HTML5实际开发过程如何应用。流程。设计交互图变成效果过程。

**Html5应用场景：**

1.PC端Web站点： 本地存储，渐强性体验，有局限性（处理兼容性）

2.基于webview的hybrid native的App开发： （原生APP--ios更新周期缓慢，需要审核，）更新灵活

3.Web App开发 ：微信HTML5广告，游戏，腾讯百度专门做HTML5网站，适配特殊浏览器使用。

**HTML5提供更好的语义标签**

<article></article>

<section></section>

<header></header>

<footer></footer>

<nav></nav>

**对API更好的支持**

Sessonstorage

Localstorage

Web Workers 交给后台进行 js单线程，想要异步操作 ，工作原理与setTimeout不同，必须与当前域名同域名的文件。所有与UI界面的代码不能放在文件中。

Canvas

Webgl

传统native APP 相比：

**快速的更新与迭代** Native 需要 更新，提交苹果市场 审核 进度慢

服务端更新

**想一次开发终端受益** 一台页面 嵌套在不同WebView下

**流畅的用户体验**

**HTML5现状，发展趋势：**

一次开发，多终端适配暂时不可以 多浏览器兼容性适配

Home menu back键都会影响体验

动画DOM操作比较差

手机端计算比PC端慢

原生性能快。

Html5 ：Media animated animated.css接口

动画卡暂时无法解决

**HTML5发展趋势**：无限趋近与native APP

Facebook 推出React 通过js编写逻辑，用来做构建界面。解决构建界面性能问题

**基础**：html/css/js (语法 对象 闭包 函数。。。) 服务器调试经验、如：fiddle扩展插件

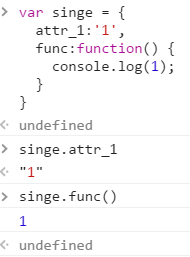
服务器调试：web :调试http:文本流。

**扩展**： html5 API CSS3

**JavaScript 设计模式:**

**单例模式： 对象**

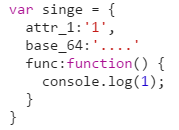
控制台：shift + enter 换行



**为什么用到单例模式：**

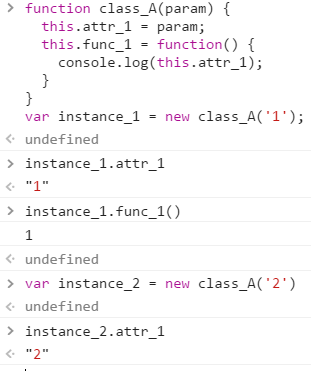
对象，属性，方法公用。不用多次实例化，创建对象，会占用很多内存。

Base64 10k-99k 100k-999k



**实例化的例子：**

Instance\_1是class\_A的实例

可以实例多个

**关于响应式布局：**

一套代码实现不同分辨率下CSS样式的适配

手机浏览器的阅读器：

1280 \* 800 平板

**准备工作：**

开发过程：1.对整个项目了解（html css3特性）

2.几个核心功能点分解（看交互图了解功能与地位）

3.项目设计图 分析核心功能 （顶部导航，底部导航，内容展示区，文本内容的排版---服务器传来的数据进行标签兑换，翻页功能（一页放慢内容，下一页进行数据加载，性能考虑））

4.所用的技术：按钮：css3 圆角，图标：base64格式图片，文本排版：Dom节点操作API，每次点进来保留之前的背景数据：html5的Local Storage。

**技术分解：**

PC端性能： <script>放在body，防止加载堵塞。

移动前端性能陷阱：

1. **减少或避免repaint,reflow 页面重绘与页面回流（减少对DOM元素的操作）**

重绘：元素位置信息大小发生变化。触发浏览器的repaint。相对reflow变化小

。Reflow 位置变，触发。 成本开销大。

PC：DOM元素属性操作

手机端：卡，避免

1. **尽量缓存所有可以缓存的数据**

PC ：http缓存 最大寿命：时间戳

 标识文件变化



总：

HTML5：**localStorage只能存字符串**

给对象序列化

localStorage.setItem('data\_1',JSON.stringify({a:1}))

localStorage.getItem('data\_1')

JSON.parse(localStorage.getItem('data\_1'))

indexedDB

1. **使用CSS3transform代替dom操作**
2. **不要给非static定位元素添加CSS3动画**
3. **适当使用硬件加速：手机设备GPU做渲染。用canvas画图，会触发硬件加速**

**transform:translate3d(0,0,0) 没任何相应，只为触发硬件加速**

**使用Base64格式制作ICON**

**优势：减少请求，加快首屏数据的显示速度**

平常：background:url....

url(data:image/png;base64,{imag\_data}) 不会是图片地址，是一个字符串

Base64格式图片生成：

通过base64图片编辑器，将图片拖进去，生成bse64格式

**缺点： 维护不方便base64导致图片大小比以前大 1/3; 而且浏览器还需要base64解码**

**使用CSS3制作ICON**

优点：体积更小

缺点：不容易维护，存在兼容性问题

用在Web App 几乎不用考虑兼容性

常用属性： border-radius box-shadow transform

使用范围：

规则图形，不复杂

**HTML5 API：更好完成性能的提升**

**DOM操作更好的支持 QuerySelector**

**本地的数据缓存 postMessage 跨域通信**

**硬件相关：**

**Window.performance.timing性能测试的api**

**传统的 new Date().getTime() 得到时间戳， 后面完成的时间戳相减**

**提升浏览器性能**

**cookit数据会发送到服务器端，浪费带宽**

**核心 HTML5 API**

遵循API调用规范

**可以跨域的Ajax请求 ：**

**之前**：jsonp iframe

<script src=”a.js”>

var xhr = new XMLHttpRequest();

xhr.onreadystatechange = function() {}

xhr,poen(url,’GET’,false); //异步

xhr.send()

Url：必须同域名 协议同，域名同

**现在：在服务器找那个设置服务器的响应头：**

**Java php:**

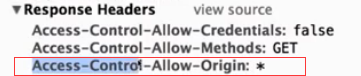
**header(‘Access-Control-Allow-Origin:\*’)**

**\*运行所有域都可以请求**

**限定特定域名：**

**header(‘Access-Control-Allow-Origin:www.baidu.com,www.qq.com’);代表运行腾讯百度跨域请求我们的接口**

**体现在：network**



**content Editable**

文本编辑器，div做成可编辑

<div contenteditable=”true”></div>

**让PC端触发触屏事件：**

var body = document.getElementsByTagName('body')

body = dody[0]

body.addEventListener('touchstart',function(){console.log('start')},false);

body.addEventListener('touchmove',function(){console.log('move')},false);

body.addEventListener('touchend’,function(){console.log('end')},false);

body.addEventListener('mousedown’,function(){console.log('mousedown')},false);

body.addEventListener('mouseup’,function(){console.log('mouseup')},false);

body.addEventListener('click’,function(){console.log('click')},false);

body.addEventListener('mousemove’,function(){console.log('mousemove')},false);

鼠标事件与触屏混用：

鼠标移动：start move end

点击地方时候： start click move end

触发mousedown 长按 start move end mousemove mouseup click

**统一交互：**

**viewport固定设备宽度**

**<meta name=”viewport” content=”width=device-width, initial-scale=1.0,maximum-scale=1.0，user-scalable=no”>**

**不能被缩放**

**开发流程：**

交互解读（用户，需求）-----设计解读------开发过程（25%）--------

交互图与需求分解：

**移动端开发技术选型的原则：**

1. **轻量化**
2. **快速开发**
3. **维护简单**
4. **高性能**

没有把js单独抽取出来成独立的文件，而是放在html文件中，优点：文件加载减少请求，性能方面加载速度提升。

缓存失效问题：base64的解码 jquery.base64.js , Jquery.jsonp.js 放在页面中

逻辑相关，数量不大的时候。

开发前准备：

阅读器项目规划中。阅读器js/css/html的文件放一起。

Jquery在移动端开发性能较差，所以使用了zepto

移动端：ul处理按钮