## 流式布局

特点：

就是百分比布局,非固定像素,内容向两侧填充,理解成流动的布局,称为流式布局

## 适口配置

### 适配要求：

1. 网页宽度必须和浏览器保持一致 即不能出现滚动条

2. 默认显示的缩放比例和PC端保持（缩放比例1.0）

3. 不允许用户自行缩放网页

满足这些要求达到了适配，国际上通用的适配方案，标准的移动端适配方案。

### 视口

视觉窗口 viewport

是移动端特有（pc端也有但是和浏览器窗口大小一样 看不见）

是一个用来承载网页的虚拟区域 这个区域在浏览器窗口内 具备缩放功能 可以设置高宽

视口设置在移动端才生效

### 视口的设置

<meta name=”viewport”

content=”width=device-width,initial-scale=1.0,user-scalable=0”>

meta:vp + tab 快捷方式

参数：

1. width 可以设置宽度 (device-width 当前设备的宽度)

2. height 可以设置高度

3. initial-scale 可以设置默认的缩放比例

4. user-scalable 可以设置是否允许用户自行缩放 （yes，no 1,0）

5. maximum-scale 可以设置最大缩放比例

6. minimum-scale 可以设置最小缩放比例

## 移动端base.css

Box-sizing:border-box; 防止内容溢出

设置内边距时盒子不会被撑大 而是内容区域被压缩

内容区域=设置的宽高-内边距的值

-webkit-tap-highlight-color: red; 点击时的高亮颜色

transparent 透明色 即清除高亮效果

文本域 输入框 初始化

Input , textarea{

Border:none; 隐藏边框

Outline:none; 隐藏 选中后的外轮廓

Vertical-align:middle;

Resize:none; 针对文本域 不能拖拽更改尺寸

-webkit-appearance:none; 移动端 清除控件的默认样式

}

其他：

禁用客户端软件的字号调整

-webkit-text-size-adjust: 100%;

解决IOS默认滑动很卡的情况

-webkit-overflow-scrolling : touch;

解决点击高亮效果

-webkit-tap-highlight-color: transparent;

设置元素的尺寸计算从边框开始

-webkit-box-sizing:border-box;

清除表单控件默认样式

-webkit-appearance: none;

## 移动端index.css

首先把所有内容放进一个div中 即版心容器

宽度限制：根据设计稿的最大最小宽度

max-width:640px;

min-width:320px;

width:100%;

margin: 0 auto;

## 双飞翼布局

两边的翅膀：用position定位

Position:absolute;

Right:0;

Top:0;

中间：用padding把两边预留出来

.box{

Width:100%;

Padding-left:60px;

Padding-right:50px;

}

## classList操作类名

H5新增属性 dom操作

Node.classList.remove('类名'；)

Node.classList.add('类名'；)

Node.classList.toggle('类名'；)

## Background

### 背景原点 origin

background-origin:

content-box / border-box / padding-box

背景平铺的起点 默认原点是内边距 padding-box 左上角开始平铺

Border-box 背景原点从边框开始

### 背景裁切 clip

background-clip:

content-box / border-box / padding-box

背景被裁切的区域；

内容以外裁剪 / 边框以外 / 内边距以外；

## 绝缘容器

左侧容器浮动；右侧容器用overflow:hidden;

作用：左右两边元素互不影响；左侧固定 右侧自适应

Html中 把需要浮动的元素（固定的元素）放在前面 必须先解析

原因是触发了BFC；

## iScroll插件

是一种区域滚动插件 使拖拽 滚动更平滑

☆ 需要做滑动效果时：

如果是纵向滑动 子容器的高度要大于父容器的高度

如果是横向滑动 子容器的宽度要大于父容器的宽度

使用方法：

引入iscoll.js文件

初始化代码：

window.onload = function () {

new IScroll('选择器',{ // 选择父容器

scrollX:true,

scrollY:false,

/\*click 不能触发 可以使用tap \*/

click:true

});

}

## touch事件组

### 事件

touchstart/touchmove/touchend/touchcancel

touchcancel 当滑动被迫终止的时候触发 （电话，弹消息）

### 触摸点集合

1. 绑定事件：box.addEventListener('touchstart',function () { });
2. 事件对象：

TouchList------触摸点（一个手指触摸就是一个触发点，和屏幕的接触点的个数）的集合

集合模式 调用时写成 xxx[0]的形式

touches 【页面】所有的触摸点；

targetToches【当前元素】所有的触摸点集合；

changedTouches 【改变后】的触摸点集合；

3.触摸点集合在每个事件触发的时候会不会去记录触摸

changedTouches 每个事件都会记录

targetTouches，touches 在离开屏幕的时候无法记录触摸点

### 触摸点信息对象中的坐标

clientX/Y:可视区域或视口 当前可以看到的内容 没显示部分不计 类似于fixed

pageX/Y:整个页面文档 没显示部分包括在内 不包括浏览器窗口、滚动条

screenX/Y:整个屏幕 包括浏览器窗口、滚动条 没显示部分不计

## tab事件

### 轻触事件

①、不是移动端原生事件；是移动端的某些库(zepto)封装了tap事件，tap事件是由touch事件衍生出来的；

②、较快的触发点击

③、移动端的click事件：会有300ms的延时（反应慢 体验差），刚开始延时300ms是为了双击放大屏幕，但是现在我们不让用户放大屏幕了，为了解决这300ms的延时问题，出现了模拟tap事件；

### 模拟实现tab

快速响应：start和end事件之间时间差<=150ms;

不能滑动：即不能有move事件；

### fastClick插件

也可以解决300ms的延时问题 awesome官网上可以查询 github上可以下载

## swipe事件

滑动手势事件：不是移动端原生事件，由touch衍生来的，封装来的；

必须滑动过 滑动的距离足够大 50px

## 响应式布局

一个网站能够兼容多个终端

IMG_256 终端归类

移动端 0-768px 称为超小屏设备

Ipad 768-992px 称为小屏设备

标屏显示器 992-1200px 称为中屏设备

Pc端 宽屏显示器 1200px以上 称为大屏设备

## 媒体查询

@media 关键字

语法： @media screen and (max-width: 768px) and (min-width: 320px){属性样式}

有空格

精简写法：

范围最大的放在最上面

@media screen and (max-width: 1200px){}

@media screen and (max-width: 992px){}

@media screen and (max-width: 768px) {}

@media screen and (min-width: 1200px){}

## 第三方代码

1. 库、js库、css库：对一些常用工具函数的封装的集合 或常用类的集合 往往只包含js css 中的一个

Jquery font-awemsome

1. 插件：针对一些常用功能的封装 基于原生js封装 一般含有html+css+js 比价齐全 需要遵循它的html结构来写

Fullpage iscroll jquery.lazyload.js

1. UI框架：用户界面框架 有着很多常用的插件集合

Jquery UI easyUI bootstrap 妹子UI mui weui

1. Js 框架 具备一定的开发思想的js库

mvc mvvm vue angular react

## 条件注释

满足条件 注释内的内容就会执行

例：

<!--[if lt IE 9]>

<h1>您的版本浏览器太低，请升级</h1> 低于IE9时执行

<![endif]-->

## ---bootstrap--------------------

响应式框架

虽然界面组件样式已经定义好了，但是扩展性相对较强，也就是说我们还可以自定义，修改默认样式。

### Normalize

Reset css 作用：重置样式为了浏览器的表现一致

Normalize 作用：为了增强跨浏览器表现的一致性 css重置样式库

☆ 区别：

Reset css 自己定义样式 破坏性 --- 直接把样式重置成一样的 不管之前是否一样 直接覆盖掉 比如 list-style:none;

Normalize 第三方重置样式的库 柔和 --- 如果样式在浏览器已经表现一致 那么就不进行操作 比如 ul没有去掉点

### 容器类

.Container 响应式布局容器

.Container-fluid 流式布局容器

## 栅格系统

行和列的布局，网格状布局

### .row 行类

可以填补父容器的内边距 margin-left + margin-right 拉伸容器

### .col 列类

Col - \* - \* \*为两个参数

第一个参数：

屏幕的大小

大屏设备 lg 大屏设备以上生效包含本身

中屏设备 md 中屏设备以上生效包含本身

小屏设备 sm 小屏设备以上生效包含本身

超小屏设备 xs 超小屏设备以上生效包含本身

第二个参数：

范围1 - 12

每一行的分等份，默认分成12等份 ，数字代表的是占多少份

无法沾满容器 每列左右均有 15px间隔 可以用.row 来填补内边距

列偏移：

Col-\*-offset-\* 偏移量也是占份数的 偏移多了一行就装不下了 就会换行

## 响应式工具

.visible-\*-\* 显示

.hidden-\*-\* 隐藏 （推荐使用）



## 自定义字体图标

记住步骤！！！ 用颜色标出的是需要变的量 其余为固定代码

1.自定义字体图标

@font-face {

font-family: heima40; 自定义一个名字

src:

url(../fonts/MiFie-Web-Font.svg) format('svg'),

url(../fonts/MiFie-Web-Font.ttf) format('truetype'),

url(../fonts/MiFie-Web-Font.woff) format('woff'),

url(../fonts/MiFie-Web-Font.eot) format('embedded-opentype');

四个文件 后面的这些事固定的 查阅文档就行

}

2.定义使用字体的类

.wjs\_icon{

font-family: heima40;

}

3.通过类 追加对应的字符编码 对应图标

.wjs\_icon\_phone::before{

content: "\e908"; 查阅图标对应的码

}

.wjs\_icon\_tel::before{

content: "\e909";

}

html代码：

<span class="wjs\_icon wjs\_icon\_phone"></span>

把两个类名都要写上

☆ 注：bootstrap 中自带的图标 只需把文档中的类名复制过来



## ---Less----------------

## 语法

@charset “utf-8” 字符编码 防止乱码 写在最前边

### 注释

/\* \*/ 会编译到css

// 不会编译到css

### 变量

定义：

@mainColor:#e92322；

@modelName:wjs;

注意： 必须以@开头 不能有特殊字符（\*）

使用：

a:hover{

Color:@mainColor; // 这样可以批量更改颜色

}

任何时候 在字符串拼接时加{ } @{ 变量名 }

.@{modelName}\_box h3{

}

## 混入

### 样式混入

.f\_left{

Float:left;

}

.createdCss{

.f\_left( ); //直接调用

}

### 函数混入

使用函数混入，在css代码中就不会出现原始代码

.f\_left( ){ //加上括号就是函数 不加括号就是样式

float:left;

}

.createdCss{

.f\_left( ); //直接调用

}

和样式混入的调用方法一样 所以不能重名

### 函数混入优化：设置参数

.floatFuc(@direction:left ){ :left是设置默认值

Float:@direction;

}

调用：

.floatFuc(left);

// 必须传参数 不传参数会报错（前提是定义的参数没有默认值）

## 嵌套

.a{

>.b{ }

.c{ }

}

.a>.b{ }

.a .c{ }

—————————————

.a{

> div{

&:last-child{ //用作字符串连接

}

}

}

.a>div:last-child{ }

## 导入

@import “ 文件名 ”；

@Import 书写在css文件里 去载入其他的css文件

总结：

Link 几乎会同时请求所有的资源

@import 需要等css文件加载完解析之后再去做下一次的引入

## 内置函数&运算

JavaScript中的内置函数： Math Date

运算：例如百分比的换算

@num:7; // 可通过更改num值来维护

.banner{

Width:100%;

ul{

Width:100% \* @num;

li{

Width:percentage(1/@num); //内置函数 1/num\*100%

}

}

}

## 引入方法

1.在html中引入：

<link rel=”stylesheet” type=”text/less” href=”less/index.less”>

被Less.js加载的前提

1. 使用js将link解析成css 写在头部

<script src=”assets/less/less.min.js”></script>

去hithub下载

<script>less.watch();</script> 不用刷新就能看到效果

这种引入只是方便编译 真正开发的项目中还是使用index.css

## 递归遍历

Less本身没有循环分支语句 使用递归来遍历 必须有结束条件

例：

.adapter(@index) when(@index<=@len){  
 *//适配代码* @media(min-width:extract(@deviceWidth,@index)){  
 html{  
 font-size:extract(@deviceWidth,@index)/@psdWidth\*@baseSize;  
 }  
 }  
 .adapter(@index+1); //递归  
};  
.adapter(1);

## ---rem---------------------

固定像素 px

相对单位 100% em rem

百分比：相对于父容器内容尺寸

em:相对于父容器的字体大小

rem:相对于html文档根元素字体大小 根元素就是html

## 适配原理

1. 流式布局：百分比布局 特点：宽度自适应 高度固定
2. 伸缩布局：使用flex 特点：宽度自适应 高度固定
3. 响应式布局：媒体查询&百分比
4. 淘宝：视口等比缩放原网页
5. rem适配方案：高宽自适应 等比缩放

### rem适配

rem基于根元素字体大小

当基准值改变时 以rem为单位的元素尺寸也会改变

在不同大小的设备中 更改基准值 根据当前设备的宽度等比更改

每适应一个设备就需要一个媒体查询 以及换算基准值

所以需要去适配所有的设备比较困难，但可以适配主流设备

### 适配方案

Rem&媒体查询

Basesize 的大小可以任意设置

Rem&flexibleJS

Basesize 的大小是 当前设备宽度/10

通用：

当前设备的基准值（font size）=

当前设备的宽度/设计稿的宽度\*初始基准值（basesize）

当前元素大小=设计稿测量大小rem/初始基准值