[**1.HTML5与移动web开发的背景** 2](#_Toc468304317)

[**2. HTML5的起始阶段** 2](#_Toc468304318)

[**3. HTML5的发展阶段** 2](#_Toc468304319)

[**4. HTML5的现状** 2](#_Toc468304320)

[**5. HTML5的应用** 2](#_Toc468304321)

[**5.1HTML5文件读取：** 3](#_Toc468304322)

[5.1.1FileList对象 3](#_Toc468304323)

[5.2.3FileReader对象 3](#_Toc468304324)

[5.2Web存储 3](#_Toc468304325)

[**5.2.3Web存储容量** 3](#_Toc468304326)

[**5.2.4数据共享** 3](#_Toc468304327)

[**5.2.5数据保存** 3](#_Toc468304328)

[**6.HTML5的新增属性** 3](#_Toc468304329)

[**6.1HTML5的离线缓存** 4](#_Toc468304330)

[**6.2HTML5的视频，音频的使用** 4](#_Toc468304331)

[**6.3HTML5的地理定位** 4](#_Toc468304332)

[**6.3.1获取地理信息的方式** 4](#_Toc468304333)

[**6. 3.2隐私** 4](#_Toc468304334)

[**6. 3.3 API详解** 4](#_Toc468304335)

[**6. 3.4获取地理信息失败** 4](#_Toc468304336)

[**6. 3.5可选参数** 4](#_Toc468304337)

[**6.移动web开发：** 4](#_Toc468304338)

[**6.1移动web开发常用的调试方法** 5](#_Toc468304339)

[**6.1.1模拟调试** 5](#_Toc468304340)

[6.2移动浏览器 5](#_Toc468304341)

[**6.2.1系统浏览器** 5](#_Toc468304342)

[6.2.2内置浏览器 5](#_Toc468304343)

[**6.2.3第三方浏览器** 6](#_Toc468304344)

[**6.3移动端的屏幕适配** 6](#_Toc468304345)

[**6.3.1ViewPort** 6](#_Toc468304346)

[**6.4控制缩放** 6](#_Toc468304347)

[**6.5避免滚动** 6](#_Toc468304348)

[6.3.4Mixin混入 7](#_Toc468304349)

HTML5与移动开发

前言：

在现今前端技术的发展越来越完善的情况下，前端在用户体验以及与后端的数据交互方面展示着越来越重要的作用。前端的职业出现的时间并不长，前端由最初的页面特效发展到现在的用户体验方面，数据交互，跨平台开发，同时也衍生出了众多的前端框架。

**1.HTML5与移动web开发的背景**

HTML5适用于所有的主流开发平台，不论是安卓还是IOS都有极好的兼容性。尤其是在移动端方面，现今的移动设备在硬件，软件，尤其是浏览器方面对于HTML5的各种特性都有极好的兼容性。甚至在实际的开发过程中移动端可以不用考虑兼容性问题，移动设备浏览器的更新速度及维护程度也远比PC端要好的多。HTML5包括HTML5+CSS3+Javascript,利用其独特的属性在现今的开发中真正的与后端独立，可以专注的与后端做数据交互，专注与用户体验，专注于和UI设计师的合作。

在移动web方面，在移动设备大量被使用的情况下，开发者和用户的关系变得越来越紧密。前端开发的工程师们所做的工作无疑都是在为用户服务。前端在用户体验，交互设计方面起着举足轻重的作用。前端的设计也决定了网站的使用量。移动开发尤其注重响应式，不同的设备不同的环境页面的适应性布局显得尤为重要。

1. **HTML5的起始阶段**

最初的前端也被成为是web前端，在web项目中，java工程师可以独立完成前后端的设计及代码的编写。但是相应的开发者的编码量势必会增加，除了需要处理后端复杂的数理逻辑之外还需兼顾前端页面的美观，用户体验，前后端的数据交互。同时随着互联网行业的发展，在用户体验方面的要求越来越高时，随着HTML5的发展，浏览器的兼容性的提高，移动设备的大量使用，前端逐渐与后端分离，独立的来完成页面的设计，着重解决用户体验以及前后端的数据交互，将在页面获取的数据直接传递给后台进行处理。

1. **HTML5的发展阶段**

HTML5的发展伴随着浏览器的发展。浏览器的发展使得HTML5的特性可以被随意使用。高版本的浏览器可以完美的实现HTML5的特性，完美的展现HTML5炫酷的页面效果。

1. **HTML5的现状**

现阶段的HTML5除了新增的属性，以及在css3以及javascript的共同使用下衍生出了众多优秀的前端框架，比如Bootstrap,amaze UI,Easy-UI。Bootstrap有极大的灵活性，良好的页面架构性，众多的组件。在web开发中总少不了后台页面的搭建，基于jQuery开发的Easy-UI可以方便网站后台页面的编写。Js方面一些流行的前端框架

1. **HTML5的应用**

现在的HTML5在各方面都有极为广泛的应用。移动APP的迅速发展使得HTML5的应用范围越来越广。HTML5的跨平台特性使其各方面都有极大的发展。

还有H5的微场景应用。微信小程序的应用。不论是web app，混合app还是原生app都离不开H5作为前端的交互。目前前端方面也用一些MVC的框架可以加速项目的开发。使用他人封装好的框架，利用其高度的继承性可以迅速开发需要的项目，同时也便于后期的维护。HTML5正建立起逐步完善的生态环境，就像现在的Java从最初产生到现在已经建立起了完整的生态圈。加之HTML5的开源性，吸引着众多的开发者。任何的网站，设备都需要页面展现在用户面前，所以不论后台技术如何发展，前端终究需要美观简洁的页面。随着技术的更新，前端的框架会越来越丰富，代码的集成性更高，同时开发过程中代码的耦合性也会被降到最低。

**5.1HTML5文件读取：**

通过FileReader对象我们可以读取本地存储的文件，可以使用 [File](https://developer.mozilla.org/zh-CN/docs/DOM/File) 对象来指定所要读取的文件或数据。File对象可以是来自用户在一个 [<input>](https://developer.mozilla.org/zh-CN/docs/HTML/Element/input) 元素上选择文件后返回的[FileList](https://developer.mozilla.org/zh-CN/docs/DOM/FileList) 对象。

### 5.1.1FileList对象

由于HTML5中我们可以通过为表单元素添加multiple属性，因此我们通过<input>上传文件后得到的是一个FileList对象（伪数组形式）。

### 5.2.3FileReader对象

HTML5新增内建对象，可以读取本地文件内容。利用var reader = new FileReader; 可以实例化一个对象

**5.2.3.1实例方法**

利用[readAsDataURL](#readAsDataURL())() 以DataURL形式读取文件

**5.2.3.2事件监听**

onload 当文读取完成时调用

**5.2.3.3属性**

result 文件读取结果

## 5.2Web存储

随着互联网的快速发展，基于网页的应用越来越普遍，同时也变的越来越复杂，为了满足各种各样的需求，会经常性在本地存储大量的数据，传统方式我们以document.cookie来进行存储的，但是由于其存储大小只有4k左右，并且解析也相当的复杂，每一次发送请求都会携带上cookie，会造成带宽的浪费，给开发带来诸多不便，HTML5规范则提出解决方案。

web存储的含义是将数据存储到用户的电脑上，这样可以缓解服务器的压力，并且提高体验,做好用户和移动端的交互。

**5.2.3Web存储容量**

同时web存储也有其自身良好的特性，信息设置，读取方便，同时容量较大，sessionStorage约5M,localStorage约20M,以此可以存储相应的信息。但其也有不足之处，web存储只能存储字符串，可以将对象JSON.stringify()编码后存储。

**5.2.4数据共享**

window.sessionStorage在同一个窗口可以数据共享，通过跳转即可实现。

**5.2.5数据保存**

window.localStorage保存的数据永久生效，除非进行手动删除。同时还可以多窗口共享。

**6.HTML5的新增属性**

HTML5的一些属性有些时候并不适用于Pc端，在PC端我们总要考虑兼容问题，写各种代码来处理兼容，在移动web开发上，HTML5完全可以大显身手。比如HTML5新增的表单属性可以帮助开发者做出漂亮的响应式的表单。

**6.1HTML5的离线缓存**

HTML5 Web Stroage API可以看作是加强版的cookie,可以不受数据的限制，可以将数据写入到本机的ROM中，还可以在浏览器关闭后再次打开时回复数据，以此来减少网络流量。同时其并不占用任何后台资源，从而减轻了硬件压力，增加设备运行的流畅性。

**6.2HTML5的视频，音频的使用**

最早的视频的制作是必须要由falsh来制作的，而且复杂并不容易实现。但是随着各大主流浏览器对于HTML5属性的支持，我们可以将一些主流格式的视频资源放在浏览器中观看，从而增加用户的体验性。

**6.3HTML5的地理定位**

随时分享位置，HTML5可以利用自身的特性进行地理定位，不需再下载大量的地图包。

**6.3.1获取地理信息的方式**

IP地址，三维坐标，GPS（Global Positioning System，全球定位系统），Wi-Fi，手机信号，用户自定义数据，浏览器会自动以最优方式获取信息。

**6. 3.2隐私**

HTML5 Geolocation 规范提供了一套保护用户隐私的机制。必须先得到用可，才能获取用户的位置信息。这样可以极大的保护用户的个人信息安全。

**6. 3.3 API详解**

使用navigator.geolocation.getCurrentPosition(successCallback, errorCallback, options) 获取当前地理信息，以及navigator.geolocation.watchPosition(successCallback, errorCallback, options) 重复获取当前地理信息。当成功获取地理信息后，会调用succssCallback，并返回一个包含位置信息的对象position。常用的表示方式有以下几种：position.coords.latitude纬度，position.coords.longitude经度，position.coords.accuracy精度，position.coords.altitude海拔高度。

**6. 3.4获取地理信息失败**

调用errorCallback，并返回错误信息error

**6. 3.5可选参数**

options 对象可以调整位置信息数据收集方式，分为

a)enableHighAccuracy 高精度模式。

b) timeout 超时设置，单位为ms。

c) maximumAge表示浏览器重新获取位置信息的时间间隔，单位为ms。

相对于PC端来讲，在开发的过程中总要考虑到各种浏览器的兼容性，尤其在特效方面浏览器各自的特性，以及其自身浏览器内核的影响，兼容性成为了首要考虑的问题。一些优秀的前端框架，不论是css3的还是JavaScript的都能为开发者提供良好的开发环境。

比如BootStrap,amaze UI在页面搭建方面为开发者提供了便于使用的工具。Bootarap是一款优秀的CSS框架，其在页面搭建方面良好的特性，诸多的特性，良好的交互性，便捷性吸引了众多的开发者。在响应式方面PC端，移动端都有良好的应用。在移动web开发上着重解决了不同尺寸的浏览器的兼容性问题。

**6.移动web开发：**

对于移动端来讲，现在大量的移动设备被用户使用，在使用的同时也会产生相应的急需解决的问题。我们知道，对于移动设备来讲其页面架构并不需要像PC端一样做的纷繁复杂，移动端的页面布局相对于PC端来讲要简单的多，在移动设备上开发人员只需将必要的，着重的产品展示给用户。面对各种不同尺寸的移动设备，那么一些流行的，便捷的CSS3的框架就成为首选。就像之前提到的Bootstrap，在移动端bootstrap良好的兼容性，尤其是其响应式的特性自然而然会成为首选。Bootstrap的各种组件为页面的搭建提供了诸多的便利。比如其自身的栅格布局，流式布局在开发的过程中可以节省大量的时间。再比如说辅之以less的使用，在编写页面的过程中避免了由于大量的css代码所造成的混乱，开发者可以自始至终对于css的页面架构有一个清晰的掌控，避免由于过多的css代码而陷入后期维护，测试的困难。

对于用户来讲，便捷迅速的来浏览网页就是最好的，在开发的过程中我们同样需要考虑移动设备的硬件体系终究要弱于PC设备，如何利用移动端现有的优势和劣势才去做好用户体验是首要考虑的问题。如果移动端的代码量太大，javascript代码量大造成页面加载缓慢，用户势必会直接退出浏览。所以在移动设备方面我们并不倡导使用一些代码集成量极大的框动架。例如像jquery,在pc端jquery的确是一款优秀的javascrip库，但是对于移动设备几十G的存储容量来讲无意是巨大的浪费。移动设备的页面结构并不复杂，我们在使用jquery的一部分属性方法时，其余的都是多余的，一方面浪费着移动设备的资源，另一方面指挥延迟网页的加载。所以从这样的角度来讲不如去考虑着重javascript原生代码的编写，从开发者的角度来讲用自己的方法封装一些必要的属性方法，使用时直接调用要好的多。同时也便于后期的维护和改版。

**6.1移动web开发常用的调试方法**

一般移动开发方面的调试方式有两种，分为模拟调试和真机调试。

**6.1.1模拟调试**

现在的主流浏览器都支持移动开发的调试，点击F12方式即可进行简单调试，比如火狐，谷歌浏览器都自带各种版本的主流移动设备的模拟器。开发者可以通过调节不同的尺寸来测试移动页面是否适用。

**6.1.2真机调试**

虽然主流的浏览器会支持各种版本的移动的开发但是开发环境复杂，一些移动设备无法适用，所以必要时需要进行真机测试。真机测试有以下两种方式。将做好的网页上传至服务器或者本地搭建服务器，然后移动设备通过网络来访问。借助第三方的调试工具，如weinre、debuggap、ghostlab等。需注意的是真机调试必须保证移动设备同服务器间的网络是相通的。

## 6.2移动浏览器

移动端开发主要是针对IOS和Android两个操作系统平台的，除此之外还有Windows Phone。移动端主要可以分成三大类，系统自带浏览器、应用内置浏览器、第三方浏览器

**6.2.1系统浏览器**

系统浏览器：指跟随移动设备操作系统一起安装的浏览器。

### 6.2.2内置浏览器

应用内置浏览器：通常在移动设备上都会安装一些APP例如QQ、微信、微博、淘宝等，这些APP里往往会内置一个浏览器，我们称这个浏览器为应用内置浏览器（也叫WebView），这个内置的浏览器一般功能比较简单，并且客户端开发人员可以更改这个浏览器的某些设置，在我们理实的开发里这个浏览器很重要。

**6.2.3第三方浏览器**

第三方浏览器：指安装在手机的浏览器如FireFox、Chrome、360等等。在IOS和Android操作系统上自带浏览器、应用内置浏览器都是基础Webkit内核的。

**6.3移动端的屏幕适配**

经过分析我们得到，移动页面最理想的状态是，避免滚动条且不被默认缩放处理，我们可以通过设置<meta name="viewport" content="">来进行控制，并改变浏览器默认的layout viewport的宽度。

**6.3.1ViewPort**

viewport 是由苹果公司为了解决移动设备浏览器渲染页面而提出的解决方案，后来被其它移动设备厂商采纳。通过设置属性content=""实现，中间以逗号分隔例如：

<meta name="viewport" content="width=device-width">

width 设置layout viewport 宽度，其取值可为数值或者 device-width。

height 设置layout viewport 高度，其取值可为数值或者device-height

initital-scale 设置页面的初始缩放值，为一个数字，可以带小数。

maximum-scale 允许用户的最大缩放值，为一个数字，可以带小数。

minimum-scale 允许用户的最小缩放值，为一个数字，可以带小数。

user-scalable 是否允许用户进行缩放，值为"no"或"yes"。

在这儿注意device-width 和 device-height 实为 ideal viewport 的宽高。

**6.4控制缩放**

设置<meta name="viewport" content="initial-scale=1">，这时我们发现网页没有被浏览器设置缩放。设置<meta name="viewport" content="width=device-width">，这时我们发现网页也没有被浏览器设设置缩放。当我们设置width=device-width，也达到了initial-scale=1的效果，得知其实 initial-scale = ideal viewport / layout viewport。虽然我们能过设置<meta name="viewport" content="">，可以控制网页的默认缩放，但是会出现滚动条。

**6.5避免滚动**

滚动条是layout viewport相对于ideal viewpor的，所以只要设置layout viewport小于或等于ideal viewport，即<meta name="viewport" content="width=device-width">。

经测试发现我们并没有完全的解决滚动条的问题，原因在于我们示例里的.box {width: 490px;}设置了一个绝对的宽度造成的，要解决这个问题需要设置一个百分比（100%）的宽度。这样就可以有效避免屏幕滚动的问题。

**6.6LESS**

CSS（层叠样式表）是一门标记性语言，同 HTML 一道被广泛应用于万维网（World Wide Web）中。HTML 主要负责文档结构的定义，CSS 负责文档表现形式或样式的定义。

作为一门标记性语言，CSS 的语法相对简单，对使用者的要求较低，但同时也带来一些问题：CSS 需要书写大量看似没有逻辑的代码，不方便维护及扩展，不利于复用，尤其对于非前端开发工程师来讲，往往会因为缺少 CSS 编写经验而很难写出组织良好且易于维护的 CSS 代码，造成这些困难的很大原因源于 CSS 是一门非程序式语言，没有变量、函数、SCOPE（作用域）等概念。LESS 在 CSS 的语法基础之上，引入了变量，Mixin（混入），运算以及函数等功能，简化了 CSS 的编写，并且降低了 CSS 的维护成本，就像它的名称所说的那样，LESS 可以让我们用更少的代码做更多的事情。

**6.3.1less的原理及使用方式**

LESS 包含一套自定义的语法及一个解析器，用户根据这些语法定义自己的样式规则，这些规则最终会通过解析器，编译生成对应的 CSS 文件。LESS 并没有裁剪 CSS 原有的特性，更不是用来取代 CSS 的，而是在现有 CSS 语法的基础上，为 CSS 加入程序式语言的特性。Less的使用使得CSS在后期的维护上变得简易，且在开发的过程中不会出错。

**6.3.2less语法**

Less有以下常用的语法：变量，Mixin混入，嵌套，Import，函数（内置函数）。

**6.3.3less变量**

LESS 允许开发者自定义变量，变量可以在全局样式中使用，变量使得样式修改起来 更加简单。比如说：

@mainColor:#E93223;  
 body{  
 color: @mainColor;  
 }

我们可以将color的颜色预先定义，在使用的时直接调用。

### 6.3.4Mixin混入

很多动态语言都支持 Mixin（混入）特 性，它是多重继承的一种实现，在 LESS 中，混入是指在一个 CLASS 中引入另外一个 已经定义的 CLASS，就像在当前 CLASS 中增加一个属性一样。

**6.3.5LESS嵌套**

在我们书写标准 CSS 的时候，遇到多层的元素嵌套这种情况时，我们要么采用从外到内的选择器嵌套定义，要么采用给特定元素 CLASS 或 ID 的方式，但是如果内容太多的话，不便于查找相应的css样式。同一个项目如果css样式太多必然造成混乱，开发者自身在阅读代码时也会发生错误。使用嵌套的方式书写CSS样式可以清晰的了解css代码的结构，而且不会产生嵌套错误。比如以下的方式：

**#wjs\_banner**{  
 .**carousel-inner**{  
 > **div**.**item**{  
 **a**.**img\_box**{  
 background: **url**(**"../images/slide\_01\_2000x410.jpg"**) **no-repeat center center**;  
 height: 410**px**;  
 */\*调用redBorder mixin\*/* display: **block**;  
 .**redBorder**();  
 */\*调用@mainColor 变量\*/* &:**hover**{  
 color: @mainColor;  
 }  
 }  
 **a**.**img\_mobile**{  
 width: 100%;  
 display: **block**;  
 **img**{  
 width: 100%;  
 display: **block**;  
 }  
 }  
 }  
 }  
}