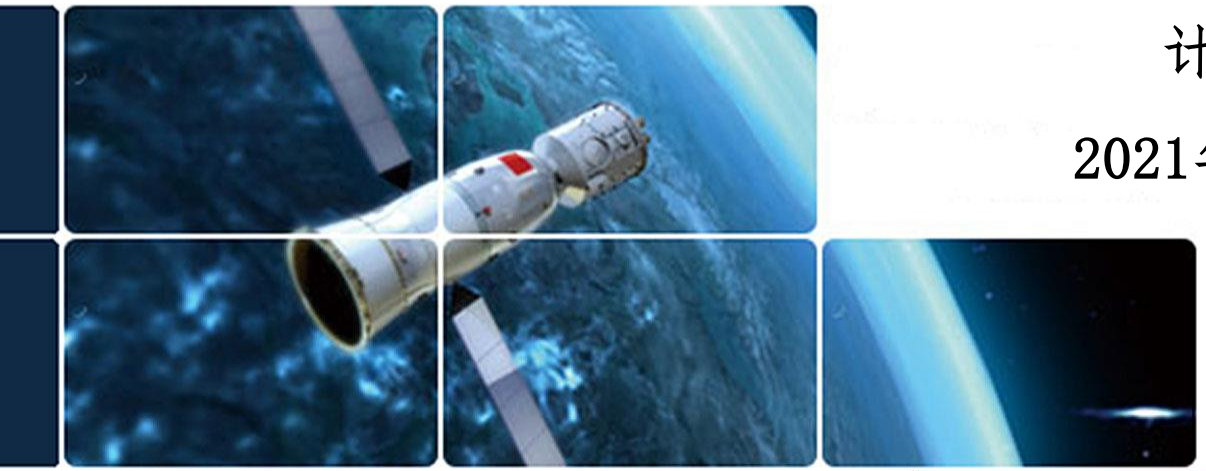
第10讲 Hybrid应用开发技术



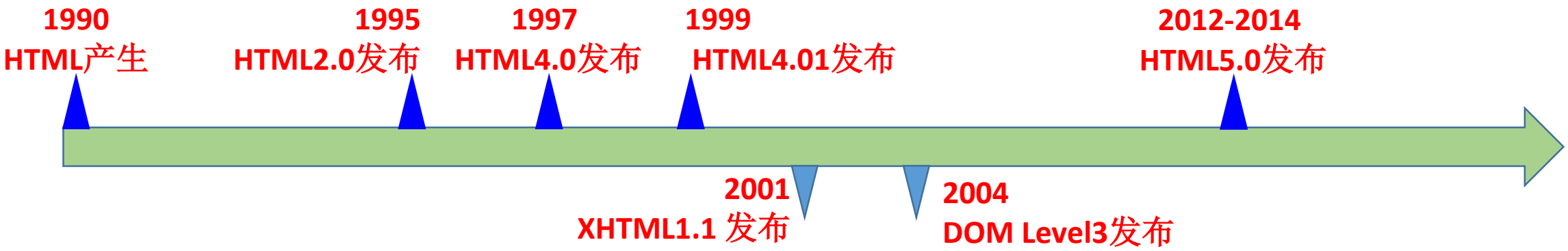
计算学部

2021年12月16日

主要内容

- 了解 HTML5
- 移动应用的种类
- 低代码开发
- 一个简单的Hybrid案例

什么是HTML5



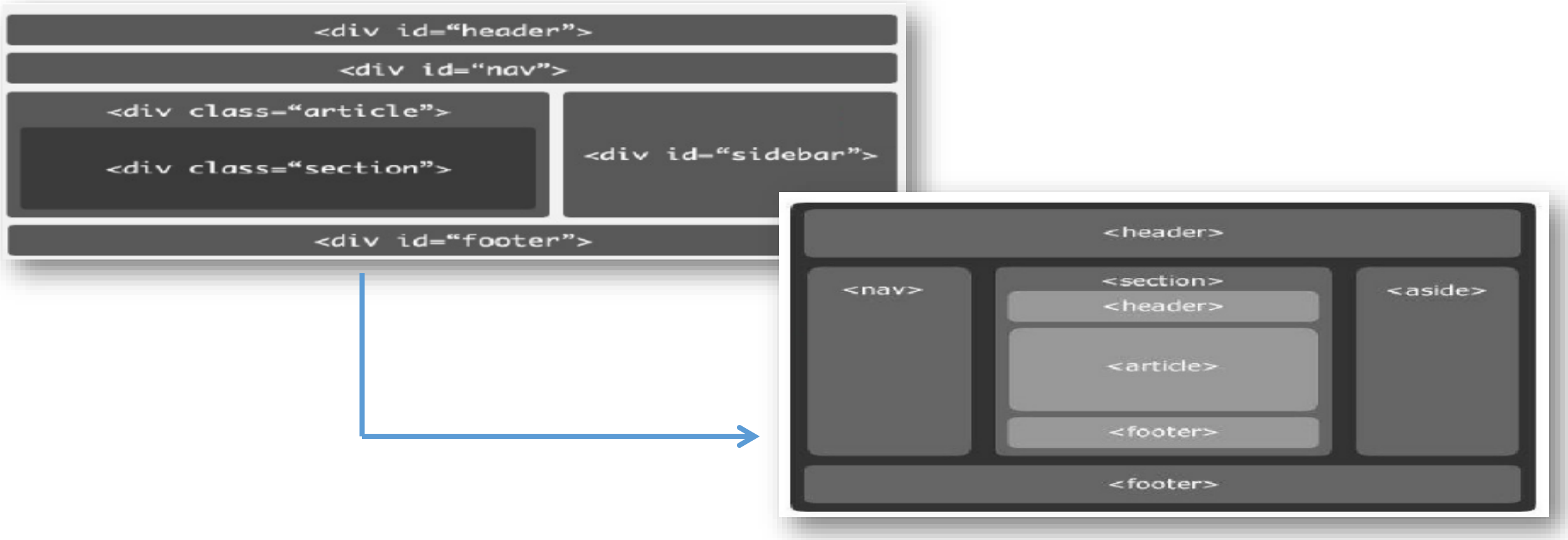
- ***HTML 5***是构建Web内容的一种语言描述方式，一个新的网络标准，是互联网的下一代标准，是***HTML***、***XHTML***和***HTML DOM***的新标准
- ***HTML 5***的目标是为了减少互联网富应用对***Flash***、***Silverlight***和***JavaFX***等浏览器插件的依赖，并且提供更多能有效增强网络应用的API

HTML5的新特性

- 语义更加丰富
- 实现离线存储
- 设备更通用
- 支持网络连接
- 支持多媒体
- 支持三维、图形与特效
- 性能与集成度提高
- 支持CSS3

HTML5语义化的标签

- ***HTML 5***引入了新的HTML元素。赋予网页更好的意义和结构。增加对RDFa（资源描述框架），微数据与微格式等方面的支持，更有助于构建对程序、对用户都更有价值的数据驱动的Web



HTML5十个常用的新标签

- <*article*>：定义文章
- <*aside*>：定义文章侧边栏
- <*figure*>：媒体对象及文字
- <*figcaption*>：定义fig标题
- <*footer*>：定义页脚
- <*header*>：定义页眉
- <*hgroup*>：定义标题组合
- <*nav*>：定义导航
- <*section*>：定义文档区段
- <*time*>：定义日期和时间

被弃用的标签：<*acronym*>、<*applet*>、<*basefont*>、<*big*>、<*center*>、<*dir*>、<*font*>、<*frame*>、<*s*>、<*isindex*>、<*noframes*>、<*frameset*>、<*strike*>、<*tt*>、<*u*>、<*xmp*> 等

- HTML5表单设计上功能更强大。<input>类型和属性的多样性增强了*HTML*可表达的表单形式，加上新增加的表单标签，使原本需要*JavaScript*实现的控件，可直接使用*HTML5*表单实现；内容提示、焦点处理、数据验证等功能也可通过*HTML5*智能表单属性标签完成。

标签	描述
<datalist>	定义选项列表。与 input 元素配合使用该元素定义 input 可能取值
<keygen>	规定用于表单的密钥对生成器字段
<output>	定义不同类型输出，比如脚本输出

HTML5本地存储

- ***HTML5*** 本地存储是一个比***cookie***更好的本地存储方式。基于HTML5开发的应用启动时间更短，联网速度更快，得益于***HTML5 APP Cache***及本地存储功能

- 客户端存储数据的两个对象为：
  - ***localStorage***： 没有时间限制的数据存储
  - ***sessionStorage***： 针对***session*** 的数据存储

接口功能	接口名称
保存数据	***setItem(key,value);***
读取数据	***getItem(key);***
删除单个数据	***removeItem(key);***
删除全部数据	***clear();***
获得数据索引	***key(index);***

- HTML5拥有更有效的服务器推送技术。*WebSocket*和*Server-Sent Event（SSE）*是其中两个特性，这两个特性能够实现数据推送功能

- ***WebSocket***是***HTML5***开始提供的一种在单个 ***TCP*** 连接上进行全双工通讯的协议，***WebSocket*** 协议本质上是一个基于 ***TCP***的协议，支持***WebSocket***和***WebSocketSecure***

- ***WebSocket*** 减少了延迟，浏览器通过 ***JavaScript*** 向服务器发出建立 ***WebSocket***连接请求，一旦建立WebSocket 连接，浏览器和服务器之间就形成了一条快速通道，两者之间就直接可以数据互相传送

- ***WebSocket API*** 是纯事件驱动的。应用程序代码监听***WebSocket***对象上的事件，以便处理输入数据和连接状态的改变，支持***open***、***message***、***error***和***close***事件

- 获取 ***WebSocket***连接后，可以通过 ***send()*** 方法向服务器发送数据，并通过 ***onmessage*** 事件接收服务器返回的数据，使用***close***() 方法， 可以关闭***WebSocket***连接或者终止连接尝试

- *HTML5* 服务器发送事件*SSE*（*server-sent event*）允许网页获得来自服务器的更新，使用*HTTP/HTTPS*协议，通过独立*Ajax*请求从客户端向服务端传送数据，相对*WebSocket*，使用*Ajax*会增加开销
- 对于服务器端返回的响应，浏览器端需要使用 *EventSource* 对象进行处理，支持*open*、*message*、*error*事件
- 基本步骤：
  - 创建 *EventSource* 对象，规定发送更新的页面

    ```
    var source=new EventSource("demo_sse.php");
    ```

  - 每接收到一次更新，就会发生 *onmessage* 事件
  - 当 *onmessage*事件发生时，可进行后续处理

HTML5多媒体支持

- ***HTML5*** 新增了视频和音频标签。 ***<video>*** 含视频标准方法，***<audio>***含音频标准方法

```
<video width="320" height="240" controls="controls">
  <source src="movie.ogg" type="video/ogg">
  <source src="movie.mp4" type="video/mp4">
Your browser does not support the video tag.
</video>
```

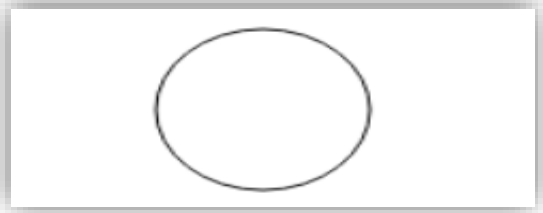
- ***<video>*** 和 ***<audio>*** 元素的方法、属性和事件均可使用***JavaScript***进行控制
  - 方法用于播放、暂停以及加载等功能
  - 属性（比如时长、音量等）可被读取或设置
  - ***DOM*** 事件起到通知作用，比如***<video>*** 元素开始播放、已暂停，已停止等等

HTML绘图支持

□*HTML5新增了<canvas>图形标签。* <*canvas*> 标签提供图形容器，可通过多种方法使用Canvas绘制路径,盒、圆、字符以及添加图像

```
<canvas id="myCanvas" width="200" height="100"
style="border:1px solid #000000;">
</canvas>
```

```
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.beginPath();
ctx.arc(95,50,40,0,2*Math.PI);
ctx.stroke();
```



HTML绘图支持

HTML5新增了SVG（矢量图形）绘图功能。定义用于网络的基于矢量的图形，SVG图形可通过文本编译器创建和修改，可被搜索、索引、脚本化或压缩，可伸缩，可在任何分辨率下高质量显示，可在图像质量不下降的情况下被放大

Canvas	*SVG*
依赖分辨率	不依赖分辨率
不支持事件处理	支持事件处理
能够以.png或.jpg格式保存结果图像	复杂度高会降低渲染速度
文字呈现功能比较简单	适合大型渲染区域
最合适图像密集的游戏	不适合游戏应用

HTML5地理定位支持

- *HTML5* 支持用户定位。使用*getCurrentPosition*()方法来获取用户的位置

```
<script>
    var x=document.getElementById("demo");
    function getLocation()
     {
       if (navigator.geolocation){navigator.geolocation.getCurrentPosition(showPosition);}
      else{x.innerHTML="Geolocation is not supported by this browser.";}
     }
   function showPosition(position)
    {
      x.innerHTML="Latitude: " + position.coords.latitude +
                          "<br />Longitude: " + position.coords.longitude;
    }
</script>
```

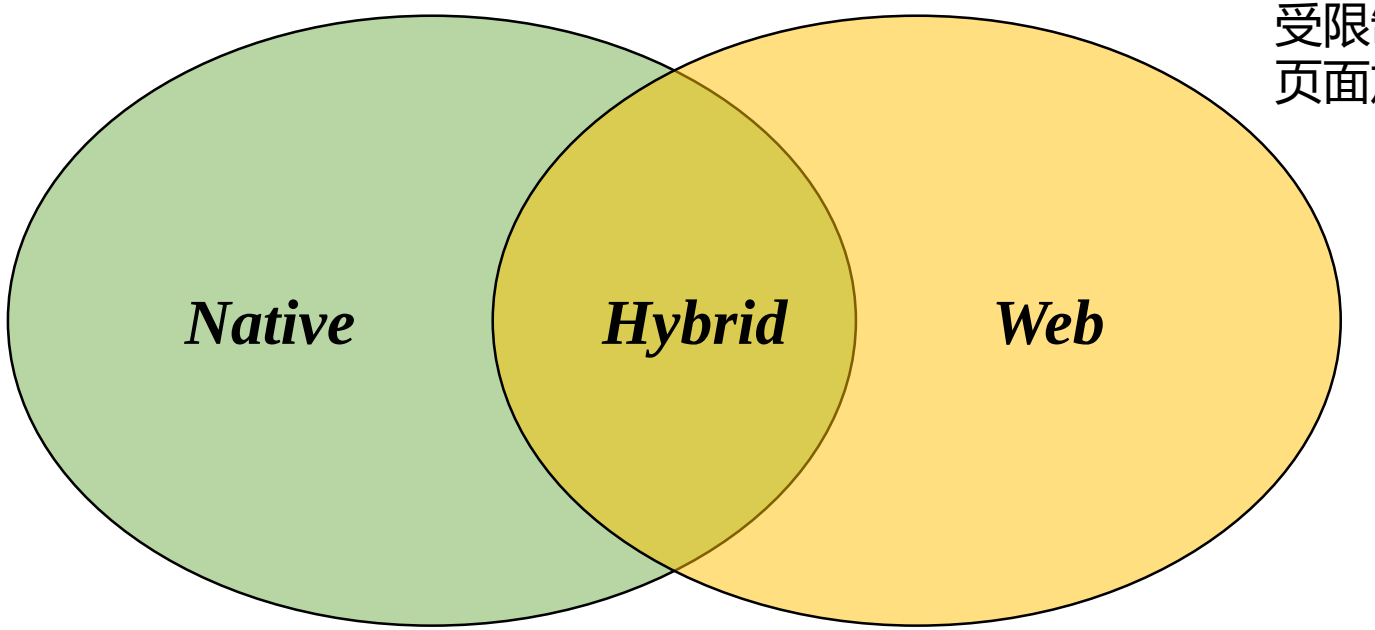
主要内容

- 了解 HTML5
- 移动应用的种类
- 低代码开发
- 一个简单的Hybrid案例

移动应用类型

Native App
语言：*Object C （IOS）*
Java （Android）
Java （Harmony）
页面：存放于本地

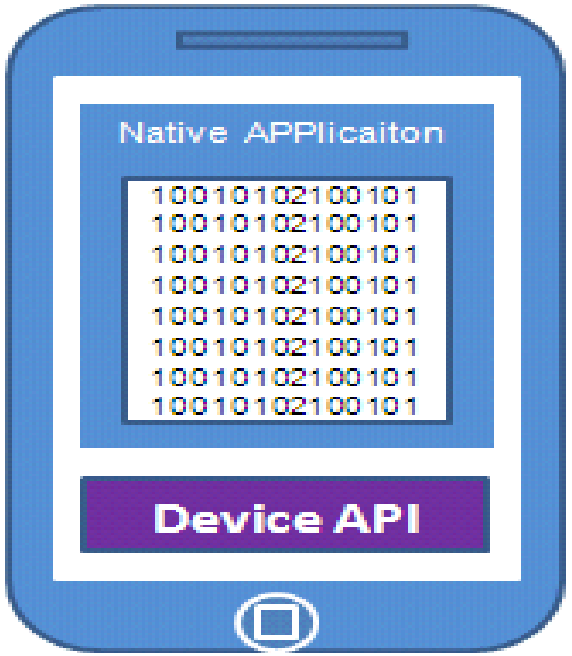
Web App
语言：*HTML+JS+CSS*
受限制于***UI WebView***
页面放于服务器



Hybrid App
语言：*Object C + HTML + JS （IOS）*
Java + HTML + JS （Android）
Java、JS + HTML （Harmony）
受限制于***UI WebView***

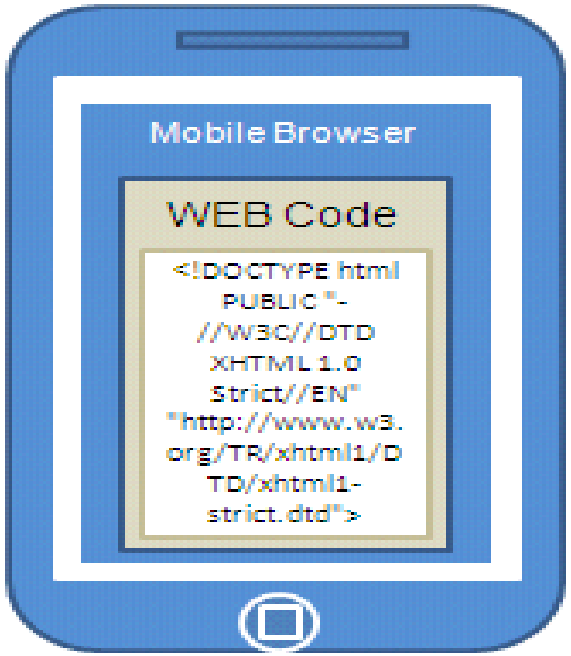
移动应用类型

Native APP



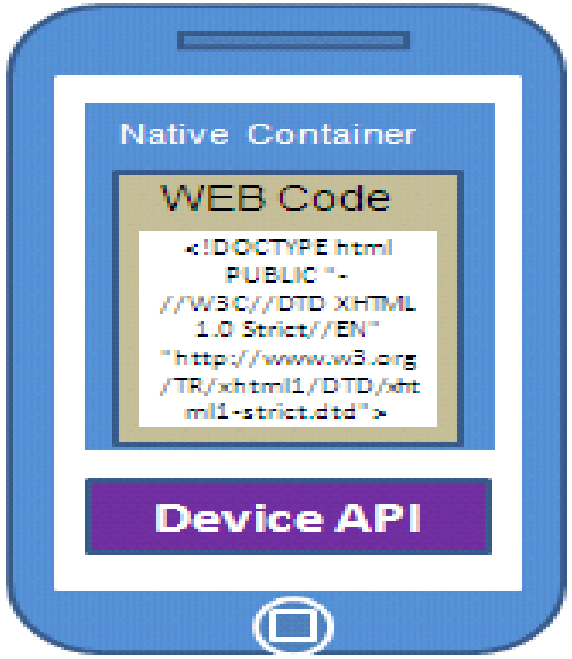
编译后代码执行
设备接口丰富

WEB APP



HTML解释执行
无设备接口或接口少

Hybrid APP



部分HTML解释
部分编译后代码执行
提供少量设备接口

开发模式之Web APP

- ***Web App***是成本最低，最快速地解决方案。响应式设计模式的不断流行，用户页面的设计根据用户行为以及设备环境（系统平台、屏幕尺寸、屏幕定向等）进行响应和调整，为 ***Web App***市场提供了良好的实践场地
- ***Web App***具有如下特点：
  - 使用浏览器运行
  - 采用纯***Web***前端架构，部分手机特性无法访问
  - 多采用***Single Page App***
  - 推广渠道多限于浏览器

开发模式之Native APP

- ***Native App***是最可靠且性能最优的移动平台解决方案，其优势包括：
  - 能够为用户提供最佳体验、最优质的用户界面、最强的交互效果
  - 每一种移动操作系统都需要独立开发项目，其开发设计能够针对不同平台、不同用户需求设计不同体验产品
  - 可自由访问本地资源，能够访问手机的所有功能，如GPS、相机等

- ***Hybrid APP***是一类移动应用，可同时使用网页语言与程序设计语言，可区分平台
- 开发成本和难度比***Native APP***小很多
- 用户需要安装使用

Hybrid APP的分支（1）

- 使用***PhoneGap***、***AppCan***等中间件，以***WebView***作为用户界面层，以***Javascript***作为基本逻辑，与中间件通讯，再由中间件访问底层API
- 这种架构通常强依赖于***WebView***层性能
- 该方案以***web***架构为重

Hybrid APP的分支（2）

- 采用***Adobe Air***、***RubyMotion***、***Appcelerator***或***Xamarin***等非官方语言工具，打包成原生应用
- 由于该类应用并没有完全使用原生提供的语言进行开发，而是通过对开发者提供友好的开发工具，并折中地把这种开发语言转换成原生语言，最终打包出整个应用，所以也属于混合应用范畴
- 该方案采用编译转换方式

Hybrid APP的分支（3）

- 在开发原生应用基础上，嵌入***WebView***，但整体架构使用原生应用提供
- 开发组由***Native***开发人员和***Web***前端开发人员组成。***Native***开发人员写基本架构及API，***Web***开发人员负责开发界面
- 该方案以***Native***架构为重

Hybrid APP与WebView

- 现有*Hybrid*移动应用一般都采用*HTML5*技术，调用系统自带浏览器内核，加载本地或*web*页面和资源，*Web View*对*HTML5*支持程度和性能表现受制于移动设备浏览器内核版本
- *Android*采用了成熟的*Chromium*浏览器内核，对*HTML5*的支持大大增强，*Hybrid App*在*Android*手机上的表现也随之增强
- 另外，随着移动设备硬件配置的迅速提升，一些比较消耗计算能力的*HTML5*特性也能被用于移动设备，并且获得和*Native App*接近的性能效果

Hybrid APP开发的技术难点

- ***Native*与*HTML5*的交互问题**
- 客户端与服务器端的交互问题
- 客户端数据缓存问题
- 消息实时通知（***HTML5*** 的 ***Websocket***技术）
- 客户端浏览器兼容性问题

APP开发模式的比较

	Web App	*Hybrid App*	*Native App*
开发成本	低	中	高
维护更新	简单	简单	复杂
体验	差	优	优
市场认可	不认可	认可	认可
安装	不需要	需要	需要
跨平台	优	优	差

投票反馈

1. 低功耗蓝牙通信中，如下哪个设备最适合充当中心设备（）

单选 非匿名 已结束 64 人已投

(A) 蓝牙耳机 2 票 3%

1190201107-陆国智、□□□、

(B) 蓝牙鼠标 2 票 3%

叶珏相、辛帅1190201909

(C) 智能手机 56 票 87%

1170300117-徐旌翰、刘迪航、1190202122陈孟、... 展开

(D) 蓝牙手环 4 票 6%

2. 低功耗蓝牙设备的广播操作由如下哪类设备发起（）

单选 非匿名 已结束 64 人已投

(A) 中心设备 24 票 37%

1190201107-陆国智、梁浩1190200717、李忠根、1... 展开

(B) 中心设备或外围设备 15 票 23%

叶珏相、刘迪航、1190201403张偲博、吉天相、119... 展开

(C) 外围设备 25 票 39%

1170300117-徐旌翰、刘雨婷、1190202415李浩、... 展开

(D) 都不支持扫描操作 0 票 0%

3. 开发wifi应用程序不需要如下哪类权限（）

单选 非匿名 已结束 64 人已投

(A) 访问网络权限 0 票 0%

(B) 访问Wifi权限 3 票 4%

叶珏相、1190401016王清爽、1190201107-陆国智

(C) 读取传感器权限 53 票 82%

袁文宇、1170300117-徐旌翰、刘迪航、吉天相、11... 展开

(D) 修改网络状态权限 8 票 12%

梁浩1190200717、□□□、、1190202009韩雄宇、郭... 展开

4. 关闭Service可由如下哪个对象完成（）

单选 非匿名 已结束 64 人已投

(A) 启动Service的对象 5 票 7%

袁文宇、BetLiD、杨东晨、惠、马子豪

(B) Service自身 4 票 6%

1170300117-徐旌翰、毛星1190200616、宋祖楠、1... 展开

(C) 设备内任意活动对象 1 票 1%

辛帅1190201909

(D) 启动Service的对象或Service自身 54 票 84%

今日投票

1. 进程间通信IPC的远程服务接口是如何定义的？（ ）

（A）通过AIDL定义的　　（B）通过Java的Interface定义

（C）客户端和服务端各自独立定义即可　（D）通过Java类自由定义

2. 如下哪项不是智能手机提供的系统级服务（ ）

（A）来电服务　　（B）电源服务

（C）短信服务　　（D）微信消息服务

3. HTML 5.0集成了如下哪些标准？（ ）

（A）HTML4.0　　（B）XHTML

（C）FLASH　　（D）DOM

4. 如下哪类移动应用类型提供的设备接口最丰富（）

（A）Native APP　　（B）都一样，没区别

（C）Web APP　　（D）Hybrid APP

主要内容

- 了解 HTML5
- 移动应用的种类
- 低代码开发
- 一个简单的Hybrid案例

Gartner观点

市场需求暴增
国内低代码尚处于早期阶段，市场需求将会出现暴增，未来市场对于应用开发的需求将远高于IT公司产能，低代码是目前可行的技术方案

5

物联网与AI
与物联网的扩展连接成为趋势，快速连接硬件设备可以帮助实现工业互联网落地，AI平台与低代码平台的融合将成为趋势。

1

平台云化
低代码开发平台云化会成为热点，低成本部署，按License、业务量、按需、分成等商业模式会成为常态

4

客户化开发加速
客户化开发会帮助行业软件实现个性化需求的定制，软件厂商与低代码开发平台合作可以快速完成个性化需求的交付

2

集成能力
低代码开发平台和数据以及业务系统的集成能力变得越来越重要

3

低代码开发平台

一站式前台编排方案

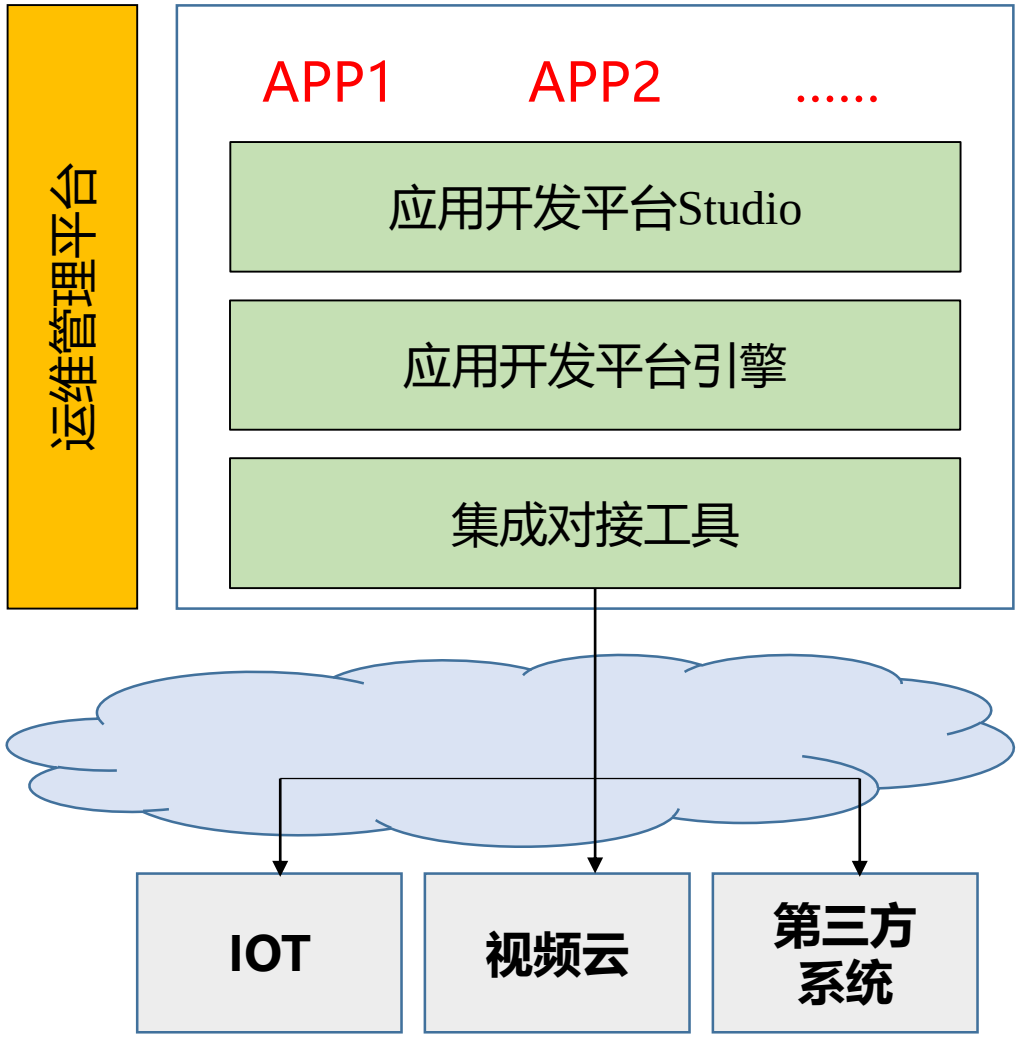
- Web应用构建
- 移动应用构建
- 大屏应用构建

模板

- 脚手架工程
- 页面
- 组件

后台逻辑处理能力

- APP管理
- 对象构建
- 服务编排
- 脚本编写
- 流程管理
- 审批流管理
- 数据接入
- 连接管理



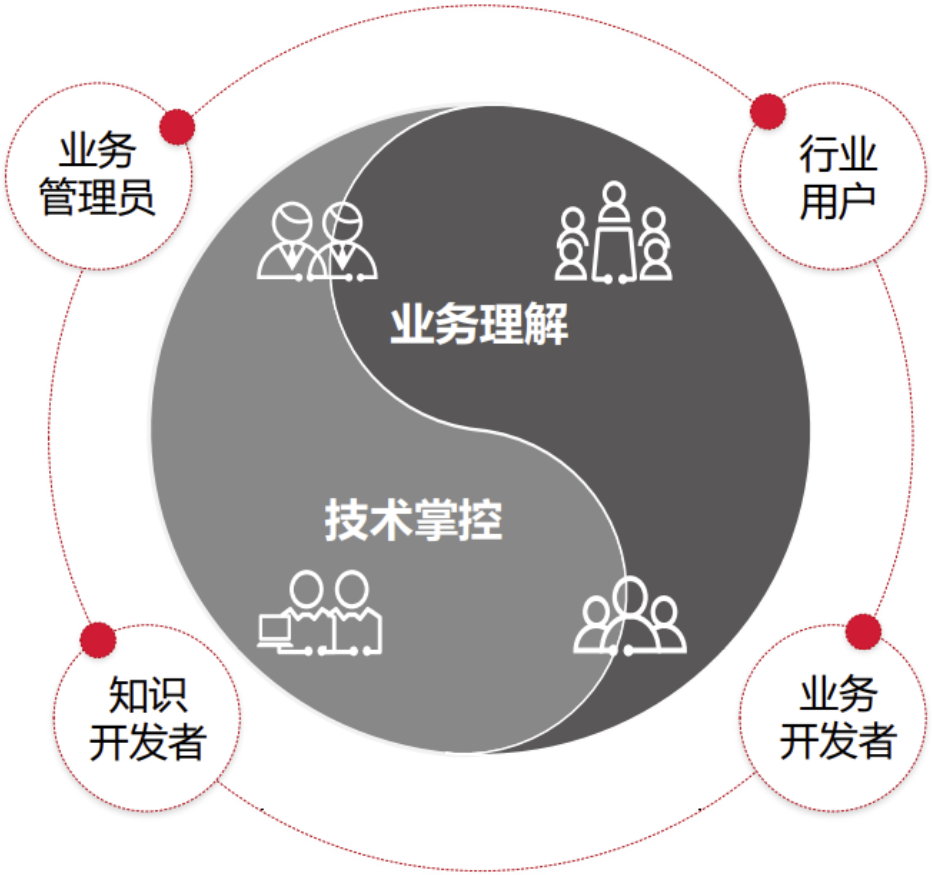
低代码价值-多角色参与的快速交付能力

业务管理员：

- 零代码：*No-Code*
- 友好在线配置开发能力
- 自主开发简单应用

知识开发者：

- 全代码接入：*Full-Code*
- 分层开发、沉淀模板，供其他角色使用
- 开放的原生微服务接入框架



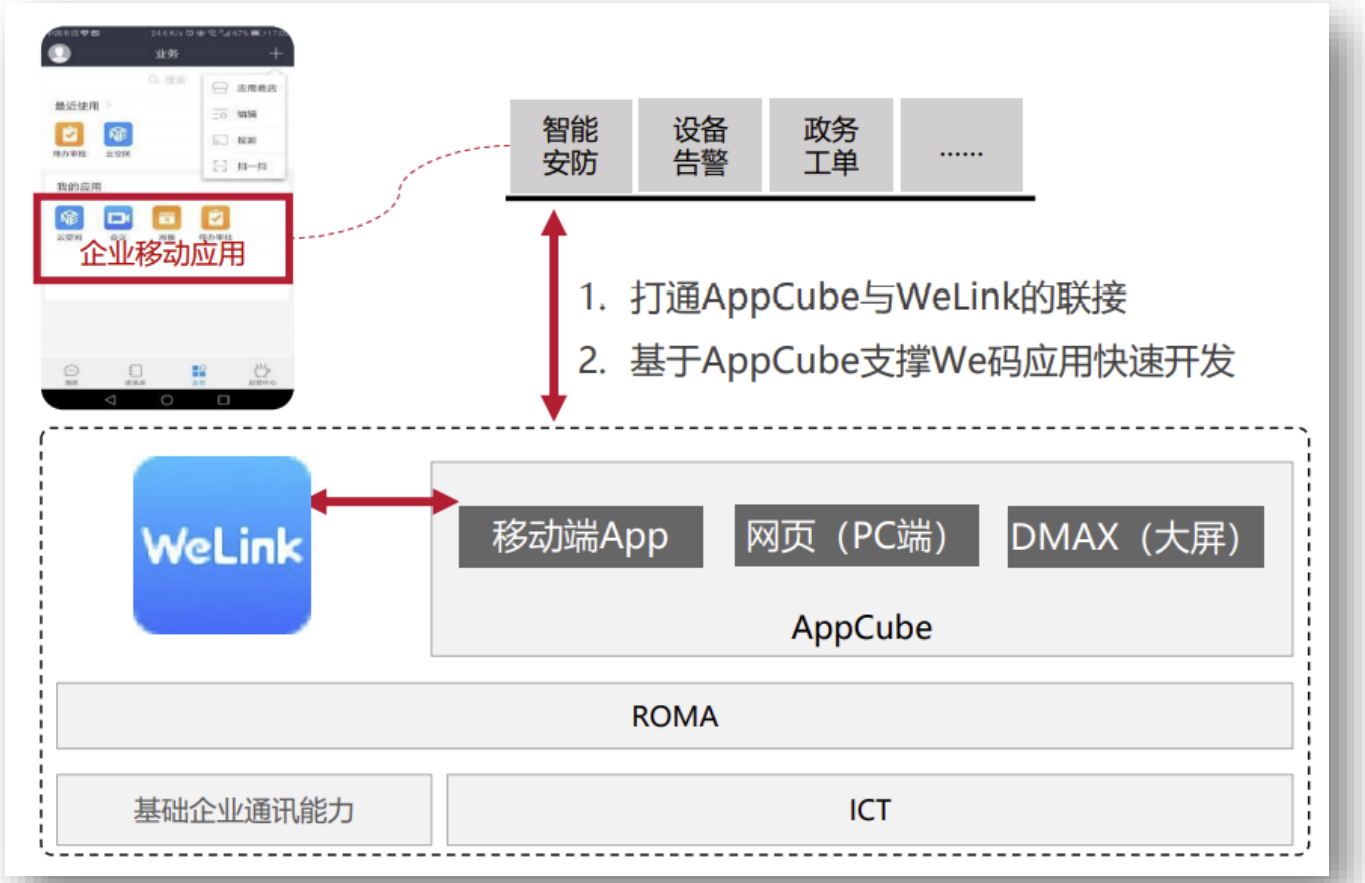
行业用户：

- 个性化：*Personalize*
- 充分的可扩展能力，支撑个性化用户设置
- 业务创新带动应用循序构建

业务开发者：

- 低代码：*Low-Code*
- 线上配置、线上编码
- 全栈开发，降低门槛
- 自动部署和配置，降低工程难度

低代码应用案例-哈工大APP



- 打通AppCube和WeLink的联接，提供一站式移动应用开发环境
- 提供预置UI组件、标准HTML页面快速生成移动应用界面
- 打通多端发布能力：一键发布小程序

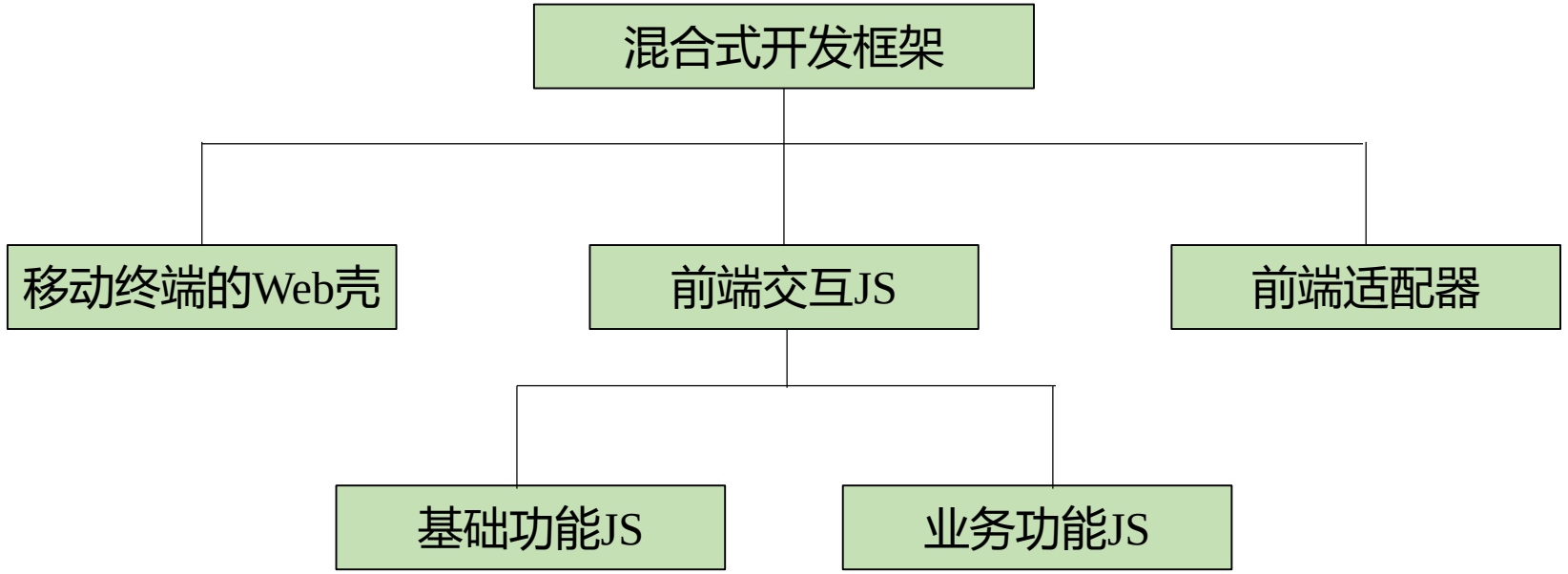
移动端的开发，与PC端开发体验一致，对开发人员的技能要求也一致

示例-泛微的低代码开发平台

主要内容

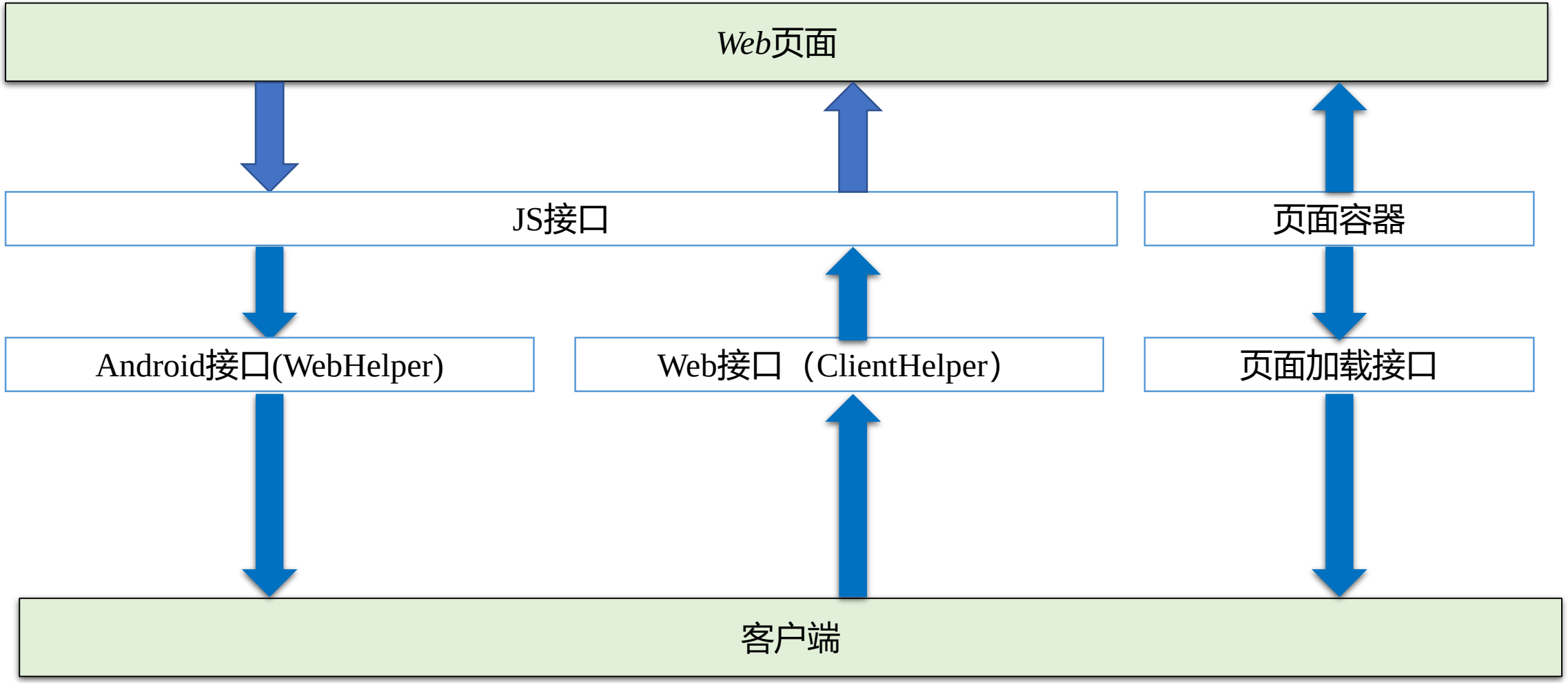
- 了解 HTML5
- 移动应用的种类
- 低代码开发
- 一个简单的Hybrid案例

技术架构



- “壳”定义了接口，允许*JavaScript*调用*Android*应用程序，提供基于*Web*应用程序的*Android API*，将*Web*嵌入到*Android*应用程序中
- *JavaScript*实现基础功能和业务功能
- 适配器用于适配不同终端

技术架构


Web页面
JS接口
页面容器
Android接口(WebHelper)
Web接口（ClientHelper）
页面加载接口
客户端

- 页面加载
  - 页面容器类：是整个框架的核心和基础，实现页面加载，对页面加载完成后的操作提供支持，例如：文件下载、*JS*支持、文件上传，数据缓存、进度条等
  - 页面加载接口：对页面的加载过程进行跟踪；例如：页面加载进度，页面开始加载、页面加载出错、页面加载完成等

- ***JS***调用***Android***功能
  - 网页：页面调用***JS***接口中的具体方法
  - JS接口：调用***Android***接口具体方法
  - Android接口：直接调用框架中集成的功能，或者通过框架接口在应用系统中自定义功能（例如，退出、返回键响应等)
  - ***ClientHelper***：如需要框架中的方法返回值给***JS***方法，可通过ClientHelper类来实现

技术架构

- 应用系统调用JS功能：应用系统通过***ClientHelper***实现对***js***功能的调用
- 应用系统调用HDF功能：应用系统可以调用框架集成的工具类、消息提示框、升级模块以及手机上常见的电话短信等功能

Activity的onCreate()方法加载页面

页面容器类

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    mEditText = (EditText) findViewById(R.id.text_edit);
    mButton = (Button) findViewById(R.id.btn_java);
    webview = (WebView) findViewById(R.id.webview);
    // 实例化接口JsInterface
    jsInterface = new JsInterface(webview);
    // 初始化WebSetting
    initWebSetting();
    webview.loadUrl("file:///android_asset/index.html");
}
```

定义接口

提供页面加载后支持

加载页面

初始化设置JS接口

页面加载接口等

```
private void initWebSetting() {
    // 允许JS交互
    webview.getSettings().setJavaScriptEnabled(true);
    // 设置JS的接口
    webview.addJavascriptInterface(jsInterface, "jsInterface");

    webview.setWebViewClient(onPageFinished(view, url) → {
            super.onPageFinished(view, url);
            mButton.setOnClickListener((v) → {
                    int param = Integer.parseInt(mEditText.getText().toString());
                    jsInterface.java_call_Js(param);
                    param = 0;
            });
    });
}
```

明确支持JS

设置JS的接口

页面加载接口-页面加载完成

应用系统调用JS功能

JS接口

```
public class JsInterface {
    private WebView mWebView;
    // 构造方法，传入一个参数WebView
    public JsInterface(WebView webView) { this.mWebView = webView; }
    // 这个方法是js调用java
    @JavascriptInterface
    public void js_call_java() {
        new Handler(Looper.getMainLooper()).post(() -> {
                // 主线程更新UI
                Toast.makeText(mWebView.getContext(), "现在调用的java的方法", Toast.LENGTH_SHORT).show();
        });
    }
    // 这个方法 是java调用js
    public void java_call_Js(int param) {
        // 这里调用html中的js代码的 java_call_Js 方法
        mWebView.loadUrl(String.format("javascript:java_call_Js(" + param + ")"));
    }
}
```

这是*Android*接口的具体方法，可直接调用框架中集成的功能，也可自定义功能

Java调用JS，通过*ClientHelper*实现对*js*功能的调用

Web页面

```
<html>
<head>
  <script language="javascript">
        function java_call_Js(param) {
            var x = document.getElementById('test');
            x.innerHTML = "这是js中的输出" + param;
        }
        function btn_js() {
            jsInterface.js_call_java();
        }
   </script>
</head>
<body>
<input type="button" onClick="btn_js()" style="height: 60px; width: 240px; margin-top: 40px ; margin-bottom: 10px"
       value="js调用Java">  </input>

<div id="test">
    <font color="#FF0000">这是已有的文本</font>
</div>
</body>
</html>
```

JS中定义的方法，Java通过***ClientHelper***实现对***JS***方法的调用

定义JS函数，调用接口中的Java方法

Web调用JS

运行结果


中国移动4G 10:12
1234567
JAVA调用JS
js调用Java
这是已有的文本


中国移动4G 10:13
1234567
JAVA调用JS
js调用Java
这是js中的输出1234567


中国移动4G 10:13
1234567
JAVA调用JS
js调用Java
这是js中的输出1234567
现在调用的java的方法

The End