第一章 移动页面布局

第一节 像素的基础知识

- 移动项目分两类：（1）响应式布局 （2）移动端布局;
  - 响应式（媒体查询）
  - 移动项目：不需兼容pc设备

  注：有些公司，pc版和手机版会分别做
- 像素的基础知识

  iphone6 宽度为： 375*667，为什么分辨率为 750*1334
  - px css pixels 逻辑像素 ，浏览器使用的抽象单位;
  - dp,pt device independent pixels 设备无关像素
  - dpr devicePIxelRatio 设备像素比

  当dpr=2时，

  平面上： 1px= Math.pow(dpr,2) pt;

  在水平或垂直方向: 1px = 2 pt(dp)

  所以对于ipone6来说 375px = 750 pt // 分辨率是以pt(dp)为单位的。
- dpi 与 ppi
  - dpi 打印机每英寸可以打印的墨汁点（印刷行业）;
  - ppi 屏幕每英寸的像素数量，即单位英寸内的像素密度;

    以iphone6为例
    ppi = √ (1334^2+750^2) / 4.7 = 326 ppi; (视网膜Retina屏)

    //计算ppi使用的是物理像素
  - 结论：ppi越高，屏幕的清晰度会越高;

第二节 viewport 基础知识

- pc网页在移动设备上的显示效果
- layout viewport (布局视口)
  - 排版需要：将页面布局在一个比较大的viewport内，ios一般为980px，保证排版准确;
  - 显示需要：移动设备屏幕比较小，对viewport进行缩放就可看到网页全貌;
  - document.body.clientWidth 来获取
- visual viewport(可视视口)
  - 无论pc还是移动设备，看到的浏览器窗口就是visual viewport
  - 可以通过window.innerWidth/height来获取
- 调整visual viewport 和 layoutview之间的距离来放大和缩小内容;

可比作相机的屏幕，当调整焦距的时候，画面中的内容大小会变化;

- demo范例:
  - 宽度为375px的元素在ipone6下的效果
  - 增加veiwport 设置后的效果(width=375, 将布局宽度调制为375px)
  - 设备的尺寸不同，所以可设置为 width=device-width;
  - 不同设备下，window.innerWidth大小会变化, 所以需要增加 initial-scale = 1;
  - 不容许用户缩放： user-scalable=no

第三节 meta标签设置viewport

```
布局视口:布局视口：设备中显示网页的区域，可能比浏览器视口大，也可能小，在手机中一般为980px;
<meta name="viewport" content="width='device-width,initial-scale=1,user-scalable=no' " />
device-width 设备宽度
```

width	设置***layout viewport*** 的宽度，为一个正整数，或字符串**"device-width"**
initial-scale	设置页面的初始缩放值，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
maximum-scale	允许用户的最大缩放值，为一个数字，可以带小数
height	设置***layout viewport*** 的高度，这个属性对我们并不重要，很少使用
user-scalable	是否允许用户进行缩放，值为"no"或"yes", no 代表不允许，yes代表允许

- 项目调试时清缓存的几个meta标签（上线时去掉）

```
<meta http-equiv="Pragma" content="no-cache">
<meta http-equiv="Cache-Control" content="no-cache">
<meta http-equiv="Expires" content="0">
```

第四节 rem知识

- em % 相对于父级元素
- rem 相对于根元素或屏幕尺寸 font-size of the root element
- vw vh viewport width / height 相对于浏览器窗口的百分比

vmin vmax 屏幕的较小或较大的一个

第五节 移动适配

- 什么是适配

> 在不同尺寸的手机设备上，页面“相对性的达到合理的展示（自适应）

> ”或者“ 保持统一效果的等比缩放（看起来差不多）”。

- 适配要素

> 一般来说，我们需要关注的是：字体、高宽、间距、图像（图标、图片），
>
> 我们可以将图片设定为父容器的100%（背景使用background-size），这样只针对元素进行布局

- 适配方法
  - 百分比尺寸+rem
  - media媒体查询

> 设计稿的尺寸
>
> - iphone 5　640　640=6.4*100
> - iphone6　750　750=7.5*100
>
> 当设备窗口 大于750 ， 只显示750
>
> 当设备窗口小于750 ， 按一个比率缩小;

第二章 弹性盒模型

参考网址 ： http://blog.csdn.net/mr_lp/article/details/50966842

弹性盒模型

传统布局方式 ： table 布局----> div+css+html (浮动,定位,清浮动) ----> 弹性盒模型

主轴(main axis) -- main start, main end;

侧轴(cross axis) -- corss start corss end;

注意：主轴或侧轴的概念是相对的，在于项目的排列方向;

display:flex

display:inline-flex; 弹性盒模型的父容器被置为行块元素;

注意，设为Flex布局以后，子元素的 float、clear 和 vertical-align 属性将失效。

第一节 主轴上的属性

- flex-direction　主轴的方向
- flex-wrap
- flex-flow
- justify-content
- align-items
- align-content

- flex-direction　主轴方向

row（默认值）	主轴为水平方向，起点在左端。
row-reverse	主轴为水平方向，起点在右端。
column	主轴为垂直方向，起点在上沿。
column-reverse	主轴为垂直方向，起点在下沿。

- flex-wrap　如果一条轴线排不下，如何换行。
  - no-wrap 不换行 (默认)
  - wrap　正常换行
  - wrap-reverse　反向换行

- justify-content　项目在主轴上的对齐方式

flex-start（默认值）	左对齐
flex-end	右对齐
center	居中
space-between	两端对齐，项目之间的间隔都相等。
space-around	每个项目两侧的间隔相等。所以，项目之间的间隔比项目与边框的间隔大一倍。

- align-items 项目在交叉轴上的对齐方式

flex-start	交叉轴的起点对齐。
flex-end	交叉轴的终点对齐。
center	交叉轴的中点对齐。
baseline	项目的第一行文字的基线对齐。
stretch（默认值）	如果项目未设置高度或设为auto，将占满整个容器的高度。

- align-content属性

align-content属性定义了多根轴线的对齐方式。如果项目只有一根轴线，该属性不起作用。

flex-start	与交叉轴的起点对齐。
flex-end	与交叉轴的终点对齐。
center	与交叉轴的中点对齐。
space-between	与交叉轴两端对齐，轴线之间的间隔平均分布。
space-around	每根轴线两侧的间隔都相等。所以，轴线之间的间隔比轴线与边框的间隔大一倍。
stretch（默认值）	轴线占满整个交叉轴。

第二节 项目上的属性

- order
- flex-grow
- flex-shrink
- flex-basis
- flex
- align-self

- order 数值越小越靠前

- flex-grow

  flex-grow 属性定义项目的放大比例，默认为0，即如果存在剩余空间，也不放大。

  如果所有子项都为1，则存在剩余空间，等比例分配，如果 一个是2 其余为1，则2的分配剩余空间的宽度是其余的1倍

- flex-shrink

  flex-shrink属性定义了项目的缩小比例，默认为1，即如果空间不足，项目将缩小。

  如果所有项目的flex-shrink属性都为1，当空间不足时，都将等比例缩小。

  如果一个项目的flex-shrink属性为0，其他项目都为1，则空间不足时，前者不缩小。

  负值对该属性无效。

- **align-self属性**

  align-self属性允许单个项目有与其他项目不一样的对齐方式，可覆盖align-items属性。

第二章 移动项目实战

- 移动端布局
- 布局使用百分比或rem,
- 百分比应该比rem更好控制一点；
- 等宽布局比如导航就可以采用box-sizing:border-box;内补白的方式平分布局。

```
// 移动字体大小的适配--方案一

@media screen and (max-width: 750px){
    html{font-size:30px;}
}

@media screen and (min-width:640px) and (max-width:749px){
    html{font-size:25px; }
}

@media screen and (min-width:480px) and (max-width:639px){
    html{font-size:20px; }
}

@media screen and (min-width:320px) and (max-width:479px){
    html{font-size:15px; }
}
```

- 关于字体
- iOS 系统

  > 默认中文字体是Heiti SC
  >
  > 默认英文字体是Helvetica
  >
  > 默认数字字体是HelveticaNeue
  >
  > 无微软雅黑字体

- Android 系统

  > 默认中文字体是Droidsansfallback
  >
  > 默认英文和数字字体是Droid Sans
  >
  > 无微软雅黑字体

- 各个手机系统有自己的默认字体，且都不支持微软雅黑，如无特殊需求，手机端无需定义中文字体，使用系统默认英文字体和数字字体可使用 Helvetica ， 三种系统都支持

  ```
  body{font-family:Helvetica;}
  ```

第一节 移动布局适配基本原理

- 范例 :移动适配框架的应用

```
    // 自定义移动适配框架

    function fontSize(){

            var _html=document.getElementsByTagName("html")[0];
            var w = document.documentElement.clientWidth;
            console.log(w);
            if(w>=750){
                    _html.style.fontSize="100px";
            }else{
                    _html.style.fontSize= w/7.5+"px";
            }

}

window.onresize = fontSize;

fontSize();
```

第二节应用框架实现移动适配布局

- 移动端的框架：
  zepto.js,Zepto.js是支持移动WebKit浏览器的JavaScript框架，具有与jQuery兼容的语法。轻量级，大小为2-5k的库，通过不错的API处理绝大多数的基本工作,比较适合移动端。
  移动端banner切换的插件可以用swiper

移动项目头部可增加的meta

```
<meta charset="UTF-8">

<!-- 优先使用 IE 最新版本和 Chrome -->
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />

<!-- 为移动设备添加 viewport -->
<meta name="viewport" content="width=device-width,initial-scale=1, maximum-scale=1, minimum-scale=1, user-scalable=no">

<!-- 添加到主屏后的标题（iOS 6 新增） -->
<meta name="apple-mobile-web-app-title" content="">

<!-- 是否启用 WebApp 全屏模式，删除苹果默认的工具栏和菜单栏 -->
<meta name="apple-mobile-web-app-capable" content="yes" />

<!-- 设置苹果工具栏颜色 -->
<meta name="apple-mobile-web-app-status-bar-style" content="black" />

<!-- 添加智能 App 广告条 Smart App Banner（iOS 6+ Safari） -->
<meta name="apple-itunes-app" content="app-id=myAppStoreID, affiliate-data=myAffiliateData, app-argument=myURL">

<!-- 忽略页面中的数字识别为电话，忽略email识别 -->
<meta name="format-detection" content="telephone=no, email=no" />
```

第三章 网易云音乐

- jquery对象绑定数据

$("#box").data("属性名","值") // 绑定数据

$("#box").data("属性名"); // 获取数据

- jq对象转换为js对象

var box = $("#box").get(0) // jq对象转js对象

var box ----> $(box) // js对象转jq对象

- 音乐播放 audio

var audio =document.getElementById("audio");

audio.play(); // 音乐播放

audio.pause(); // 音乐暂停

- hasClass(); 判断一个元素是否具有某个类名;
- localStorage 本地存储
  - cookie 存储数据相当于4kb 而localstorage 存储容量为5M
  - session 存储是基于登录状态的，登录过期后session数据就丢失，而 locastorage是永久保存的;

- localStorage 保存的是字符类型数据，因此对于json对象要字符串化后进行保存，使用时在解析为json对象

JSON.stringify() -- 将json对象转换为json字符串;

JSON.parse() -- 将json字符串 "{name:'tom'}" 转换为json对象;