**webkit webApp 开发技术要点总结**

如果你是一名前端er,又想在移动设备上开发出自己的应用，那怎么实现呢？幸好，webkit内核的浏览器能帮助我们完成这一切。接触 webkit webApp的开发已经有一段时间了，现把一些技巧分享给大家 ：

**1. viewport：**  
也就是可视区域。对于桌面浏览器，我们都很清楚viewport是什么，就是出去了所有工具栏、状态栏、滚动条等等之后用于看网页的区域，  
这是真正有效的区域。由于移动设备屏幕宽度不同于传统web,因此我们需要改变viewport;

实际上我们可以操作的属性有4 个：

|  |
| --- |
| width -             //  viewport 的宽度 （范围从200 到10,000，默认为980 像素）  height -            //  viewport 的高度 （范围从223 到10,000）    initial-scale -     //  初始的缩放比例 （范围从>0 到10）    minimum-scale -    //   允许用户缩放到的最小比例  maximum-scale -    //   允许用户缩放到的最大比例    user-scalable -    //   用户是否可以手动缩 (no,yes) |

那么到底这些设置如何让Safari 知道？其实很简单，就一个meta，形如：

|  |
| --- |
| <meta http-equiv="Content-Type" content="text/html; charset=utf-8">   //编码  <meta id="viewport" name="viewport" content="width=320; initial-scale=1.0;maximum-scale=1.0; user-scalable=no;"/>  <meta name=”apple-mobile-web-app-capable” content=”yes” />  // 离线应用的另一个技巧  <meta name=”apple-mobile-web-app-status-bar-style” content=black” />  // 隐藏状态栏  <meta content="black" name="apple-mobile-web-app-status-bar-style" /> //指定的iphone中safari顶端的状态条的样式  <meta content="telephone=no" name="format-detection" />       //告诉设备忽略将页面中的数字识别为电话号码  <meta name="Author" contect="Mr.He"/ > |

在设置了initial-scale=1 之后，我们终于可以以1:1 的比例进行页面设计了。关于viewport，还有一个很重要的概念是：iphone 的safari 浏览器完全没有滚动条，而且不是简单的“隐藏滚动条”，是根本没有这个功能。iphone 的safari 浏览器实际上从一开始就完整显示了这个网页，然后用viewport 查看其中的一部分。当你用手指拖动时，其实拖的不是页面，而是viewport。浏览器行为的改变不止是滚动条，交互事件也跟普通桌面不一样。(请参考：[指尖的下JS](http://www.cnblogs.com/pifoo/archive/2011/05/23/webkit-touch-event-1.html) 系列文章)  
  
**2. link:**

|  |
| --- |
| <link rel=”apple-touch-startup-image” href=”startup.png” /> // 设置开始页面图片  <link rel=”apple-touch-icon” href=”iphon\_tetris\_icon.png”/> // 在设置书签的时候可以显示好看的图标  <link rel="stylesheet" media="all and (orientation:portrait)" href="portrait.css">    // 肖像模式样式  <link rel="stylesheet" media="all and (orientation:landscape)" href="landscape.css"   // 风景模式样式    //竖屏时使用的样式  <style media="all and (orientation:portrait)" type="text/css">  #landscape { display: none; }  </style>    //横屏时使用的样式  <style media="all and (orientation:landscape)" type="text/css">  #portrait { display: none; }  </style> |

**3. 事件 ： (**请参考：[指尖的下JS](http://www.cnblogs.com/pifoo/archive/2011/05/23/webkit-touch-event-1.html) 系列文章)

|  |
| --- |
| // 手势事件  touchstart            //当手指接触屏幕时触发  touchmove           //当已经接触屏幕的手指开始移动后触发  touchend             //当手指离开屏幕时触发  touchcancel    // 触摸事件  gesturestart          //当两个手指接触屏幕时触发  gesturechange      //当两个手指接触屏幕后开始移动时触发  gestureend    // 屏幕旋转事件  onorientationchange    // 检测触摸屏幕的手指何时改变方向  orientationchange    // touch事件支持的相关属性  touches  targetTouches  changedTouches  clientX　　　　// X coordinate of touch relative to the viewport (excludes scroll offset)  clientY　　　　// Y coordinate of touch relative to the viewport (excludes scroll offset)  screenX　　　 // Relative to the screen  screenY 　　  // Relative to the screen  pageX　　 　　// Relative to the full page (includes scrolling)  pageY　　　　 // Relative to the full page (includes scrolling)  target　　　　 // Node the touch event originated from  identifier　　   // An identifying number, unique to each touch event |

**4. 屏幕旋转事件**：onorientationchange  
添加屏幕旋转事件侦听，可随时发现屏幕旋转状态（左旋、右旋还是没旋）。例子：

|  |
| --- |
| // 判断屏幕是否旋转  function orientationChange() {      switch(window.orientation) {      　　case 0:              alert("肖像模式 0,screen-width: " + screen.width + "; screen-height:" + screen.height);              break;      　　case -90:              alert("左旋 -90,screen-width: " + screen.width + "; screen-height:" + screen.height);              break;      　　case 90:              alert("右旋 90,screen-width: " + screen.width + "; screen-height:" + screen.height);              break;      　　case 180:          　　alert("风景模式 180,screen-width: " + screen.width + "; screen-height:" + screen.height);          　　break;      };<br>};  // 添加事件监听  addEventListener('load', function(){      orientationChange();      window.onorientationchange = orientationChange;  }); |

**5. 隐藏地址栏 & 处理事件的时候，防止滚动条出现：**

|  |
| --- |
| // 隐藏地址栏  & 处理事件的时候 ，防止滚动条出现  addEventListener('load', function(){          setTimeout(function(){ window.scrollTo(0, 1); }, 100);  }); |

**6. 双手指滑动事件：**

|  |
| --- |
| // 双手指滑动事件  addEventListener('load',　　function(){ window.onmousewheel = twoFingerScroll;},       false              // 兼容各浏览器，表示在冒泡阶段调用事件处理程序 (true 捕获阶段)  );  function twoFingerScroll(ev) {      var delta =ev.wheelDelta/120;              //对 delta 值进行判断(比如正负) ，而后执行相应操作      return true;  }; |

**7. 判断是否为iPhone：**

|  |
| --- |
| // 判断是否为 iPhone ：  function isAppleMobile() {      return (navigator.platform.indexOf('iPad') != -1);  }; |

**8. localStorage:**  
 例子 ：（注意数据名称  n  要用引号引起来）

|  |
| --- |
| var v = localStorage.getItem('n') ? localStorage.getItem('n') : "";   // 如果名称是  n 的数据存在 ，则将其读出 ，赋予变量  v  。  localStorage.setItem('n', v);                                           // 写入名称为 n、值为  v  的数据  localStorage.removeItem('n');                                           // 删除名称为  n  的数据 |

**9. 使用特殊链接：**  
 如果你关闭自动识别后 ，又希望某些电话号码能够链接到 iPhone 的拨号功能 ，那么可以通过这样来声明电话链接 ,

|  |
| --- |
| <a href="tel:12345654321">打电话给我</a>  <a href="sms:12345654321">发短信</a>  或用于单元格：  <td onclick="location.href='tel:122'"> |

**10. 自动大写与自动修正**  
要关闭这两项功能，可以通过autocapitalize 与autocorrect 这两个选项：

|  |
| --- |
| <input type="text" autocapitalize="off" autocorrect="off" /> |

**11. WebKit CSS:**  
①“盒模型”的具体描述性质的包围盒块内容，包括边界，填充等等。

|  |
| --- |
| -webkit-border-bottom-left-radius: radius;  -webkit-border-top-left-radius: horizontal\_radius vertical\_radius;  -webkit-border-radius: radius;      //容器圆角  -webkit-box-sizing: sizing\_model; 边框常量值：border-box/content-box  -webkit-box-shadow: hoff voff blur color; //容器阴影（参数分别为：水平X 方向偏移量；垂直Y 方向偏移量；高斯模糊半径值；阴影颜色值）  -webkit-margin-bottom-collapse: collapse\_behavior; 常量值：collapse/discard/separate  -webkit-margin-start: width;  -webkit-padding-start: width;  -webkit-border-image: url(borderimg.gif) 25 25 25 25 round/stretch round/stretch;  -webkit-appearance: push-button;   //内置的CSS 表现，暂时只支持push-button |

②“视觉格式化模型”描述性质，确定了位置和大小的块元素。

|  |
| --- |
| direction: rtl  unicode-bidi: bidi-override; 常量：bidi-override/embed/normal |

③“视觉效果”描述属性，调整的视觉效果块内容，包括溢出行为，调整行为，能见度，动画，变换，和过渡。

|  |
| --- |
| clip: rect(10px, 5px, 10px, 5px)  resize: auto; 常量：auto/both/horizontal/none/vertical  visibility: visible; 常量: collapse/hidden/visible  -webkit-transition: opacity 1s linear; 动画效果 ease/linear/ease-in/ease-out/ease-in-out  -webkit-backface-visibility: visibler; 常量：visible(默认值)/hidden  -webkit-box-reflect: right 1px; 镜向反转  -webkit-box-reflect: below 4px -webkit-gradient(linear, left top, left bottom,  from(transparent), color-stop(0.5, transparent), to(white));  -webkit-mask-image: -webkit-gradient(linear, left top, left bottom, from(rgba(0,0,0,1)), to(rgba(0,0,0,0)));;   //CSS 遮罩/蒙板效果  -webkit-mask-attachment: fixed; 常量：fixed/scroll  -webkit-perspective: value; 常量：none(默认)  -webkit-perspective-origin: left top;  -webkit-transform: rotate(5deg);  -webkit-transform-style: preserve-3d; 常量：flat/preserve-3d; (2D 与3D) |

④“生成的内容，自动编号，并列出”描述属性，允许您更改内容的一个组成部分，创建自动编号的章节和标题，和操纵的风格清单的内容。

|  |
| --- |
| content: “Item” counter(section) ” “;  This resets the counter.  First section  >two section  three section  counter-increment: section 1;  counter-reset: section; |

⑤“分页媒体”描述性能与外观的属性，控制印刷版本的网页，如分页符的行为。

|  |
| --- |
| page-break-after: auto; 常量：always/auto/avoid/left/right  page-break-before: auto; 常量：always/auto/avoid/left/right  page-break-inside: auto; 常量：auto/avoid |

⑥“颜色和背景”描述属性控制背景下的块级元素和颜色的文本内容的组成部分。

|  |
| --- |
| -webkit-background-clip: content; 常量：border/content/padding/text  -webkit-background-origin: padding; 常量：border/content/padding/text  -webkit-background-size: 55px; 常量：length/length\_x/length\_y |

⑦ “字型”的具体描述性质的文字字体的选择范围内的一个因素。报告还描述属性用于下载字体定义。

|  |
| --- |
| unicode-range: U+00-FF, U+980-9FF; |

⑧“文本”描述属性的特定文字样式，间距和自动滚屏。

|  |
| --- |
| text-shadow: #00FFFC 10px 10px 5px;  text-transform: capitalize; 常量：capitalize/lowercase/none/uppercase  word-wrap: break-word; 常量：break-word/normal  -webkit-marquee: right large infinite normal 10s; 常量：direction(方向) increment(迭代次数) repetition(重复) style(样式) speed(速度);  -webkit-marquee-direction: ahead/auto/backwards/down/forwards/left/reverse/right/up  -webkit-marquee-incrementt: 1-n/infinite(无穷次)  -webkit-marquee-speed: fast/normal/slow  -webkit-marquee-style: alternate/none/scroll/slide  -webkit-text-fill-color: #ff6600; 常量：capitalize, lowercase, none, uppercase  -webkit-text-security: circle; 常量：circle/disc/none/square  -webkit-text-size-adjust: none; 常量:auto/none;  -webkit-text-stroke: 15px #fff;  -webkit-line-break: after-white-space; 常量：normal/after-white-space  -webkit-appearance: caps-lock-indicator;  -webkit-nbsp-mode: space; 常量： normal/space  -webkit-rtl-ordering: logical; 常量：visual/logical  -webkit-user-drag: element; 常量：element/auto/none  -webkit-user-modify: read- only; 常量：read-write-plaintext-only/read-write/read-only  -webkit-user-select: text; 常量：text/auto/none |

⑨“表格”描述的布局和设计性能表的具体内容。

|  |
| --- |
| -webkit-border-horizontal-spacing: 2px;  -webkit-border-vertical-spacing: 2px;  -webkit-column-break-after: right; 常量：always/auto/avoid/left/right  -webkit-column-break-before: right; 常量：always/auto/avoid/left/right  –webkit-column-break-inside: logical; 常量：avoid/auto  -webkit-column-count: 3; //分栏  -webkit-column-rule: 1px solid #fff;  style:dashed,dotted,double,groove,hidden,inset,none,outset,ridge,solid |

⑩“用户界面”描述属性，涉及到用户界面元素在浏览器中，如滚动文字区，滚动条，等等。报告还描述属性，范围以外的网页内容，如光标的标注样式和显示当您按住触摸触摸  
目标，如在iPhone上的链接。

|  |
| --- |
| -webkit-box-align: baseline,center,end,start,stretch 常量：baseline/center/end/start/stretch  -webkit-box-direction: normal;常量：normal/reverse  -webkit-box-flex: flex\_valuet  -webkit-box-flex-group: group\_number  -webkit-box-lines: multiple; 常量：multiple/single  -webkit-box-ordinal-group: group\_number  -webkit-box-orient: block-axis; 常量：block-axis/horizontal/inline-axis/vertical/orientation  –webkit-box-pack: alignment; 常量：center/end/justify/start |

**12. 动画过渡**  
这是 Webkit 中最具创新力的特性：使用过渡函数定义动画。

|  |
| --- |
| -webkit-animation: title infinite ease-in-out 3s;  animation 有这几个属性：  -webkit-animation-name： //属性名，就是我们定义的keyframes  -webkit-animation-duration：3s //持续时间  -webkit-animation-timing-function： //过渡类型：ease/ linear(线性) /ease-in(慢到快)/ease-out(快到慢) /ease-in-out(慢到快再到慢) /cubic-bezier  -webkit-animation-delay：10ms //动画延迟(默认0)  -webkit-animation-iteration-count： //循环次数(默认1)，infinite 为无限  -webkit-animation-direction： //动画方式：normal(默认 正向播放)； alternate(交替方向，第偶数次正向播放，第奇数次反向播放) |

这些同样是可以简写的。但真正让我觉的很爽的是keyframes，它能定义一个动画的转变过程供调用，过程为0%到100%或from(0%)到to(100%)。简单点说，只要你有想法，你想让元素在这个过程中以什么样的方式改变都是很简单的。

|  |
| --- |
| -webkit-transform: 类型（缩放scale/旋转rotate/倾斜skew/位移translate）  scale(num,num) 放大倍率。scaleX 和 scaleY(3)，可以简写为：scale(\* , \*)  rotate(\*deg) 转动角度。rotateX 和 rotateY，可以简写为：rotate(\* , \*)  Skew(\*deg) 倾斜角度。skewX 和skewY，可简写为：skew(\* , \*)  translate(\*,\*) 坐标移动。translateX 和translateY，可简写为：translate(\* , \*)。 |

实现模拟弹出消息框（Alert）的例子：  
①定义过渡（在<style type="text/css">段中描述keyframes）：

|  |
| --- |
| @-webkit-keyframes DivZoom  {  0% { -webkit-transform: scale(0.01) }  60% { -webkit-transform: scale(1.05) }  80% { -webkit-transform: scale(0.95) }  100% { -webkit-transform: scale(1.00) }  }  .sZoom { -webkit-animation: DivZoom 0.5s ease-in-out } |

（很容易看懂，将元素从缩小的0.01 倍--很小但不能为0 倍，放大到1.05 倍，再缩小到0.95倍，最后到1 倍即正常大小。整个过渡过程事件为0.5 秒，动画方式为ease-in-out，即慢到快再到慢，默认只进行1 次过渡。这正是大家经常看到的 iPhone 弹出的提示信息的动画效果！）  
②定义元素（在<body>段中）：

|  |
| --- |
| <div id="layerH" style="-webkit-border-radius:12px; border:2px solid #FFF;-webkit-box-shadow: 0px 2px 4px #888;position: absolute; left: 24px; top: 106px;<br>width: 256px; height: 268px; padding-left: 8px; padding-right: 8px;color: #FFFFFF; text-shadow: 1px 1px 1px #000; text-align: center;background-color: RGBA(32,48,96,0.9);  background-image:url('BG-Msg.png'); background-repeat:no-repeat;  z-index: 1; visibility: hidden; ">  <p><span style="font-size: 16pt; font-weight: bold">使用说明</span></p>  <hr noshade size="1">  <div id="HelpText" style="height: 120px">说明文字</div>  <hr noshade size="1">  <form name="formV" method="POST">  <input type="button" value="确认" name="B1"  style="width: 100%; height: 40px; font-size: 14pt; ont-weight: bold;  color: #FFFFFF; text-shadow: 0px -1px 1px #000;"  onclick=" layerH.style.visibility='hidden'">  </form>  </div> |

③启动动画（在 javascript 定义的函数中）

|  |
| --- |
| function pHelp()  {  layerH.style.visibility = 'visible'  layerH.style.cssText = "-webkit-animation-delay: " + Math.random() + "ms"  layerH.className = 'sZoom'  } |

(这个启动函数就很好理解了。但是为什么要使用-webkit-animation-delay 这句呢？因为当一个元素过渡显示完成后，若其样式没有变化，下一次将无法进行过渡动画显示。我们巧妙的利用其动画延迟时间定义，使其有所变化，就避免了上述问题。其中使用随机数函数Math.random()，产生一个大于0 小于1 的随机数。当然，延迟零点几毫秒，用户是不会察觉的。)  
  
**补充：**  
1. 锁定 viewport

|  |
| --- |
| ontouchmove="event.preventDefault()" //锁定viewport，任何屏幕操作不移动用户界面（弹出键盘除外）。 |

2. 被点击元素的外观变化，可以使用样式来设定：

|  |
| --- |
| -webkit-tap-highlight-color: 颜色 |

3. 侦测iPhone/iPod  
开发特定设备的移动网站，首先要做的就是设备侦测了。下面是使用Javascript侦测iPhone/iPod的UA，然后转向到专属的URL。

|  |
| --- |
| if((navigator.userAgent.match(/iPhone/i)) || (navigator.userAgent.match(/iPod/i))) {  　　if (document.cookie.indexOf("iphone\_redirect=false") == -1) {  　　　　window.location = "http://m.example.com";  　　}  } |

虽然Javascript是可以在水果设备上运行的，但是用户还是可以禁用。它也会造成客户端刷新和额外的数据传输，所以下面是服务器端侦测和转向：

|  |
| --- |
| if(strstr($\_SERVER['HTTP\_USER\_AGENT'],'iPhone') || strstr($\_SERVER['HTTP\_USER\_AGENT'],'iPod')) {  　　header('Location: http://yoursite.com/iphone');  　　exit();  } |

4. 阻止旋转屏幕时自动调整字体大小

|  |
| --- |
| html, body, form, fieldset, p, div, h1, h2, h3, h4, h5, h6 {-webkit-text-size-adjust:none;} |

5. iPhone才识别的CSS  
如果不想设备侦测，可以用CSS媒体查询来专为iPhone/iPod定义样式。

|  |
| --- |
| @media screen and (max-device-width: 480px) {} |

6. 缩小图片  
网站的大图通常宽度都超过480像素，如果用前面的代码限制了缩放，这些图片在iPhone版显示显然会超过屏幕。好在iPhone机能还够，我们可以用CSS让iPhone自动将大图片缩小显示。

|  |
| --- |
| @media screen and (max-device-width: 480px){  　　img{max-width:100%;height:auto;}  } |

7. 模拟:hover伪类  
因为iPhone并没有鼠标指针，所以没有hover事件。那么CSS :hover伪类就没用了。但是iPhone有Touch事件，onTouchStart 类似 onMouseOver，onTouchEnd 类似 onMouseOut。所以我们可以用它来模拟hover。使用Javascript：

|  |
| --- |
| var myLinks = document.getElementsByTagName('a');  for(var i = 0; i < myLinks.length; i++){  　　myLinks[i].addEventListener(’touchstart’, function(){this.className = “hover”;}, false);  　　myLinks[i].addEventListener(’touchend’, function(){this.className = “”;}, false);  } |

然后用CSS增加hover效果：

|  |
| --- |
| a:hover, a.hover { /\* 你的hover效果 \*/ } |

这样设计一个链接，感觉可以更像按钮。并且，这个模拟可以用在任何元素上。