多端统一框架 Taro实战

8：30

环境准备

1. 微信小程序开发工具 https://developers.weixin.qq.com/miniprogram/dev/devtools/download.html?t=19030616
2. 百度小程序开发工具 https://smartprogram.baidu.com/docs/develop/tutorial/index_first/
3. 支付宝小程序开发工具 https://docs.alipay.com/mini/ide/download
4. 头条小程序开发工具 https://developer.toutiao.com/docs/devtool/versionUpdate.html
5. taro
6. Taro-ui https://taro-ui.aotu.io/#/

讲师个人介绍

前百度前端架构师 8年前端经验，react/vue源码级，擅长nodejs

知识点

小程序缺陷

小程序对于前端程序员来说应该算得上是福音了，用前端相关的技术，获得丝般顺滑的 `Native` 体验，前端们又可以在产品小姐姐面前硬气一把了。可以说小程序给前端程序员打开了一扇新的大门，大家都应该感谢微信，但是从开发的角度来说，小程序的开发体验就非常值得商榷了，不仅语法上显得有些不伦不类，而且有些莫名其妙的坑也经常让人不经意间感叹一下和谐社会，从市面上层出不穷的小程序开发框架就可见一斑。

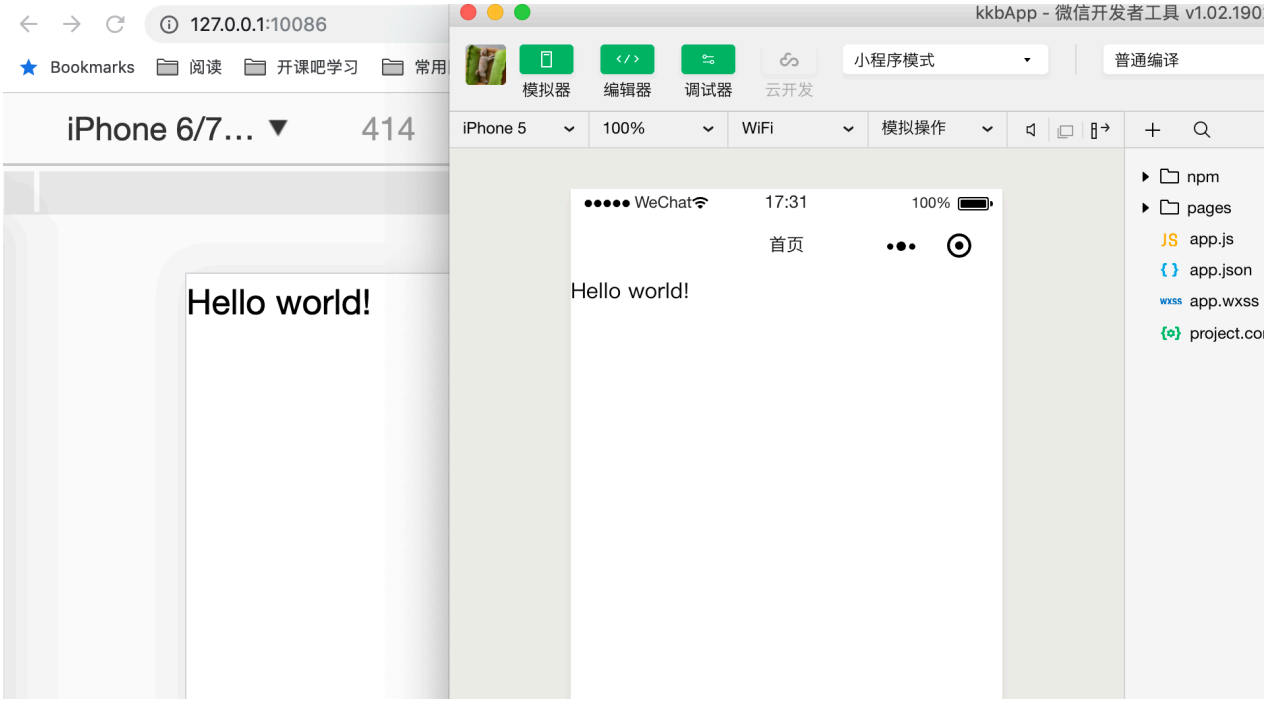
1. 虽然现在能用npm了，但是还是不能用sass等预处理器
2. 不是完整的ES 最新语法语法支持

使用react写小程序

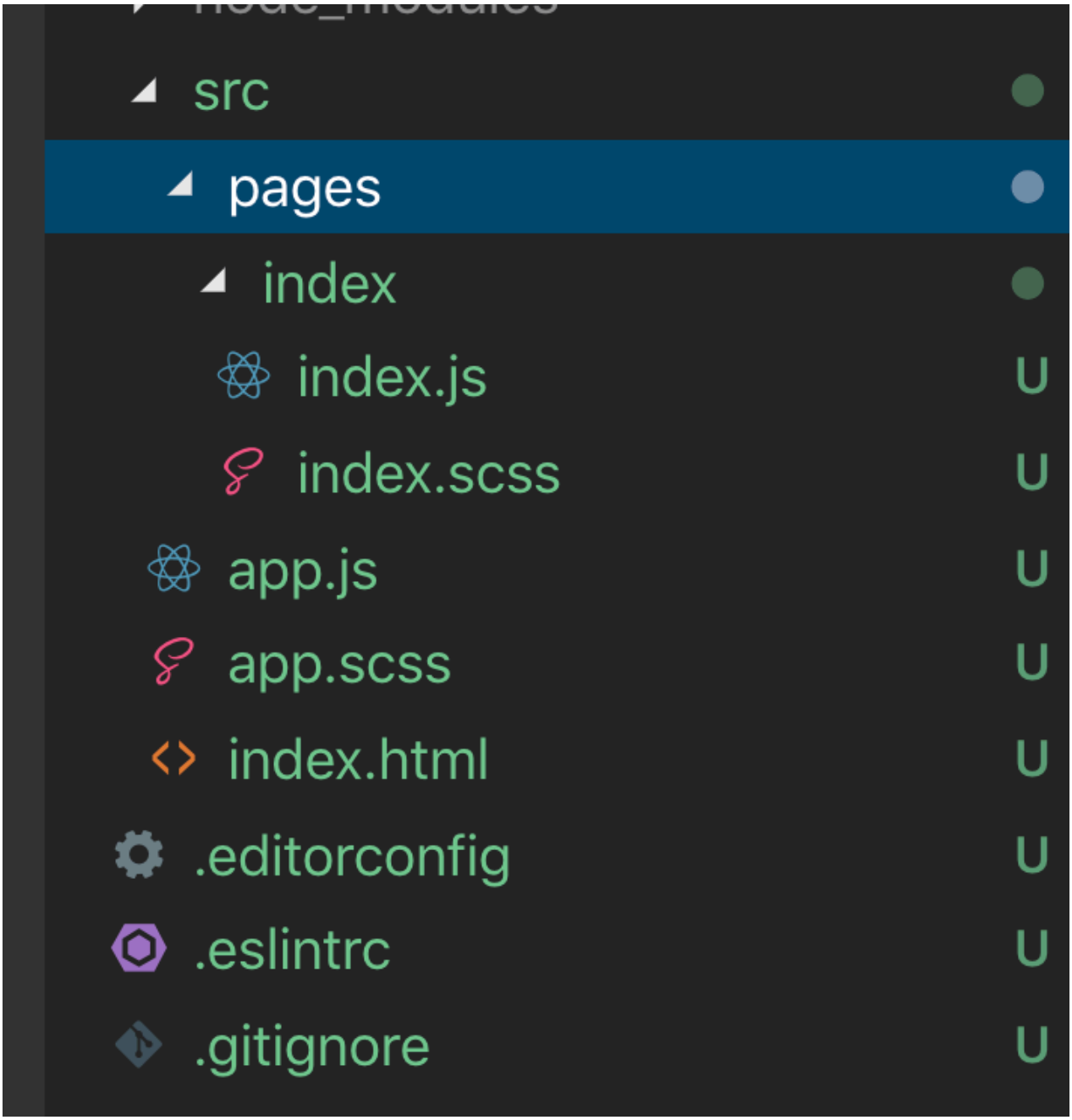
市面上陆陆续续出现了wepy,mpvue uni-app等框架，都是基于vuejs的，直到京东出品了taro 我们可以很愉快使用react来开发小程序，来体验一下

```
npm install -g @tarojs/cli
taro init kkbApp
cd kkbApp
npm run dev:h5
npm run dev:weapp
```

使用微信开发者工具打开项目和浏览器



bingo 我们来看下代码结构



app.js

```
import Taro, { Component } from '@tarojs/taro'
import Index from './pages/index'

import './app.scss'
```

```
// 如果需要在 h5 环境中开启 React Devtools
// 取消以下注释:
// if (process.env.NODE_ENV !== 'production' &&
process.env.TARO_ENV === 'h5')  {
//   require('nerv-devtools')
// }

class App extends Component {

  config = {
    pages: [
      'pages/index/index'
    ],
    window: {
      backgroundTextStyle: 'light',
      navigationBarBackgroundColor: '#fff',
      navigationBarTitleText: 'WeChat',
      navigationBarTextStyle: 'black'
    }
  }

  componentDidMount () {}

  componentDidShow () {}

  componentDidHide () {}

  componentDidCatchError () {}
```

```
  // 在 App 类中的 render() 函数没有实际作用
  // 请勿修改此函数
  render () {
    return (
      <Index />
    )
  }
}

Taro.render(<App />,
document.getElementById('app'))
```

基本完全是react的语法，使用Component+jsx的方式，config就是app.json的配置，我们渲染一个列表 修改page/index.js

```
import Taro, { Component } from '@tarojs/taro'
import { View, Text } from '@tarojs/components'
import './index.scss'

export default class Index extends Component {

  config = {
    navigationBarTitleText: '首页'
  }

  componentWillMount () { }
```

```
  componentDidMount () { }

  componentWillUnmount () { }

  componentDidShow () { }

  componentDidHide () { }

  render () {
    return (
      <View className='index'>
        <Text>Hello world!</Text>
      </View>
    )
  }
}
```

生命周期也完全是React的

渲染列表

```
export default class Index extends Component {

  config = {
    navigationBarTitleText: '今日清单'
```

```
  }
  constructor(props){
    super(props)
    this.state = {
      todos:[
        {title:'吃饭',done:false},
        {title:'睡觉',done:false},
        {title:'开课吧学习',done:false},
      ]
    }
  }
  render () {
    return (
      <View className='index'>
        <Text>Hello 开课吧!</Text>
        {
          this.state.todos.map(todo=>{
            return <View>{todo.title}</View>
          })
        }
      </View>
    )
  }
}
```

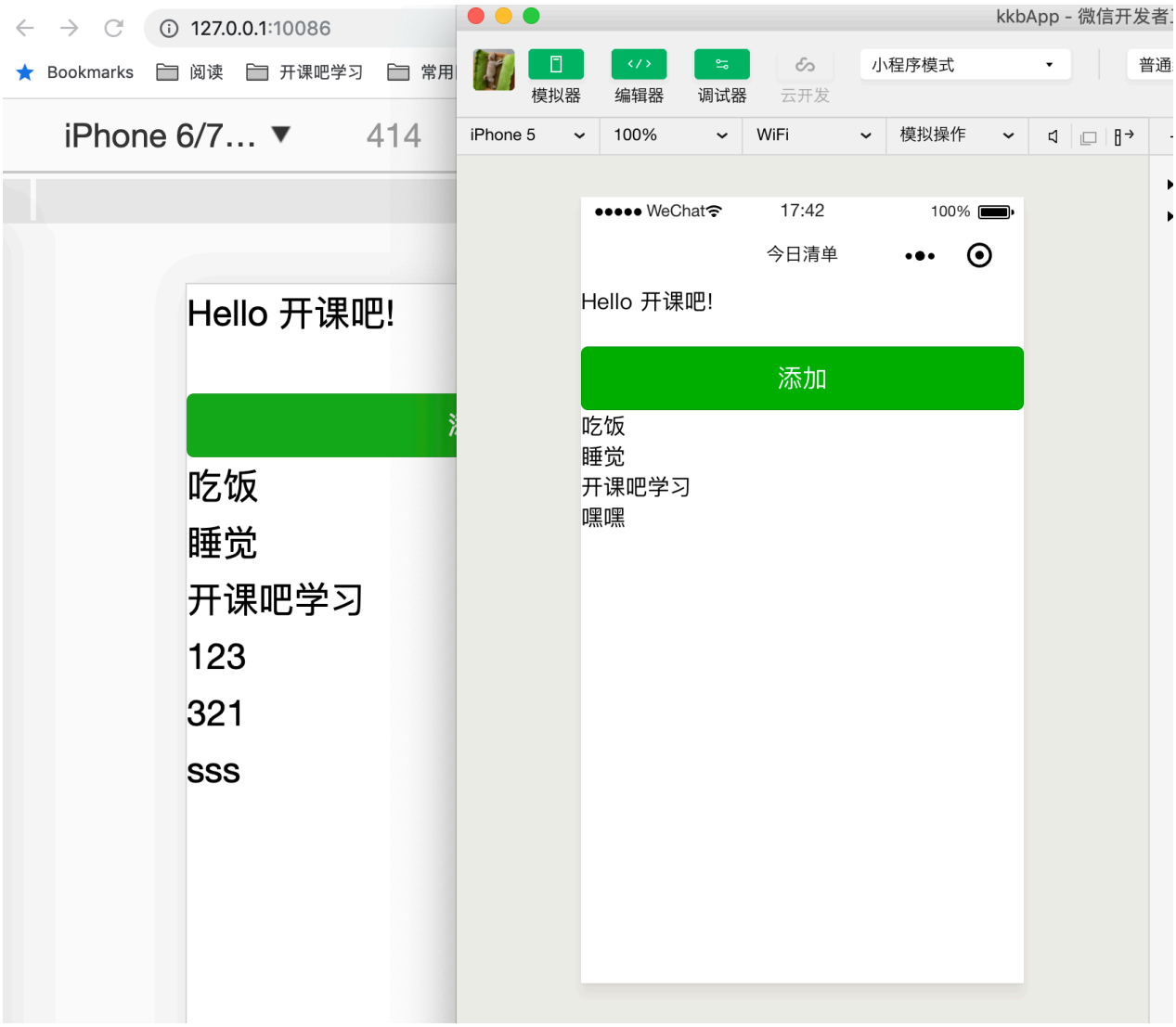
添加用户输入

```
import Taro, { Component } from '@tarojs/taro'
import { View, Text ,Input,Button} from
'@tarojs/components'
import './index.scss'

export default class Index extends Component {

  config = {
    navigationBarTitleText: '今日清单'
  }
  constructor(props){
    super(props)
    this.state = {
      val:'',
      todos:[
        {title:'吃饭',done:false},
        {title:'睡觉',done:false},
        {title:'开课吧学习',done:false},
      ]
    }
  }
  handleInput = (e)=>{
    this.setState({
      val:e.detail.value
    })
  }
  handleClick = ()=>{
    this.setState({
      todos:[...this.state.todos,
{title:this.state.val,done:false}],
```

```
      val:''
    })
  }
  render () {
    return (
      <View className='index'>
        <Text>Hello 开课吧!</Text>
        <View>
          <Input value={this.state.val} onInput=
{this.handleInput}></Input>
        </View>
        <Button type='primary' onClick=
{this.handleClick}>添加</Button>
        {
          this.state.todos.map(todo=>{
            return <View>{todo.title}</View>
          })
        }
      </View>
    )
  }
}
```



添加UI库

taro-ui

```
npm install taro-ui --save
```

H5需要额外一个小配置，在 taro 项目的 `config/index.js` 中新增如下配置项：

```
h5: {
  esnextModules: ['taro-ui']
}
```

引入样式，在app.scss入口处

```
@import "~taro-ui/dist/style/index.scss"; // 引入组件样式，仅需引入一次即可
```

优化列表

```
import Taro, { Component } from '@tarojs/taro'
import { View, Text ,Input,Button} from '@tarojs/components'
import './index.scss'
import { AtButton,AtInput,AtList, AtListItem } from 'taro-ui'

export default class Index extends Component {

  config = {
    navigationBarTitleText: '今日清单'
  }
  constructor(props){
    super(props)
    this.state = {
      val:'',
      todos:[
        {title:'吃饭',done:false},
        {title:'睡觉',done:false},
```

```
        {title:'开课吧学习',done:false},
      ]
    }
  }
  handleInput = (val)=>{
    this.setState({
      val
    })
  }
  handleClick = ()=>{
    this.setState({
      todos:[...this.state.todos,
{title:this.state.val,done:false}],
      val:''
    })
  }
  render () {
    return (
      <View className='index'>
        <Text>Hello 开课吧!</Text>


        <AtInput value={this.state.val} onChange=
{this.handleInput}></AtInput>
        <AtButton type='primary' onClick=
{this.handleClick}>添加</AtButton>
        <AtList>

        {
          this.state.todos.map((todo,i)=>{
```

```
            return <AtListItem
                      title={i+':'+todo.title}
                    ></AtListItem>
          })
        }
        </AtList>

      </View>
    )
  }
}
```

列表状态 + 样式

ww使用list的switch配置

```
import Taro, { Component } from '@tarojs/taro'
import { View, Text ,Input,Button} from
'@tarojs/components'
import './index.scss'
import { AtButton,AtInput,AtList, AtListItem } from
'taro-ui'
```

```
export default class Index extends Component {

  config = {
    navigationBarTitleText: '今日清单'
  }
  constructor(props){
    super(props)
    this.state = {
      val:'',
      todos:[
        {title:'吃饭',done:true},
        {title:'睡觉',done:false},
        {title:'开课吧学习',done:false},
      ]
    }
  }
  handleInput = (val)=>{
    this.setState({
      val
    })
  }
  handleClick = ()=>{
    this.setState({
      todos:[...this.state.todos,
{title:this.state.val,done:false}],
      val:''
    })
  }
  handleSwitchChange = (e,i)=>{
```

```
    const todos = [...this.state.todos]
    todos[i].done = e.detail.value
    this.setState({
      todos:todos
    })
  }
  render () {
    return (
      <View className='index'>
        <Text>Hello 开课吧!</Text>


        <AtInput value={this.state.val} onChange=
{this.handleInput}></AtInput>
        <AtButton type='primary' onClick=
{this.handleClick}>添加</AtButton>
        <AtList>

        {
          this.state.todos.map((todo,i)=>{
            return <AtListItem
                      className={{done:todo.done}}
                      title={i+':'+todo.title}
                      isSwitch
                      switchIsCheck={todo.done}
                      onSwitchChange=
{(e)=>this.handleSwitchChange(e,i)}
                    ></AtListItem>
          })
        }
```

```
        </AtList>

      </View>
    )
  }
}
```

index.scss

```
.done{
    text-decoration: line-through;
    color:red;
}
```

弹窗信息

新增一个清空已完成的功能 Taro封装了弹窗等功能，很方便多端统一弹出信息 **https://nervjs.github.io/taro/docs/api-desc.html**

Taro.showLoading 和 Taro.hideLoading

```
handleClear = ()=>{
  Taro.showLoading({
    title: '清理中'
  })
  setTimeout(()=>{
    this.setState({
      todos: this.state.todos.filter(v=>!v.done)
    })
    Taro.hideLoading()
  },2000)
}
```

数据持久化

Taro.setStorageSync 和getStorageSync,小程序端使用小程序的存储，H5使用localStorage，接口一致

```
constructor(props){
  super(props)
  this.state = {
    val:'',
    todos:Taro.getStorageSync('todos')||[]
  }
}
save(){
   Taro.setStorageSync('todos', this.state.todos)

}
```

```
    this.setState({
      todos:[...this.state.todos,
{title:this.state.val,done:false}],
      val:''
    },this.save)
```

页面导航

直接app.js配置tabbar，小程序端原生，H5使用组件模拟

```
  config = {
    pages: [
      'pages/index/index',
      "pages/user/user"
    ],
    window: {
      backgroundTextStyle: 'light',
      navigationBarBackgroundColor: '#fff',
      navigationBarTitleText: 'WeChat',
      navigationBarTextStyle: 'black'
    },
    tabBar: {
      selectedColor: "#b4282d",
      list: [{
        pagePath: "pages/index/index",
        iconPath: "./assets/home.png",
```

```
        selectedIconPath: "./assets/home-
active.png",
        text: "首页"
      },
      {
        pagePath: "pages/user/user",
        iconPath: "./assets/user.png",
        selectedIconPath: "./assets/user-
active.png",
        text: "个人"
      }]
    }
  }
```

微信小程序

npm run build:weapp

头条小程序

Npm run build:tt

支付宝

npm run build:alipay

百度小程序

```
npm run dev:swan
```

React-native

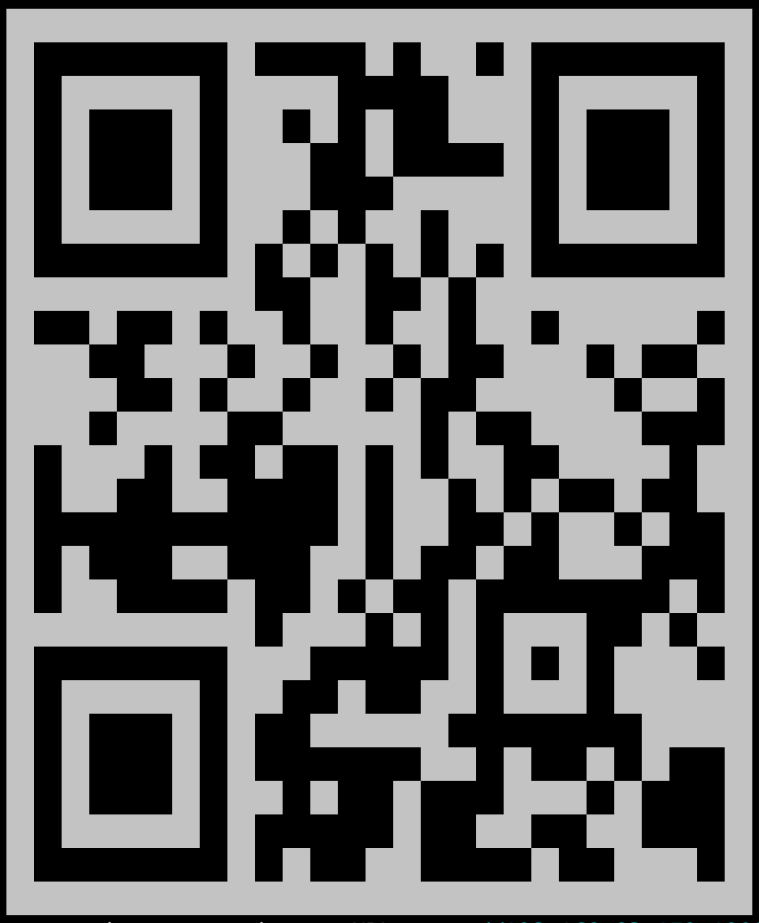
https://nervjs.github.io/taro/docs/react-native.html

npm run dev:rn

```
npm (node)

初始化完毕，监听文件修改中...

Packager started!
```

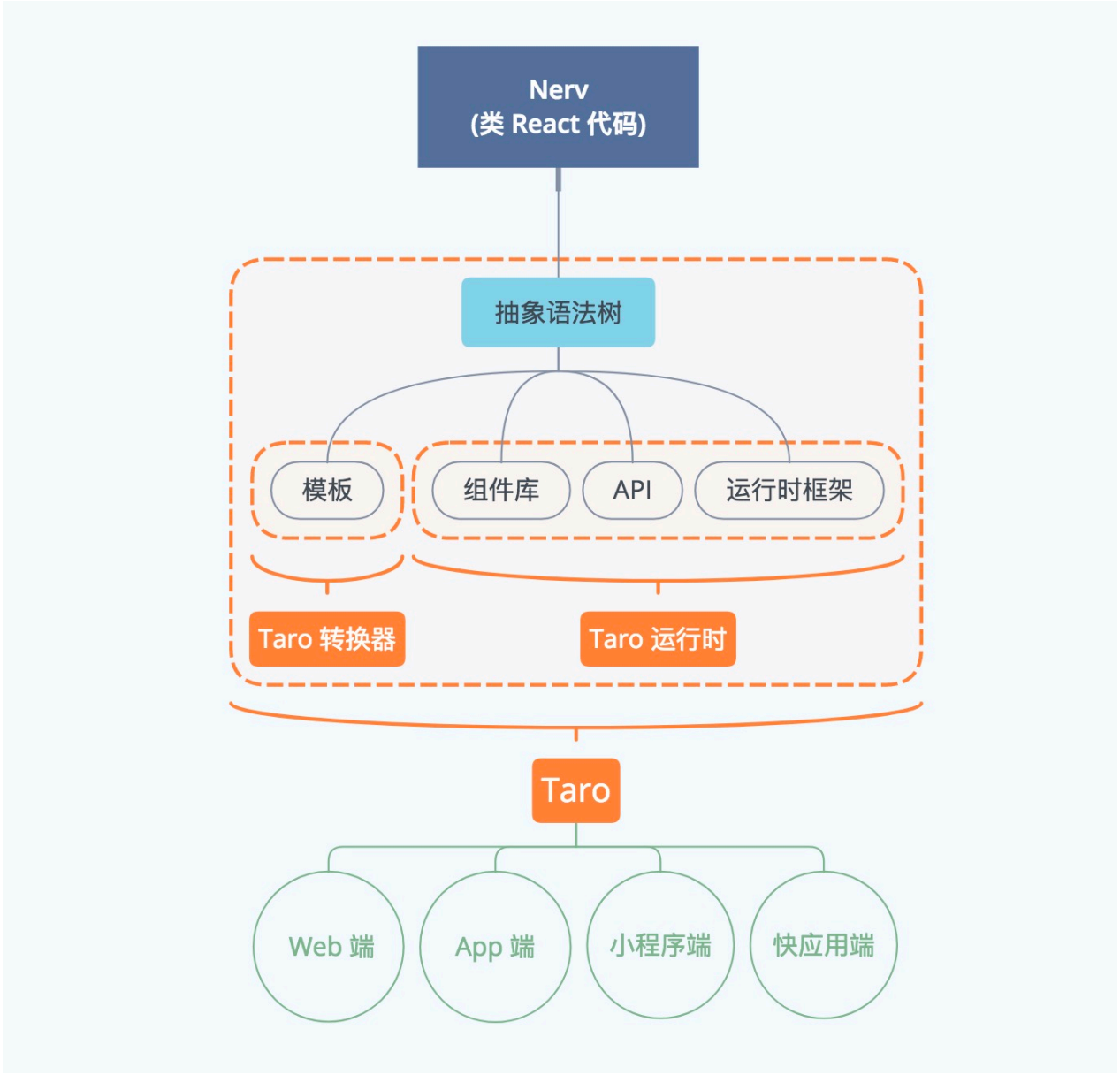


```
Your app is now running at URL: exp://192.168.83.159:19000
View your app with live reloading:
  Android device:
    -> Point the Expo app to the QR code above.
       (You'll find the QR scanner on the Projects tab of the app.)
  iOS device:
    -> Press s to email/text the app URL to your phone.
  Emulator:
    -> Press a (Android) or i (iOS) to start an emulator.
Your phone will need to be on the same local network as this computer.
For links to install the Expo app, please visit https://expo.io.
Logs from serving your app will appear here. Press Ctrl+C at any time to stop.

 › Press a to open Android device or emulator, or i to open iOS emulator.
 › Press s to send the app URL to your phone number or email address
 › Press q to display QR code.
 › Press r to restart packager, or R to restart packager and clear cache.
 › Press d to toggle development mode. (current mode: development)
```

taro-ui

Taro原理



小程序组件

组件	作用
view	容器组件
text	文子组件
image	图片组件
icon	图标组件
swiper	轮播组件
button	按钮组件
…	…

Web 组件

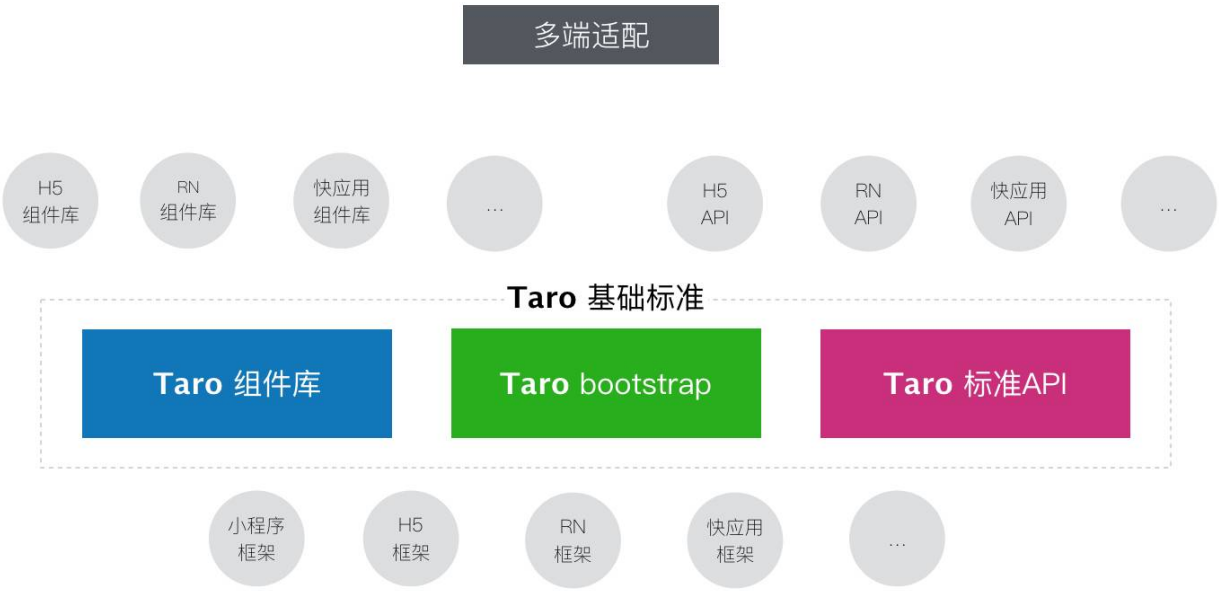
组件	作用
div	块级元素标签
span	行内元素标签
a	超链接标签
table	表格标签
ul	无序列表
p	段落标签
…	…

小程序 API

API	作用
wx.request	网络请求
wx.uploadFile	上传文件
wx.downloadFile	下载文件
wx.login	微信登录
wx.getStorage	获取本地存储
wx.getStorage	设置本地存储
…	…

Web API

组件	作用
window.fetch	网络请求
window.localStorage	本地存储相关
window.open	打开新页面
window.scrollTo	页面滚动
window.navigator	浏览器相关信息
window.location	文档当前位置信息
…	…



框架对比

	微信小程序	mpvue	wepy	Taro
语法规范	小程序规范	vue.js 规范	类 vue.js 规范	React 规范
模板系统	字符串模板	字符串模板	字符串模板	JSX
类型系统	不支持	业务代码	业务代码	业务代码 + JSX 模板
组件规范	小程序组件	html 标签 + 小程序组件	小程序组件	小程序组件
样式规范	wxss	sass, less, postcss	sass, less, styus	sass, less, postcss
组件化	小程序组件化	vue 组件化规范	自定义组件化	React 组件化规范
多端复用	无	复用为 H5 *	复用为 H5	复用为 H5 *
自动构建	无	Webpack 构建	内建构建系统	内建构建系统 + Webpack
上手成本	全新学习	熟悉 Vue.js	熟悉 Vue.js + wepy	熟悉 React
数据流管理	不支持	Vuex	Redux	Redux

* mpvue 将通过 weex 支持移动端，Taro 将通过 React Native 支持移动端

特性

1. 多端运行
    1. 微信小程序
    2. 百度小程序
    3. 头条小程序
    4. 支付宝小程序
    5. React-native
    6. H5
2. 完全使用React语法标准
3. 组件化
4. 全面支持typescript
5. 全面支持redux / mobx
6. 开发体验，代码智能提示
7. 现代化开发流程

扩展

---

1. 多端差异
    1. 条件编译
2. 更丰富的组件
3. 用户系统

总结

下次课预告

还是我， React源码