用友集团iUAP中心

技术分享

分享人： 郭永峰

CONTENT

- 跨终端、多屏高清适配
- Viewport
- meta的使用
- 切图、图标icon、图标字体
- 移动端事件件
- 微信分享
- 调试

跨终端、多屏高清适配

- em
    - 对于元素的父元素的font-size进行计算
- media query
    - 响应布局实现之利器
    - 维护
- rem
    - 相对于根元素html的font-size进行计算
    - 避开很多层级的关系

> 移动端布局，为了适配各种大屏手机，目前最好用的方案莫过于使用相对单位 rem。

CANIUSE

rem (root em) units - CR

Global 93.37% + 2.37% = 95.74%

Type of unit similar to `em`, but relative only to the root element, not any parent element. Thus compounding does not occur as it does with `em` units.

Current aligned | Usage relative | Show all

IE / Edge	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
8							4.1	
9		31					4.3	
10		42					4.4	
11	38	43	7.1		7.1		4.4.4	
Edge	39	44	8	30	8.4	8	40	42
	40	45	9	31	9			
	41	46		32				
	42	47						

原理

> 针对不同手机屏幕尺寸和dpr动态的改变根节点html的font-size大小(基准值)。

- 屏幕尺寸
- dpr (devicePixelRatio设备像素比)
- 动态改变

计算html的font-size大小

```
html的font-size值(单位是rem) = 屏幕宽度 * dpr / 基数

1. dpr
2. 除以基数，是为了取整，方便计算(理论上可以是任何值)
```

- 以上是rem的css实现
- 当然，也还有rem的js实现
  - 根据设备dpr和屏幕宽度来计算viewport宽度，并根据设备dpr计算viewport缩放比例，然后在html标签中设置font-size的属性值

Viewport

场景

把一个普通的在PC上开发的HTML页面直接放手机上，你会发现不管多大的页面都可以在小小的手机屏幕上显示，但是图文都会显示的特别小；如果你用JAVASCRIPT获取下页面宽度，你会发现，大多数的页面宽度都是980px的，这个就跟viewport有关。

- 明白viewport的概念
  - **layout viewport** 浏览器默认输出
  - **visual viewport** 可以理解为设备自己的宽度
  - **ideal viewport** 一个完美适配移动设备的 viewport
- 弄清跟viewport有关的meta标签的使用

viewport的使用

```
<meta name="viewport" content="
    height = [ pixel_value |device-height] ,
    width = [ pixel_value |device-width ] ,
    initial-scale = float_value ,
    minimum-scale = float_value ,
    maximum-scale = float_value ,
    user-scalable =[yes | no] ,
    target- densitydpi = [ dpi_value | device-dpi| high-dpi | medium-dpi | low-dpi] "
```

meta标签

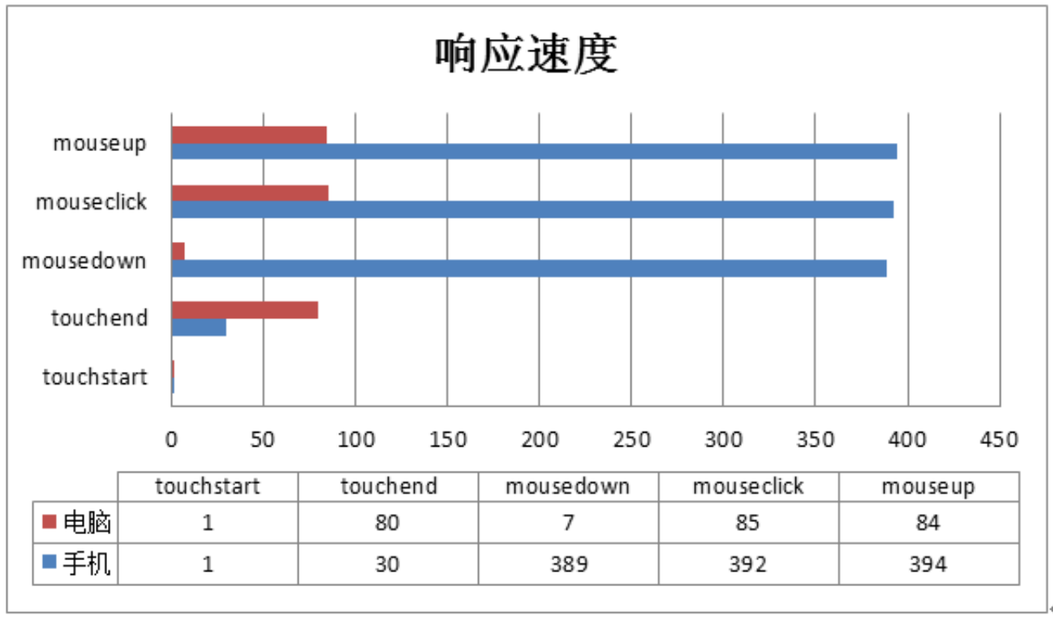
```
<!-- 是否启动webapp功能，会删除默认的苹果工具栏和菜单栏。 -->
<meta name="apple-mobile-web-app-capable" content="yes" />
<!-- 显示手机信号、时间、电池的顶部导航栏的颜色。默认值为default（白色），可以定为black（黑色）
<meta name="apple-mobile-web-app-status-bar-style" content="black" />
<!-- 忽略页面中的数字识别为电话号码或是邮箱 -->
<meta name="format-detection" content="telphone=no, email=no" />
<!-- ios 在网页加载时隐藏地址栏与导航栏 -->
<meta name="viewport" content="minimal-ui">
<!-- 启用360浏览器的极速模式(webkit) -->
<meta name="renderer" content="webkit">
<!-- 避免IE使用兼容模式 -->
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<!-- 针对手持设备优化，主要是针对一些老的不识别viewport的浏览器，比如黑莓 -->
<meta name="HandheldFriendly" content="true">
<!-- 微软的老式浏览器 -->
<meta name="MobileOptimized" content="320">
<!-- uc强制竖屏 -->
<meta name="screen-orientation" content="portrait">
<!-- QQ强制竖屏 -->
<meta name="x5-orientation" content="portrait">
<!-- UC强制全屏 -->
<meta name="full-screen" content="yes">
<!-- QQ强制全屏 -->
<meta name="x5-fullscreen" content="true">
<!-- UC应用模式 -->
<meta name="browsermode" content="application">
<!-- QQ应用模式 -->
<meta name="x5-page-mode" content="app">
<!-- windows phone 点击无高光 -->
<meta name="msapplication-tap-highlight" content="no">
```

切图、图标icon、图标字体

- png24 png8 jpg gif jpeg
- css sprite
- svg
- 图标字体

移动端click延迟和点透的问题

数据说明



	touchstart	touchend	mousedown	mouseclick	mouseup
电脑	1	80	7	85	84
手机	1	30	389	392	394

- 原因
- 不用click用touch
- zepto.js的tap事件

- 使用zepto出现的点透问题（特定场景）

> 在重叠的区域里，被遮盖的元素绑定click，遮盖的元素绑定touch事件，且touch后遮盖的元素会隐藏的话，就会造成穿透，因为click是在touch之后延迟触发的，浏览器会误认为是在遮盖的元素上触发了click。

方案

- 用touchend代替tap事件并阻止掉touchend的默认行为preventDefault()
- 延迟一定的时间(300ms+)来处理事件
- tap.js：监听tap事件，不使用click
- FastClick库（10KB），检测到touchend事件的时候，会通过 DOM 自定义事件立即触发一个模拟click事件，并把浏览器在 300毫秒之后真正触发的click事件阻止掉。

微信分享

- 确保引入js-sdk
- 封装统一的调用接口
- 接口调用，传入参数

调试

- 生成页面二维码
- fiddler抓包
- 模拟器
- 使用 Weinre 调试
- browser-sync

THANKS