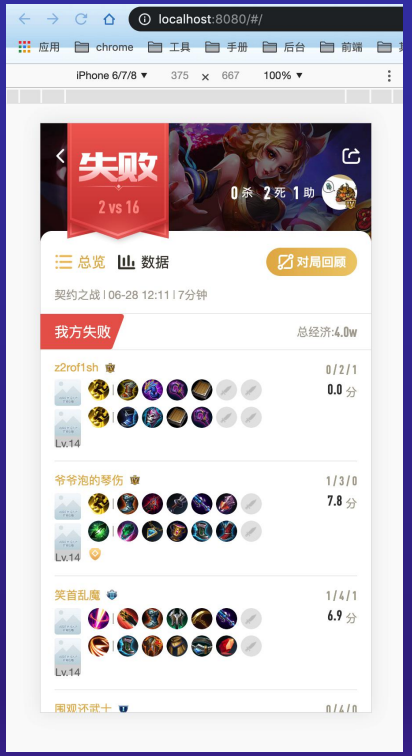
腾讯开源协同

- 开发接近传统浏览器
- 可复用业内 Web 生态
- 前端开发主导，终端实现
- 腾讯六大部门协力开发
- 16个已上线 App，日活过亿

跨 Web 能力

全民 K 歌
React + Hippy-React + Hippy-React-Web





王者营地
Vue + Hippy-Vue

Coding with Hippy

```react
// Hippy React
import React from "react";
import { Hippy, View, Text } from "@tencent/hippy-react";

function HelloWorld() {
  return (
    <View>
      <Text style={{ color: 'blue' }}>Hello World</Text>
    </View>
  );
}

new Hippy({
  appName: "MyHippyReactApp",
  entryPage: HelloWorld
}).regist();
```

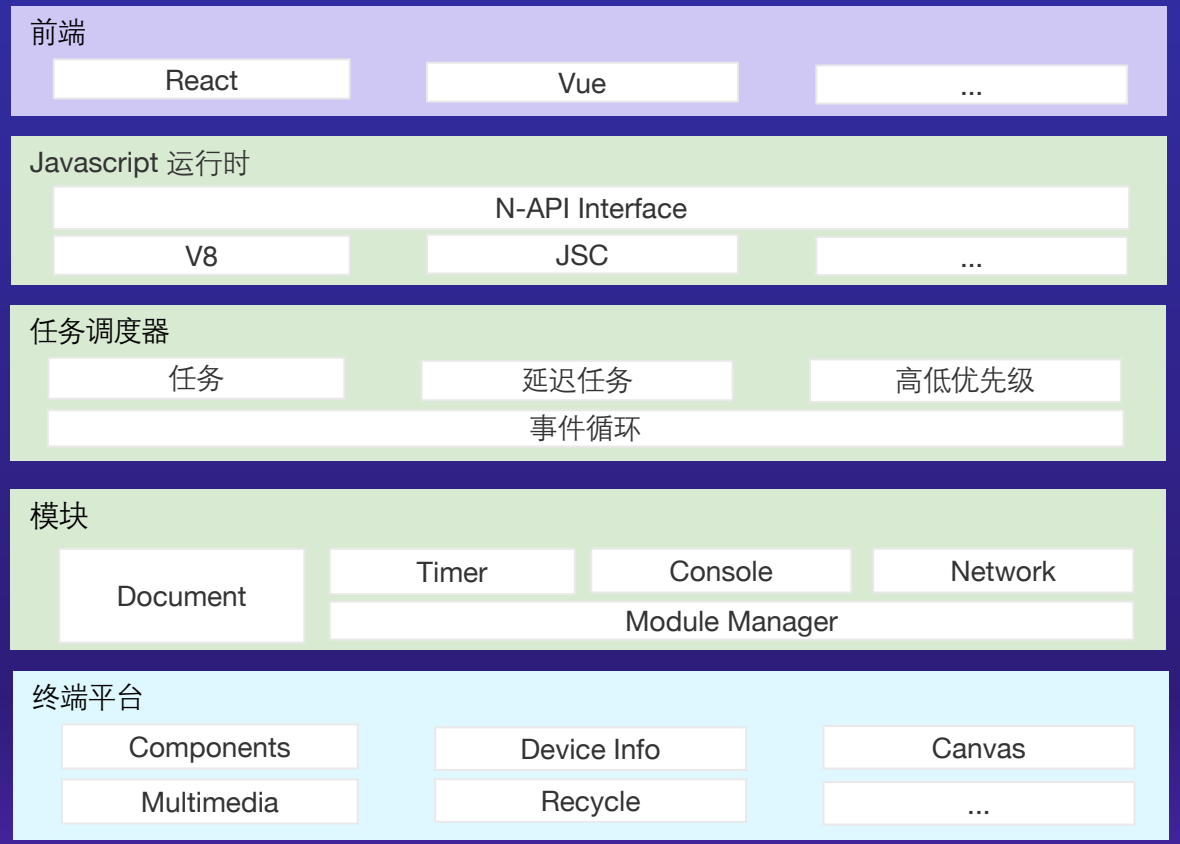
```vue
// Hippy Vue
// hello-world.vue
<template>
  <div id="root">
    <p id="text">Hello world</p>
  </div>
</template>
<style>
  p#text { color: blue }
<style>

// native-main.js
import HippyVue from '@tencent/hippy-vue';
import HippyVueRouter from '@tencent/hippy-vue-router';
import HelloWorld from './hello-world.vue';
import routes from './routes';

HippyVue.use(VueRouter);
const router = new HippyVueRouter(routes);

new HippyVue({
  appName: 'MyHippyVueApp',
  rootView: '#root',
  render: h => h(HelloWorld),
  router,
}).$start();
```

Hippy 整体架构

Javascript

C++

Java/OC



目前已经支持前端主流的框架，
简单扩展即可支持厂内其它框架

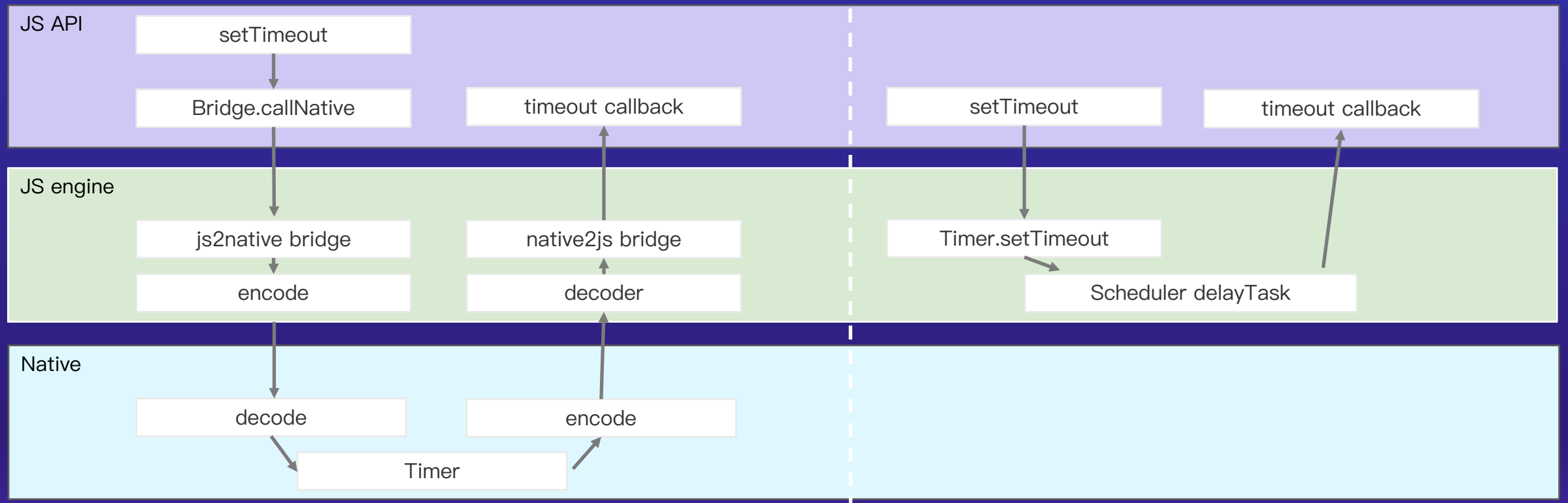
封装 V8、JavaScriptCore 或其它 JS Engine 接口

Hippy Core 任务调度

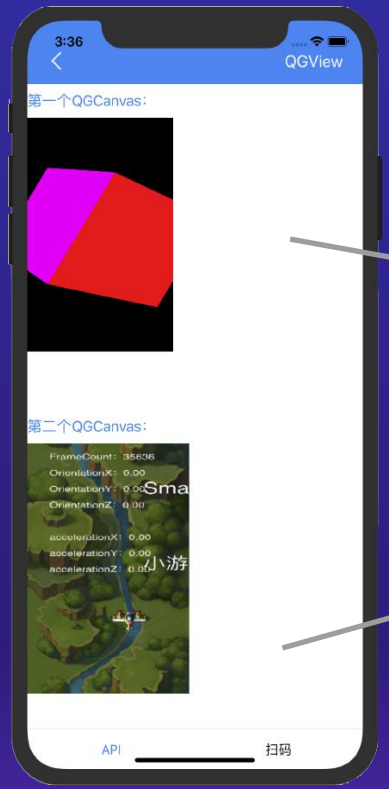
C++ 开发的高性能模块，
通过 Binding 模式进行 JS 和终端的通讯

目前支持 iOS、Android、Web 三端，可以通过 kbone/mpvue 嫁接到小程序

前终端通讯方式


JS API
setTimeout
Bridge.callNative
timeout callback
setTimeout
timeout callback
JS engine
js2native bridge
native2js bridge
Timer.setTimeout
encode
decoder
Scheduler delayTask
Native
decode
encode
Timer
传统 Bridge 模式
新的 Binding 模式

QG - Binding 模式下的高性能绘图

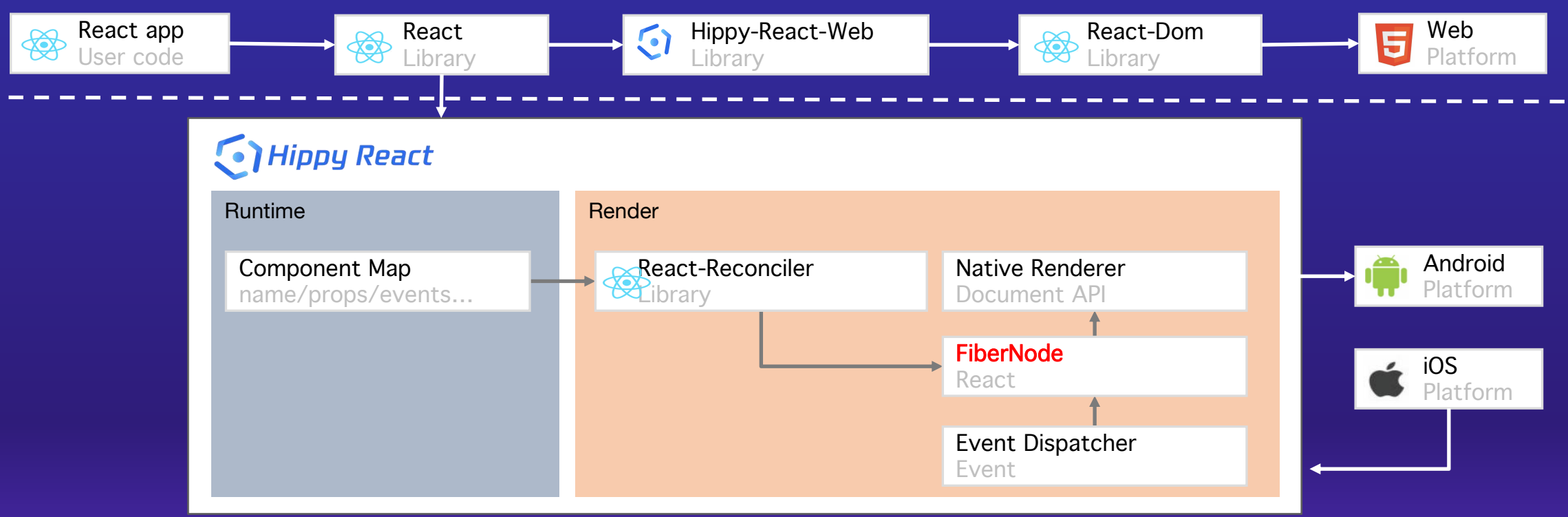


3D Cube

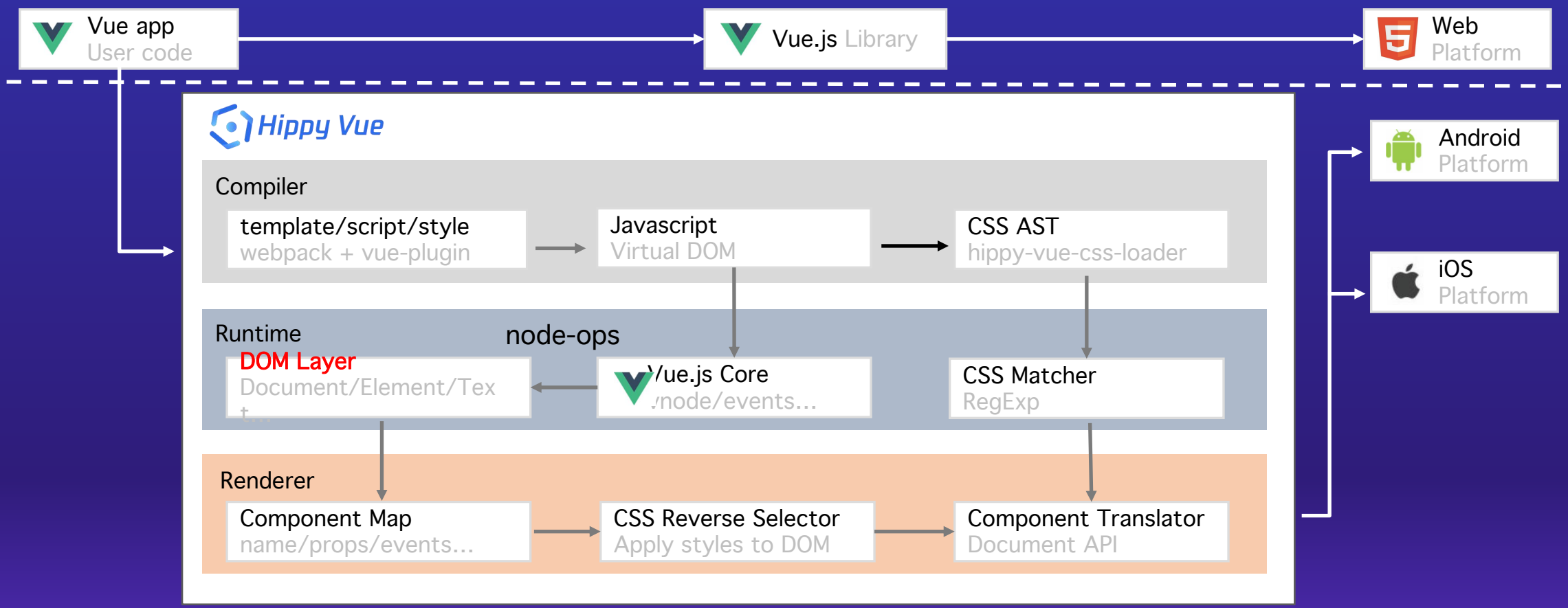
2D
打飞机小游戏

```
render() {
    return (
        <View style={styles.container}>
            <Text style={[styles.normalText, { marginTop: 10 }]}>第一个QGCanvas: </Text>
            <QGView ref='qgview1' viewId={100}
                style={[styles.QGView, { marginTop: 10, width: 180, height: 280 }]}
                drawCanvas={(window, canvas) => drawRotateBox(window, canvas) } />
            <Text style={[styles.normalText, { marginTop: 80 }]}>第二个QGCanvas: </Text>
            <QGView ref='qgview2' viewId={200}
                style={[styles.QGView, { marginTop: 10, width: 200, height: 300 }]}
                drawCanvas={(window, canvas) => this.drawAirplane(window, canvas)} />
        </View>
    );
};
```

Hippy-React 架构



Hippy-Vue 架构


Vue app
User code
Vue.js Library
Web
Platform
Hippy Vue
Compiler
template/script/style
webpack + vue-plugin
Javascript
Virtual DOM
CSS AST
hippy-vue-css-loader
Runtime
node-ops
DOM Layer
Document/Element/Tex
t...
Vue.js Core
/node/events...
CSS Matcher
RegExp
Renderer
Component Map
name/props/events...
CSS Reverse Selector
Apply styles to DOM
Component Translator
Document API
Android
Platform
iOS
Platform

Hippy-Vue 样式解析器

```
#id { color: red }
#id .class { background-color: blue }
```

CSS AST
hippy-vue-css-loader

```
[
  {
    selector: [['#id']],
    style: [{ color: 12345 }],
  },
  {
    selector: [['#id', '.class']],
    style: [{ backgroundColor: 56789 }]
  }
]
```

CSS Matcher
RegExp

```
[
  {
    selector: [[IdSelector('id')]],
    style: [ { color: 12345 } ],
  },
  {
    selector: [[IdSelector('id'), ClassSelector('class')]],
    style: [ { backgroundColor: 56789 } ]
  }
]
```

```
<p>Hello</p>
<p id="id">
  TLC 2019
  <p class="class">Shenzhen</p>
</p>
```

CSS Reverse Selector
Apply styles to DOM

0: Matched [false, false];
1: Matched [true, false];
2: Matched [true, true];

```
Hello
TLC 2019
Shenzhen
```

可复用长列表

灰框部分是

Hippy

实现的长列表

iPhone Xs Max — 12.4

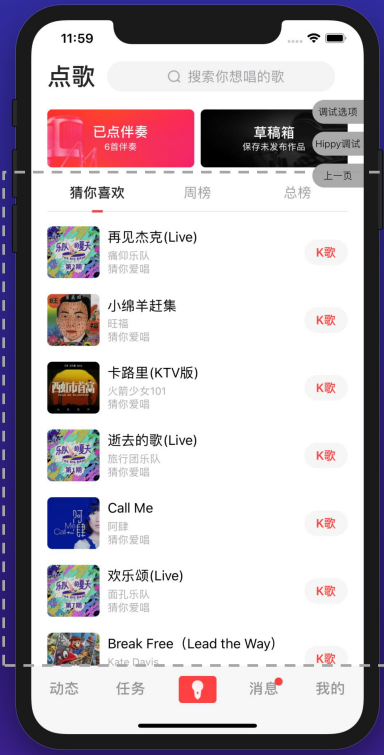
QQ浏览器

iPhone Xs Max — 12.4

腾讯自选股

iPhone Xs Max — 12.4

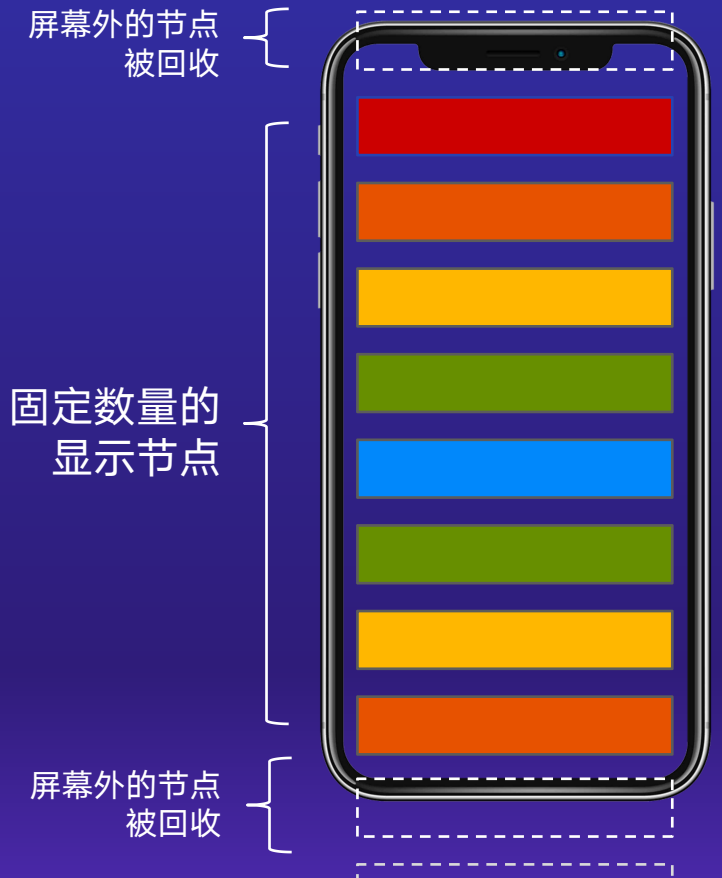
大丰满满



iPhone Xs Max — 12.4

全民K歌极速版

可复用列表能力实现




```
<ul id="list" @scroll="onListScroll" :numberOfRows="data.length">
  <li v-for="ui in data" :key="ui.key" :type="`row-${ui.style}`">
    <style-one v-if="ui.style == 1" :item="ui.item" />
    <style-two v-if="ui.style == 2" :item="ui.item" />
    <style-three v-if="ui.style == 3" :item="ui.item" />
  </li>
</ul>
```

Hippy 的可复用列表对前端而言非常简单，实现主要在终端层，前端只需要提交数据。但有两个参数非常重要：

- numberOfRows：达到数量时上屏
- type：以 Type 为 key 缓存已经渲染的界面，当滚动相同 type 的 li（映射到终端的 ListItemView）时，使用缓存中已经渲染界面，更新数据后上屏

动画方案(抱歉，我们暂时还没做到 CSS 的 transtition)

- 一次性动画下发，性能更好
- 可以应用用样式上的任何参数

```
class AnimationExample extends React.Component {
  componentWillMount() {
    this.heightAnimation = new Animation({
      startValue: 40,   // 动画开始值
      toValue: 80,      // 动画结束值
      duration: 1000,   // 动画持续时长
    });
  }
  componentWillUnmount() {
    //如果动画没有销毁，需要在此处保证销毁动画，以免动画后台运行耗电
    this.verticalAnimation.destroy();
  }
  render() {
    return (
      <View>
        <Text>高度动画</Text>
        <View onClick={() => this.verticalAnimation.start()}>
          <Text>开始</Text>
        </View>
        <View style={[
          { width: 40, height: 40, backgroundColor: 'red' },
          { height: this.heightlAnimation } // 指定动画样式
        ]} />
      </View>
    );
  }
}
```

```
<template>
  // Vue 抽象成了一个组件，这样可以使用组件的生命周期
  <animation :playing="playing" :actions=”heightAnimation" class="animation">
    <slot />
  </animation>
</template>

<script>
export default {
  data() {
    return {
      playing: true,          // 数据驱动控制动画播放状态
      heightAnimation: {
        height: {             // 指定动画样式
          startValue: 40,     // 动画开始值
          toValue: 40,        // 动画结束值
          duration: 1000,     // 动画持续时长
        },
      },
    };
  },
};
</script>

<style scope>
  .animation {
    width: 40px;
    height: 40px;
  }
</style>
```

总结

拥抱社区

紧跟W3C标准
腾讯开源协同
即将对外开源

追求性能

C++ 模块
自绘、原生混合绘图
组件复用

更多组件

QG Canvas
Lottie/PAG 动画
腾讯地图

设计适用

设计稿直出代码

总结

Hippy 的方向是精简版的浏览器

对前端：把 H5 的开发体验带到跨端
对终端：更强的扩展、性能和控制力

感谢 | TLC 2019 大前端信息流

Hippy交流群

该二维码7天内(8月20日前)有效，重新进入将更新