================

浏览器内核： IE 谷歌 火狐 Safari Opera

Trident Blink Gecko Webkit Presto（现为Blink）

==================

Web标准

\* 结构标准：主要包括XML和XHML

\* 样式标准：主要是指CSS

\* 行为标准：主要包括DOM和ECMAScript

网页：

组成：文字、图片、按钮、搜索框、视频、元素（html标签）组成的

总结：网页其实就是由html标签组成的

============

HTML

<!DOCTYPE html> // Doctype：告诉浏览器当前文档的类型

<html lang="en"> // html: 结构中的根标签

<head> // head: 可以设置网页样式、标题、关键字、描述信息...

   <meta charset="UTF-8">

<meta name="viewport" content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">

   <title>Document</title>

</head>

<body> // body: 网页的主体（我们看到的所有信息）

</body>

</html>

标签

\* 单标签：

<hr> //横线

<br> //换行

<meta charset="UTF-8"> // 设置网页编码

<!-- 内容 --> // 注释

\* 双标签

<h1></h1> 在页面中只能出现一次

废弃标签

<strong>加粗</strong> <b>加粗</b>

<em>文本倾斜</em> <i>文本倾斜</i>

<del>删除线</del> <s>删除线</s>

<ins>下划线</ins> <u>下划线</u>

<sup>上标</sup> <sub>下标</sub>

<font color=“red” size=“5”>红色文字</font>

a标签

<a herf=“test.html”>超链接</a>

<a herf=“test.html”target=“\_blank”>链接</a>

“\_self” 默认值，在当前窗口打开页面

<a herf=“test.html”title=“鼠标悬停到超链接上的文字提示”>链接</a>

把 <base target=“\_blank”> 放在head标签里，一般是link标签的位置

让页面中所有的超链接都在新窗口打开

超链接在当前页面内跳转

\* 直接设置<a href=“#”></a>

\* 锚链接

<p id=“sd”>文本</p>

<a href=“#sd”>跳转</a> //跳转到<p id=“sd”>文本</p>这个锚点

实际应用：点击快速回到段落文本顶部

标签语义化

根据内容的结构化（语义化），选择合适的标签（代码语义化）

作用：

\* 网页结构更合理

\* 有利于seo：和搜索引擎建立良好沟通，有了良好的结构和语义，网页内容自然容易被搜索引擎抓起

\* 方便其他设备解析（如屏幕阅读器、移动设备）

\* 便于团队开发和维护

实现：

\* 在语义不明确的时候，选择有语义的表签

\* 在设置样式时，用css

语义标签

<nav></nav> 导航区域

<header></header> 头部区域

<footer></footer> 结尾区域

<article></article> 呈现一篇文章 主体

<aside></aside> 侧边栏区域

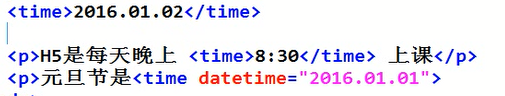
<section></section> 用来划分区域 划分不同的章节

<dialog></dialog> 标记定义一个对话框（会话框）类似微信

<figure></figure> 用于对元素进行组合，一般是图片或者视频

<figcaption></figcaption> figure子元素，对figure进行解释说明

<time></time> 定义时间



<hgroup></hgroup> 标题组合

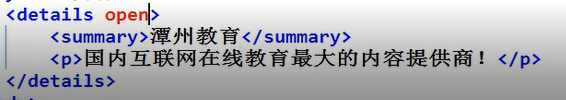
<aside></aside> 和主体相关的附属信息

<datalist></datalist> 和表单相关联

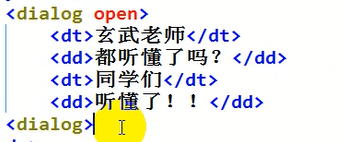
<option><option>或ajax实现

<details></details> 详情 open属性 自己展开





<dialog></dialog> 对话标签



<address></address> 定义文章或作者的详细信息

<mark></mark> 标记 默认为黄色背景 <mark>作者</mark>



<keygen></keygen> 公钥

<progress></progress> 进度条



<meter></meter> 温度计等



<menu></menu> 重新定义用户界面的菜单，配合commond或者menuitem使用

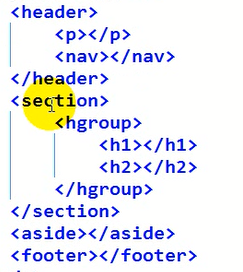


<ruby></ruby>





加括号是为了兼容部分浏览器



列表

无序列表

<ul type=“circle”></ul>

circle 空心圆圈

disc 实心圆圈

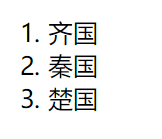
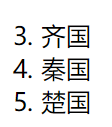
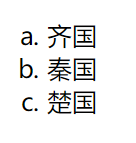
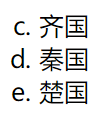
square 小方块

有序列表

<ol type=“1” start=“3”></ol>

type=“1” type=“1” type=“a” type=“m”

start=“3” start=“3”

自定义列表

<dl>

<dt>送货方式</dt>

<dd>自取</dd>

<dd>快递</dd>

</dl>

多媒体标签

音频标签

<audio src=“”controls loop autoplay width=“300”></audio>

<audio controls loop autoplay>

<source src=“01.mp3”></source>

<source src=“01.otg”></source>

</audio>

embed（多媒体）

可以用来插入各种多媒体，格式可以是MP3、wav等

背景音乐标签 (会自动播放)

<embed src=“1.mp3”hidden=“false”>

插入视频

<embed src=“1.mp4”hidden=“false”>

视频标签

<video src=“”controls loop autoplay width=“300”></video>

controls 显示控制栏

loop 设置视频循环播放

autoplay 自动播放

width 设置宽度，高度会随宽度自动调整

<video controls>

<source src=“01.avi”>

<source src=“01.mp4”>

<source src=“01.itcast”>

</video>

滚动

<marquee behavior=“side” direction=“down”height=“300px” bgcolor=“red”>文本或图片</marquee>

behavior 滚动的方式

alternate 在两端之间来回滚动

scroll 一端滚动到另一端，会重复

slide 一端滚动到另一端，不重复

direction 滚动的方向

meta标签

<meta charset="utf-8"> //设置字符集编码格式

gb2312 //简体中文

gbk

unicode

big5 //繁体中文

作用： 防止页面乱码

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <meta http-equiv="X-UA-Compatible" content="ie=edge">

<meta name=“keywords”content=“日用百货，男装，女装”>

<meta name=“description”content=“我们店里的销售日用百货，有...”>

实现网页重定向

<meta http-equiv=“refresh”content=“3; https://www.baidu.com”>

打开当前页面，3s后自动跳转到百度页面

应用：搬迁新网站

link标签

<link rel=“icon”href=“favicon.ico”> //设置网页图标

一般将网页图标放置到网站的跟目录下

<link rel="stylesheet" href="base.css">

表格

<table border=“1” cellspacing=“0” aling=“center” width=“500” height=“600”bgcolor=“red”>

<caption><h3>表头（如 课程表）</h3></caption>

<tr>

<td></td>

<td></td>

</tr>

</table>

border 设置边框 默认为0

cellspacing 表示td与td之间的距离 默认为2

align 设置对齐方式 left|center|right

在table上，只能改变table的对齐方式，内部不能改变

在 tr 或 td 标签上设置，则会改变内容的对齐方式

caption 设置表格表头

colspan=“2” 横向合并

rowspan=“3” 纵向合并

注意：合并完后要把多余的列或行删掉

table { border-collapse:collapse; }

//border-collapse 属性设置表格的边框被合并为一个单一的边框

表单

<form action=“” method=“” novalidate>

// action 设置一个用来接收数据的后台程序

// method 提交方式 get|post

// novalidata 关闭验证（如 email的验证）

\* 文本输入框

<input type=“text”name=“username”value=“” maxlength=“6” >

属性：

maxlength 设置控件中最多能输入多少个字符

readonly 设置控件为只读模式

disabled 设置控件为未激活

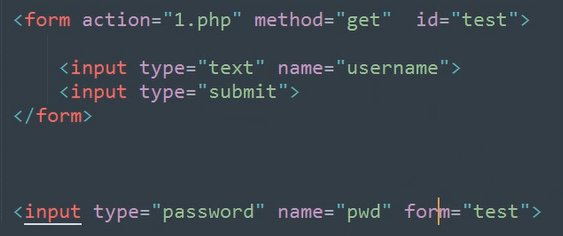
placeholder 提示信息

autofocus 自动获取焦点

autocomplete=“off | on” 自动完成（是否记录每次提交的信息）

required 必填项

form=“表单ID” <input type=“text” form=“表单ID”>



</form>

\* 密码输入框

<input type=“password”>

属性：和文本输入框一样

\* 复选框

<input type=“checkbox”checked>学习

<input type=“checkbox”>跑步

<input type=“checkbox”>工作

属性：

checked 默认选中项

\* 单选框

<input type=“radio” name=“sex” checked>男

<input type=“radio” name=“sex”>女

下拉列表

<select multiple>

<option>河北</option>

<option selected>江西</option>

<option>上海</option>

<option>北京</option>

</select>

属性： multiple 设置多选

<select multiple>

<optgroup label=“江西省”>

<option>南昌</option>

<option selected>宜春</option>

<option>新余</option>

<option>赣州</option>

</optgroup>

</select>

<input type="text" list="list\_1">

<datalist id=“list\_1”>

<option value=“江西”>江西</ooption>

<option value=“新疆”>新疆</ooption>

<option value=“西藏”>西藏</ooption>

</datalist>

滑块

<input type=“range” value=“50”>

上传文件

<input type=“file”>

多行文本域

<textarea clos=“” rows=“”></textarea>

邮箱

<input type=“email”>

电话号码（移动端使用）

<input type=“tel”>

网址

<input type=“url”>

数字

<input type=“number” step=“5”>

颜色

<input type=“color”>

日期 （可以弹出日历选择日期）

<input type=“date”>

<input type=“week”>

<input type=“month”>

按钮

<input type=“submit”value=“注册”>

value 默认为‘提交’二字

图片按钮

<input type=“image” src=“01.jpg”>

两个按钮都可用于提交表单信息

<input type=“reset”>

value 默认为‘重置’二字

普通按钮

<input type=“button”value=“注册”>

注意： \* 该按钮不能直接在表单域中提交数据

\* 一般配合js使用

<form>

<fieldset>

<legend>分组</legend>

</fieldset>

</form>

特殊标签

&nbsp 空格符

&lt 小于号

&gt 大于号

h5中添加了data-\*的方式来自定义属性

<div id=”test” data-age=”age”></div>

可以通过dataset设置自定义属性 （如下：data-name）

document.querySelect(“#test”).dataset.name = “Byron”;

document.querySelect(“#test”).dataset.userName = “小龙”;

读取自定义属性方法：

document.querySelect(“#test”).dataset.name;

document.querySelect(“#test”).dataset.userName;

jQ中用data()的方法

设置

$(“#.test”).data(“userName”, “小明”);

读取

$(“#.test”).data(“userName”);

==============

CSS

作用： \* 美化网页

\* 网页布局

行内式、内联、外联

选择器

基础选择器

标签选择器

类选择器

ID选择器

通配符选择器

复合选择器（交集选择器）

标签指定式选择器

后代选择器

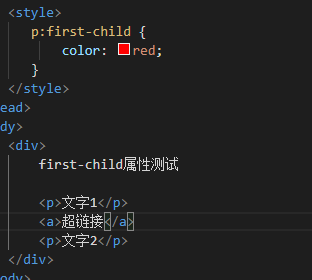
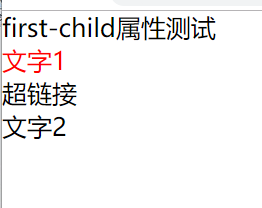
子代选择器

并集选择器

属性选择器

伪类选择器

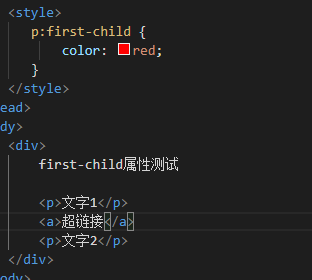
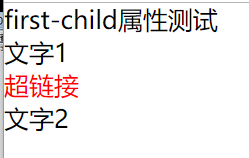
P:first-child 父元素里的第一个子元素，且是p标签

:first-child 相当于 \*:first-child

p:last-child

a:nth-child(2) 从1开始取值

p:nth-child(n) n的取值0,1,2......

p:nth-child(2n+3)

p:nth-child(odd) 奇数

p:nth-child(even) 偶数

:nth-last-child(n) 选中倒数第n个子元素

div + div的下一个兄弟

div ~ div后面的所有兄弟

:not 反着选

p:not(.demo) 没有demo类的p元素

:only-child 唯一的儿子(独生子)

:empty 空标签

<P></P> 没有任何文字、空格、换行、tab

:checked 选中到单选按钮、复选框被勾选

:disabled 不可编辑的或按钮不能点击的

:enabled 可编辑或按钮能点击的

伪元素选择器

::first-line 选中第一行文字

::first-letter 选中第一个字母或汉字

::selection 设置鼠标选中区域的样式

伪元素 属于行内元素

::before

::after

伪类

a:link | visited | hover | active

:active 鼠标按下时一瞬间的属性（移动端手指按下），也可以用于其他标签（如button）

移动端应用： 手指按下背景色变色，抬起还原（ios可能不激活）

input:focus { } 当获取焦点时的样式

a:focus { } 超链接也可以有焦点(点击a链接时获取焦点)

目标伪类

:target

注： 1、目标伪类与描点连接配合使用

2、只有当被描点链接指向该标签的时候才会执行目标伪类中的css代码

例子：点击切换元素或图片（:target应用）

CSS特性

\* 层叠性

在同一标签中，如果定义的样式发生冲突，那最后一次定义的样式会覆盖掉前面定义的样式

\* 继承性

- 只有在嵌套关系中才会出项

- 如果子元素默认没有样式，该元素会受父元素的样式影响

- color， font属性 ，text-align...都可以被子元素继承

- 注意：   
> width 和 height 不能被继承  
> a标签在默认情况下，不受父元素的文字颜色影响  
> 标题标签在默认情况下，不能直接等于父元素设置的文字大小

\* 优先级

标签选择器 < 类选择器 < ID选择器 < 行内样式 < !important  
10 100 1000 10000 10000以上

- 继承的权重为0

- 权重可以叠加

标准流： 元素自上而下，自左而右，块元素独占一行，行内元素在一行上显示，碰到父元素的边框换行

块元素 div ul ol li p h1-h6 dt ......

特点： 1、不设置宽度，则为父元素宽度

2、独占一行

3、可以设置宽、高

行内元素 span a strong ( b i )

特点： 1、所有元素都在一行显示

2、不能设置宽、高

行内块元素 input img

特点：1、可以设置宽、高

2、所有元素都在一行显示

display: block | inline | inline-block

p标签不能包含div、ul、h1-h6 ...

h1-h6可以包含div、p ...

div可以包含所有的标签

行内元素尽量包含行内元素，不要包含块元素

border

border-style: solid | dotted | dashed

点线

border-left: 10px solid transparent;

边框合并 （只能用在表格中）

border-collapse: collapse;

圆角边框

border-radius: 10px;

border-radius: 20px/30px

水平半径/垂直半径

border-radius: 20px 0px 0px 0px;

左上角变成圆角

outline 设置元素周围的轮廓

outline: #ccc dotted thick;

outline: #ccc dotted; //两个值也可以

outline-color

outline-style

outline-width

padding

margin

子元素设置margin-top: 20px;

父元素也向下移动20px;

解决方法： 1、父元素加一个边框

2、父元素设置 overflow: hidden; (推荐)

3、让子元素或者父元素脱标

margin: 0 auto; 只能让标准流的盒子居中

注：不能给行内元素设置上下内外边距

vertical-align: baseline | middle | top | bottom | sub | super | text-top ...

基线对齐 中线对齐

（默认）

表示当前元素在父元素中的排列的垂直位置，同时还包含了当前元素和兄弟元素之间的关系

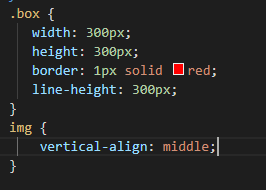
注：（在子元素中的行内块元素或table中使用，表示各子元素的垂直对齐方式）

baseline 子元素放置在父元素的基线上

middle 子元素放置在父元素的中部

应用：图片和文字垂直居中对齐

图片垂直居中



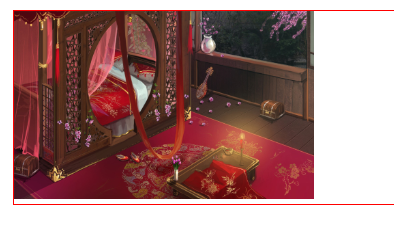
解决图片底部多出来的高度：三种方式

1、 vertical-align: top | bottom | middle

2、 display: block;

3、 font-size: 0; （放在img的父容器中）

（font-size越大，这个高就越大，font-size=0时，这个高就位0）

元素隐藏

overflow: hidden: 溢出隐藏

display: none: (不占据原来的位置)

visibility: hidden; （还占据原来的位置）

overflow: visible | hidden | scroll | auto

定位

position: static | relative | absolute | fixed | inherit

静态定位（默认）

绝对定位、固定定位的元素会转为行内块（因为脱标了）

z-index 只有定位的元素（static除外）才能设置

opacity 颜色和图片都会有透明效果，不需望文字透明，只是颜色，则使用background-color: rgba(255,255,255,.9);

font属性

font-family

宋体 SimSun \5B88\4F53 (Unicde编码)

新宋体 NSimSun \65B0\5BBB\4F53

微软雅黑 microsoft yahei \5FAE\8F6F\96C5\9ED1

font-style: normal | italic

斜体

font-size: medium | inherit | 50% | 16px

medium: 默认值

inherit: 从父元素继承字体尺寸

50%: 基于父元素的一个百分比值

属性连写：

{ font-style:italic;

font-variant:small-caps;

font-weight:700;

font-size:12px;

line-height:24px;

font-family:"Microsoft YaHei";

}

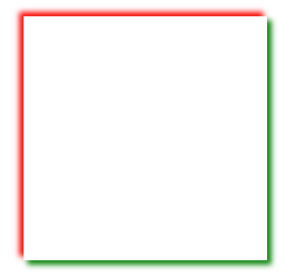
font: italic small-caps 700 12px/24px "Microsoft YaHei";

必选 必选

盒子阴影

box-shadow: 5px 5px 10px green,

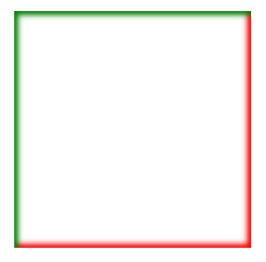
-5px -5px 10px red;



盒子内阴影

box-shadow: inset 5px 5px 10px green,

inset -5px -5px 10px red;



文本属性

\* color：rgba()

r red 红色 0-255

g green 绿色 0-255

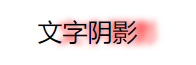
b blue 蓝色 0-255

a alpha 透明 0-1

\* text-shadow 文字阴影

text-shadow: 0px 0px 10px red;

第一个值：设置文字阴影在水平方向的一个偏移量， 正值向右偏移，负值左移

第二个值：垂直方向的偏移量

第三个值：阴影的模糊度， 数字越大越模糊，不能设置负数

第四个值：颜色

应用：立体文字

设置多个阴影 text-shadow: 0px 0px 10px red,

1px 2px 5px green;

\* text-decoration 文本修饰线

text-decoration-line 指定文本装饰的种类。

相当于CSS1时的text-decoration属性

text-decoration: none | underline | line-through

下划线 删除线

text-decoration-style 指定文本装饰的样式。

text-decoration-color 指定文本装饰的颜色。

联写 text-decoration : #F00 double overline

\* text-transform 文字的大小写

none： 默认值 无转换

capitalize： 将每个单词的第一个字母转换成大写

uppercase： 转换成大写

lowercase： 转换成小写

\* ext-fill-color 文字填充颜色

\* text-stroke 设置文字的描边

text-stroke-width 文字的描边厚度

text-stroke-color 文字的描边颜色

\* word-wrap 当前行超过指定容器的边界时是否断开转行

normal 默认值

允许内容顶开或溢出指定的容器边界。

break-word

内容将在边界内换行。如果需要，单词内部允许断行。

white-space: normal | pre | nowrap | pre-wrap | pre-line | inherit

normal 默认。

空白(如 空格键，多个空格键保留一个)， 空白符系列(如 &nbsp; 有几个保留几个)，不保留换行符， 文本超出正常换行

pre 空白(如 空格键，多个空格键保留多个)， 保留空白符系列(如 &nbsp; 有几个保留几个)，保留换行符（如 Enter键换行， 有几个保留几个）， 每一行的文本超出不换行，有换行符才换行

nowrap 空白(如 空格键，多个空格键只也保留一个)， 保留空白符系列(如 &nbsp; 有几个保留几个)，换行符（如 Enter键换行， 有多个也默认为一个空白），文本超出不换行，有换行符也不换行

pre-wrap 空白(如 空格键，多个空格键保留多个)， 保留空白符系列(如 &nbsp; 有几个保留几个)，保留换行符（如 Enter键换行， 有几个保留几个），每一行最后有一个默认的换行符， 文本超出正常换行， 有换行符正常换行

pre-line 空白(如 空格键， 多个空格键只也保留一个)，保留空白符系列(如 &nbsp; 有几个保留几个)，保留换行符（如 Enter键换行， 有几个保留几个）， 文本超出正常换行， 有换行符正常换行

inherit 规定应该从父元素继承white-space属性的值

\* text-overflow 是否使用一个省略标记（...）标示对象内文本的溢出

text-overflow: clip | ellipsis

默认

一行文本溢出显示省列号(...)

overflow: hidden;

text-overflow: ellipsis;

white-space: nowrap:

\* text-align 文本的对齐方式

left 默认值 左对齐

right 右对齐

center 居中

justify 内容两端对齐

start 开始边界对齐

end 结束边界对齐

背景

background-color: transparent | red ...

透明色（默认）

background-image: url(“ ”);

background-repeat: repeat | no-repeat | repeat-x | repeat-y

默认

background-position: 20px 50px;

一个值时，第二个值默认为center

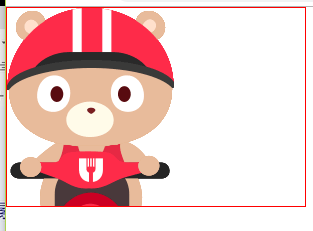
background-position: top right;

background-attachment: scroll | fixed

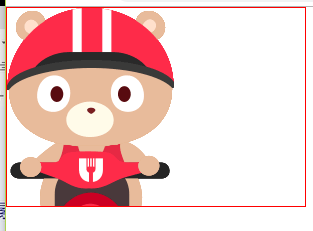
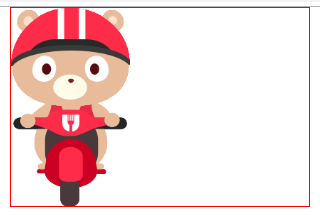
scroll:

background-size: 100px 200px;

background-size: cover; 按照图片原来的缩放比，始终占满整个父元素

background-size: contain; 按照图片原来的缩放比，自己完整的显示在父容器中

联写

background: url(“img/01.jpg”) 30px 60px no-repeat red;

没有顺序限制

多背景 background: url(“img/01.jpg”) no-repeat left top,

url(“img/02.jpg”) no-repeat right top,

#fff url(“img/03.jpg”) no-repeat center;

渐变

线性渐变：背景颜色由一种颜色向另外一种颜色渐变

background-image: linear-gradient (

to right, /\* 渐变的方向从左向右 \*/

red, /\* 渐变的开始颜色 \*/

green /\* 渐变的结束颜色 \*/

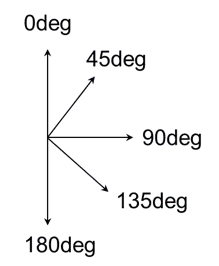
)



background-image: linear-gradient(

90deg /\* 渐变的角度（和to right的效果一样） \*/

/\* 0deg（和to top的效果一样） \*/



red 50%, /\* 0%到50%区域是红色向红色的渐变 \*/

green 100% /\* 50%到100%区域是红色向蓝色渐变 \*/

)



background-image: linear-gradient (

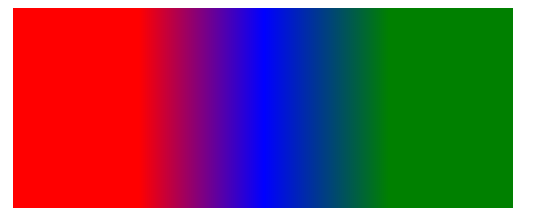
to right,

red 25%, /\* 25%-50%的区域是red向blue渐变 \*/

blue 50%, /\* 50%-75% 是blue向green渐变 \*/

green 75% /\* 75%-100% 是green向green渐变 \*/

)



background-image: linear-gradient (

to right,

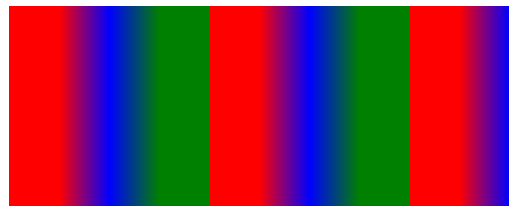
red 25%, /\* 25%-50%的区域是red向blue渐变 \*/

blue 50%, /\* 50%-75% 是blue向green渐变 \*/

green 75% /\* 75%-100% 是green向green渐变 \*/

)

background-size: 200px 100px;



没有设置background-size属性时，百分比是相对于父元素的宽高

设置了background-size属性时，百分比是相对于background-size设置的宽高

径向渐变

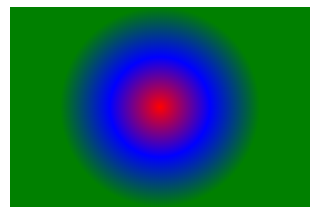
background-image: radial-gradient (

100px at center, /\* 圆心在容器中心 , 半径为100px \*/

red,

blue

)



通过 at top left

at 20px 50px

设置圆心的位置

只设置一个值时100px时，默认垂直半径也是100px

background-origin: content-box;

表示背景图起源，从盒子原本(不包括padding)区域的左上角开始向四周平铺，且平铺到padding区域

没加这个属性则是从盒子(若有padding则从padding区域)的左上角开始向四周平铺

background-clip: content-box;

背景图裁剪,只显示盒子原本(不包括padding)区域的背景平铺图，

padding区域的被隐藏了

background: url() content-box; 等价于下面的

background-image: url();

background-origin: content-box;

background-clip: content-box;

两者都作用时，则是表示从盒子原本区域(不包括padding区域)的左上角向四周平铺，且平铺到padding区域的被隐藏

若盒子没有padding属性，则这两者对盒子没影响

行高 line-height

行高 = 文字大小 + 上下间距

- 通过设置文字的行高，改变的就是文字的上下间距

- 不设置行高时，文字的大小也会影响行高

- 文字字体也会影响行高

font-size: 16px;

实际行高

line-height: 20px 20 px

20rem; 20\*16 px (设置的行高\*文字大小)

30% 16\*30 px (设置的行高\*文字大小)

1.5 16\*1.5 px (设置的行高\*文字大小)

当父元素设置行高时，且font-size: 18px;

父元素(行高) 子元素(行高)

20px 20px（直接继承）

20em 18\*20px (父元素文字大小\*父元素的行高)

（与子元素的文字大小无关）

10% 18\*10px (父元素文字大小\*父元素的行高)

（与子元素的文字大小无关）

2 2(父元素的行高)\*子元素文字大小

（先继承后计算）

过渡

transition-property: all, background-color, ...

transition-duration: 1s 过渡完成的时间

transition-delay: 1s 1s后执行过渡

transition-timing-function: ease | ease-in | ease-out | ease-in-out | linear

默认 匀速

联写 transition: all 1s linear 1s;

过渡可以放到开始的状态，也可以放到结束状态，一般放到开始状态里

css3为transition提供了一组过渡事件,可以很方便的监听过渡过程

webkitTransitionStart 过渡开始

webkitTransitionEnd 过渡结束

webkitTransitionIteration 过渡重复执行

2D转换

旋转

transform: rotate(60deg) 以盒子正中心顺时针旋转60度

改变圆心的位置

transform-origin: top left 以左上角为圆心旋转

100px 200px

位移

transform: translate(400px,200px);

X方向 ，y方向

transform: translate(50%，50%);

百分比相对于自己的宽、高

设置一个值，就只是水平方向位移

缩放

transform: scale(0.8);

一个值，表示x、y方向同时缩放相同大小

倾斜

transform: skew(30deg,60deg);

一个值，x方向

两个值，x、y方向

联写： transform: translate() rotate() skew()

3D变形

位移

transform: translateX()

transform: translateY()

transform: translateZ()

perspective: 600px; 透视 （放在父元素上）

transform-style: preserve-3D; 将平面图行转换为立体图形

旋转

transform: rotateX()

transform: rotateY()

transform: rotateZ()

transform-origin: 0 20px; 改变X旋转轴的上下位移

transform-origin: 20% 0; 改变Y旋转轴的左右位移

缩放

transform: scaleX()

Y

Z

补充

backface-visibility: hidden; 背面不可变

animation 动画

1、定义动画

@keyframes yidong {

from {

left: 100px;

}

to {

left: 600px;

}

}

// 定义动画(可用百分比)(比分比是时间)(from对应0%，to对应100%)

0% {

left:100px;

}

90% {

left:150px;

}

100% {

left:600px;

}

2、使用animation属性调用

animation: run 10s linear 0s;

animation: run 10s linear 0s infinite alternate forwords;

run 运动名称

10s 运动时间

linear 缓冲ease不匀速

linear匀速

0s 延迟时间

infinite 执行次数

infinite 无限

alternate 来回运动

forwords 停留在原地

css3为animation提供了一组动画事件,可以很方便的监听动画过程

webkitAnimationStart 动画开始

webkitAnimationEnd 动画结束

webkitAnimationIteration 动画重复执行

网页优化

内容移出 text-indent: -5000em | padding-tap: 100px;

浮动

特点：1、不占位置（脱标）

2、浮动的元素会转换为行内块

3、浮动只影响后面的元素

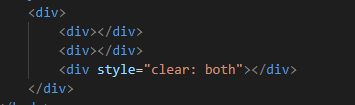
4、浮动的元素以顶部为基准对齐

5、浮动找浮动，不浮动找不浮动

使用：为了实现图片和文字环绕效果 （图片设置浮动，<p>文字</p>不设置浮动，图片会压在p元素上，但文字不会被图片压着，会在图片范围的之外）

清除浮动

1、在父元素的最后面加一个clear: both的盒子（不推荐使用，会产生冗余代码）

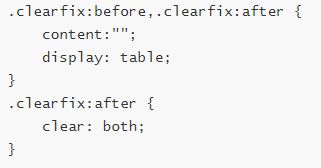


2、使用后伪元素清除浮动



3、双伪元素清除浮动

给浮动元素的父元素使用双伪元素清除浮动



3、给父元素设置overflow: hidden;

如果父元素中有定位的子元素，且该子元素超出了父元素，则不推荐使用这方法

4、给浮动元素的父元素设置高度

外边距塌陷

嵌套的盒子，若给子盒子设置垂直方向外边距时（如 margin-top: 10px）,会把父盒子也带下10px

解决方式：1、给父盒子设置边框

2、父盒子设置 overflow: hidden（推荐使用）

box-sizing: content-box | border-box | inherit

默认值 从父元素继承

css初始化

body, p, h1, h2, h3, h4, h5, h6, ul, ol , li, dl, dt, dd {

margin: 0;

pdding: 0;

font-size: 16px;

font-family: “宋体”;

color: #333;

}

.cleafix::after {

content: “”;

height: 0;

line-height: 0px;

display: block;

clear: both;

visibility: hidden;

}

input {

border: none;

outline-style: none;

padding: 0;

}

=============

移动端

移动端浏览器

基于webkit或 blink内核：

qq浏览器、百度浏览器、Safari浏览器（苹果上的） uc浏览器

视口 (viewport)

移动端特有， 是一个虚拟的区域，承载网页的

承载关系：浏览器 ---> viewport ----> 网页

一般来讲我们都会将视口(viewport)的宽度设置为设备宽度

<meta name="viewport" content="width=device-width, initial-scale=1.0">

width 设置宽度 （device-width 为当前设备宽度）

height 设置高度

initial-scale 设置默认的缩放比例

user-scalable 设置是否允许用户自行缩放

maximum-scale 设置最大缩放比例

minimum-scale 设置最小缩放比例

\* 布局视口 (Layout viewport)

默认一般为980px

布局视口不会受到缩放的影响，缩放不会导致页面重排渲染，对于移动端宝贵的性能来说非常重要。

布局视口的宽度可以通过js的document.documentElement.clientWidth获取

在桌面端获取的宽度是 980

若设置为为理想视口后，获取的宽度就是设备的宽度

例：ip5 ------ 320

ip6 ------ 375

ip6Plus --- 414

一般来讲我们都会将布局视口（Layout viewport）的宽度设为设备宽度（视觉视口宽度），此时的布局视口的状态我们就称为理想视口（ideal viewport）

<meta name="viewport" content="width=device-width, initial-scale=1.0">

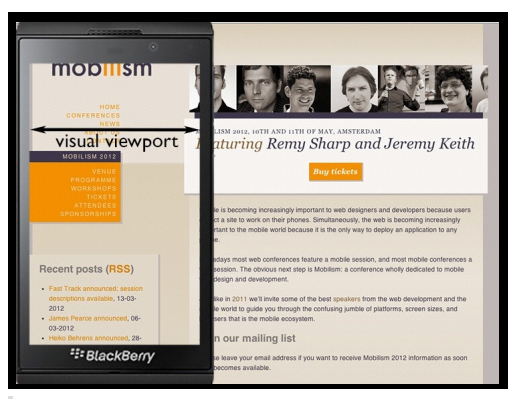
ip6

window.screen.height // 375

window.screen.height // 667

\* 视觉视口： 手持设备的物理屏幕的可视区域。

对应的javascript属性是window.innerWidth/Height



用户可以通过缩放来操作视觉视口，同时不会影响布局视口。布局视口还是保持在原来的宽度。

视觉视口的区域和设备的屏幕一样宽，并且它的CSS像素的数量会随着用户缩放而改变

\* 理想视口(ideal viewport)

1、不需要用户缩放和横向滚动条就能正常的查看网站的所有内容；

2、显示的文字的大小是合适，比如一段14px大小的文字，不会因为在一个高密度像素的屏幕里显示得太小而无法看清，理想的情况是这段14px的文字无论是在何种密度屏幕，何种分辨率下，显示出来的大小都是差不多的。当然，不只是文字，其他元素像图片什么的也是这个道理。

理想视口的宽度 = 设备像素 / dpr（设备像素比）

也就是当布局视口的宽度 = 设备独立像素的宽度时就是理想视口。

屏幕尺寸： 指屏幕的对角线的长度

iPhone4 3.5 英寸

iPhone5 4

屏幕分辨率：指在横纵上的像素点数

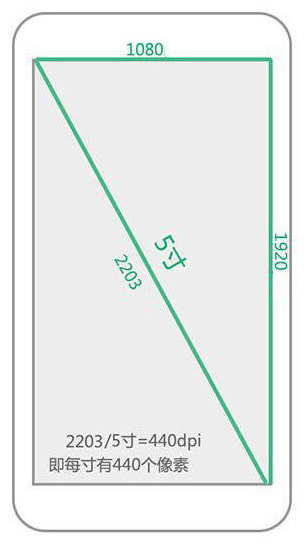
iphone4 640\*960

iphone5 640\*1136

iphone6 750\*1334

屏幕像素密度（dpi或ppi）: 屏幕上每英寸可以显示的像素点的数量，越多图像越清晰

dpi或ppi = 对角线分辨率/屏幕尺寸



像素

设备像素：设备屏幕的物理像素，任何设备的物理像素的数量都是固定的。

屏幕渲染图像的最小单位，其值大小决定了屏幕渲染图像的品质。

iPhone5 的分辨率是640x1136px 横向上有640个物理像素

css像素：为Web开发者创造的，在CSS（和JavaScript）中使用的一个抽象的层

width:200px 的元素跨越了 200 个CSS像素。

1个css像素可以对应多个物理像素

CSS像素等于多少个设备像素取决于屏幕的特性（是否高密度）和用户进行的缩放。

当用户放大的越大，一个CSS像素覆盖的设备像素就越多，因此这个元素不一定会跨越css设置等值的设备像素。

在旧的屏幕上，当缩放程度为100%时，一个CSS像素等于一个设备像素。

在高密度屏幕上，例如苹果的视网膜屏幕，一个CSS像素跨越了多个设备像素

px 是相对长度单位，相对于设备像素（物理像素）。

假设老马电脑的物理分辨率是1024\* 768。 如果我故意设置操作系统分辨率为512\*384（水平和垂直各缩小1倍），那么此时css定义的1px像素的盒子在屏幕中的显示的宽度比原来高分辨率的宽度增加一倍，所以CSS中的像素只是相对，不是绝对的。

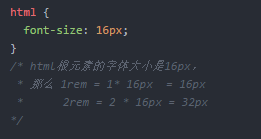
iPhone 5使用的是Retina视网膜屏幕，横向css像素是320px，但是实际物理像素是640，所以水平方向就会有2个物理像素点对应1个css像素，垂直也是两倍的关系，也就是1个css像素：由水平2个物理像素点和垂直2个像素点也就是（2乘2=4）4个物理像素点 显示1px宽1px高的一个css像素。



em 是相对长度单位。相对于当前对象内文本的字体尺寸。如果当前行内文本的字体尺寸未被设置，则相对于浏览器的默认字体尺寸。

1em = 1个字体的宽度

rem CSS3新增的，仍然是相对长度单位，相对于HTML根元素的字体尺寸。



vw 1vm等于视口宽度（viewport width）的百分之一, css3新增的。

vh 1vh等于视口高度（viewport height）的百分之一

设备独立像素（DIP）：在移动Web开发中就是指的CSS像素

设备屏幕像素比（dpr）： = 设备像素 / CSS像素

window.devicePixelRatio;

iphone5为例：水平物理像素640 页面缩放100%时，水平css像素为320px，则dpr = 640 / 320 = 2

DPR不一定都是整数，尤其是android设备十分的碎片化！

在dpr大于1的屏幕上的图片显示会模糊

移动端适配（及初始化设置）

常见适配方案：

\* 方案一：

1、固定屏幕为理想视口宽度

2、少许的媒体查询设置字体

3、水平百分比布局（流式布局）

注意：

padding设置百分比 无论水平的padding还是竖直的padding,指的是父元素的width的百分之多少

margin设置百分比 无论水平的margin还是竖直的margin,指的是父元素的width的百分之多少

border不能用百分比表示

能够设置百分比的属性是width height padding margin

其他如border、font-size不能设置百分比

水平方向部分也可以使用弹性布局

\* 方案二：rem解决方案

1、rem的大小取值：根据页面的dpr(设备像素比)动态改变

2、rem的取值： 1rem = 100px 或者 1rem = 1/10 \* 理想视口的宽度

3、chrome浏览器字体小于12px（会被重置为12px)

\* 方案三：固定设计稿的宽度+根据设备动态适配缩放

1、直接按照设计稿编写固定尺寸元素

2、页面加载完成后用js动态根据dpr改变页面的缩放值

推荐使用： flexible方案

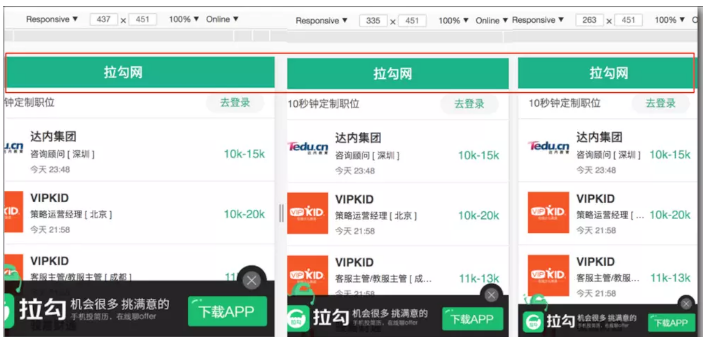
百分比与固定高度布局方案

设置屏幕为理想视口，然后通过水平百分比布局或者弹性布局，垂直方向一般用固定像素。

优点：布局快速简单方便、在移动设备中水平表现良好差异不大。

缺点：由于垂直方向像素恒定，可能水平很宽的屏幕会被拉伸变形严重，另外在高倍屏幕上1像素可能被多倍的物理像素显示，会变的非常粗。

看案例： 拉勾网移动端首页顶部的logo区域，不管如何变化浏览器的宽度，高度不变化，宽度自适应。



/\* 宽度默认不设置，就是100% \*/

height: 45px;

line-height: 45px;

background-color: #00b38a;

color: #fff;

font-size: 1.8rem;

/\* 也可以根据媒体查询适当调整字体大小 \*/

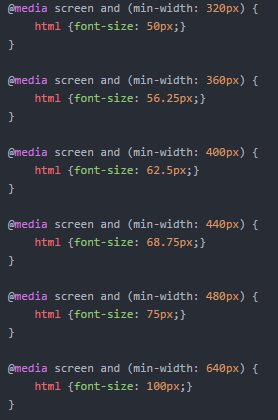
text-align: center;

position: relative;

媒体查询改变根元素的字体大小

CSS3新增加了媒体查询功能，也就是可以根据设备媒体的属性做相关的判断，然后执行不同的CSS规则。

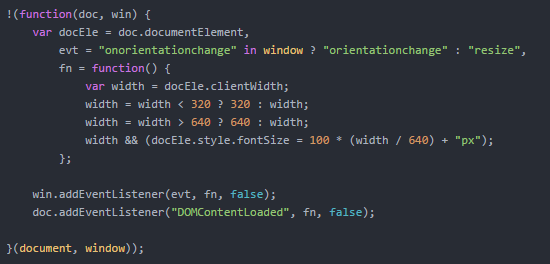
/\* 媒体查询以@media开头，然后后面可以跟上判断的条件，比如：screen表示屏幕设备, and是并且的意思， min-width是最小宽度的意思。 \*/



缺点：媒体查询不能完全枚举，毕竟android的屏幕尺寸碎片化严重，各种尺寸都有，不能完全覆盖，只能大体覆盖。如果想要精确覆盖要么通过js实现，要么用最新的浏览器已经支持的calc实现。

js实现动态改变根元素的字体大小

假设，640px的设备 1rem = 100px  
公式： rem = document.clientWidth / 640 \* 100px;



1、头像在不同设备上一般都会设置固定像素的大小

2、宽度可能超过50%的盒子尽量用百分比或者弹性盒子布局

3、高度和宽度需要同时根据屏幕自动缩放的时候，推荐使用rem布局

开发的时候根据设计搞完全还原像素，然后根据屏幕的宽度通过js动态改变页面的缩放，恰好是理想视口的大小。

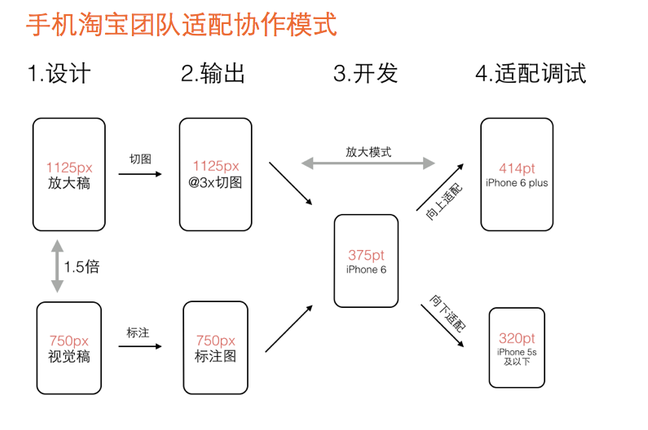
淘宝的flexible方案最靠谱，应用也最广泛。

整个手淘设计师和前端开发的适配协作基本思路是：

1、选择一种尺寸作为设计和开发基准

2、定义一套适配规则，自动适配剩下的两种尺寸(其实不仅这两种)

3、特殊适配效果给出设计效果



手淘设计师常选择iPhone6作为基准设计尺寸，交付给前端的设计尺寸是按750px \* 1334px为准(高度会随着内容多少而改变)。前端开发人员通过一套适配规则自动适配到其他的尺寸。

1、根据屏幕像素比（dpr）来设置视口

var deviceWidth = document.documentElement.clientWidth;

var dpr = window.devicePixelRatio || 1;

var scale = 1 / dpr;

document.querySelector("meta[name = 'viewport']").setAttribute("content", "width="+ dpr \* deviceWidth +", initial-scale="+ scale +", maximum-scale="+ scale +", minimum-scale="+ scale +", user-scalable=no");

2、设置html的font-size样式

将设计稿(ip6为例)划分为7.5份，每份为100px

var rem = deviceWidth \* dpr / 7.5;

document.querySelector("html").style.fontSize = rem +"px";

document.querySelector("html").setAttribute("data-dpr", dpr);

3、设置某些元素的font-size的值

.text { font-size: 16px; }

[data-dpr = "2"] .text { font-size: 32px; }

[data-dpr = "2.5"] .text { font-size: 40px; }

[data-dpr = "3"] .text { font-size: 48px; }

[data-dpr = "4"] .text { font-size: 64px; }

网易移动端适配

1、将设计稿(ip6为例)划分为7.5份，每份为100px

2、设置html的font-size的值

document.documentElement.style.fontSize = document.documentElement.clientWidth / 7.5 + "px";

ip6 device-width = 375 fontSize = 1rem = 375 / 7.5 = 50px

ip5 device-width = 320 fontSize = 1rem = 320 / 7.5 = 42.66667px

ip6p device-width = 414 fontSize = 1rem = 414 / 7.5 = 55.2px

在ip6中，实际计算px和rem时，

在宽高方面按1rem = 100px计算，为了和设计稿的尺寸一样

字体的font-size则要按 1rem = 50px 计算, 或者用媒体查询的方式

@media screen and (max-width: 321px) {

.m-navList { font-size: 15px }

}

@media screen and (min-width: 321px) and (max-width: 400px) {

.m-navList { font-size: 16px }

}

@media screen and (min-width: 400px) {

.m-navList { font-size: 18px }

}

初始化设置：

1、<meta name="viewport" content="width=device-width, initial-scale=1.0">

3、移动端浏览器会有遇见点击出现高亮的效果，通常把这个点击的颜色设置成透明的

tap-highlight-color: transparent

\*,

\*::before,

\*::after {

margin: 0;

padding: 0;

tap-highlight-color: transparent;

-webkit-tap-highlight-color: transparent;

}

1. 盒子模型

box-sizing: border-box

\*,

\*::before,

\*::after {

tap-highlight-color: transparent;

-webkit-tap-highlight-color: transparent;

box-sizing: border-box; //保证盒子大小不变，内容可变

-webkit-box-sizing: border-box;

}

1. 清除input,textarea默认样式

input,textarea {

border: none;

outline: none;

resize: none; //不允许改变尺寸

-webkit-appearance: none; // 元素的外观设置为没有任何样式

}

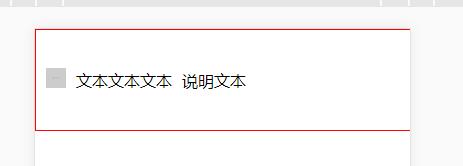
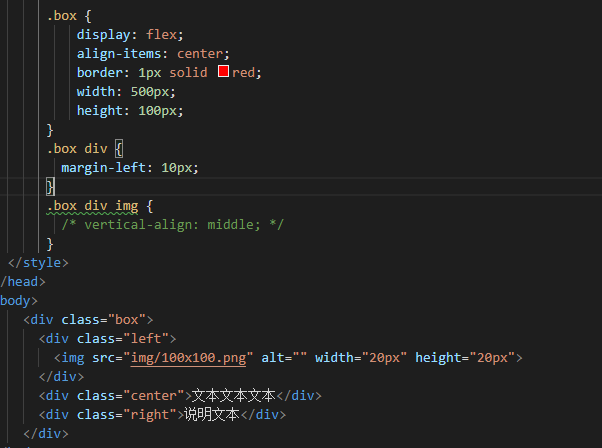
6、防止移动设备在大尺寸和小尺寸的屏幕下不会过度缩放失去清晰度和布局错落

max-width: 640px; //一般设计图使用的是640px

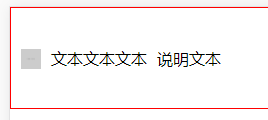
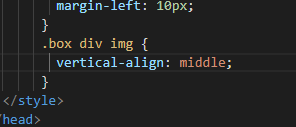
min-width: 320px;

display: flex 问题

1. align-items: center; 垂直居中的问题



img加上 vertical-align: middle 时才能对齐同一行的文本



响应式开发

<link rel="stylesheet" type="text/css" href="css/index1200.css" media="screen and (min-width:1200px)" >

<link rel="stylesheet" type="text/css" href="css/index980.css" media="screen and (min-width:980px) and (max-width:1200px)" >

<link rel="stylesheet" type="text/css" href="css/index640.css" media="screen and (min-width:640px) and (max-width:980px)" >

js实现响应式开发方法

var rwdlink = document.getElementById("");

setCss();

window.onresize = setCss;

function setCss() {

var windowWidth = document.documentElement.clientWidth;

if(windowWidth>=1200) {

rwdlink.href = "";

} else if(windowWidth>=980) {

rwdlink.href = "css/index980.css";

} else {

rwdlink.href = "css/index640.css";

}

}

=======

HTML5新特性

1、canvas 绘画

2、video 和 audio

3、本地离线存储

localStorage 长期存储数据，浏览器关闭后数据不丢失；

sessionStorage 数据在浏览器关闭后自动删除

区别：

localStorage属于永久性存储，而sessionStorage属于当会话结束的时候，sessionStorage中的键值对会被清空

1、html5拖拽事件

源对象

ondragstart 拖拽开始

ondrag 拖拽过程中

ondragend 拖拽结束

目标对象

ondragenter 进入目标元素

ondragover 在目标元素上移动

ondragleave 离开目标元素

ondrop 鼠标在目标元素上释放

2、html5的数据交互对象

本地的图片默认可以拖拽，所以拖拽本地图片时不需要源对象

如何在拖拽对象和目标对象之间传递数据 e.dataTransfer

backdrop-filter: blur(10px);

//只支持ios端；只作用于当前元素；为背景添加模糊效果；如果目标元素内包裹着其他内容 则应用filter属性；

filter: blur(10px);

//兼容性比较好，不仅仅作用于当前元素，后代元素也会继承这个属性，作用于一个空背景元素没有效果

（效果其实还是通过作用于具体的img元素才实现模糊效果，如果单单作用于一个空元素背景 则没有效果）

**localStorage**

**优势:**

1. localStorage拓展了cookie的4K限制, localStorage中一般浏览器支持的是5M大小
2. localStorage可以将第一次请求的数据直接存储到本地，这个相当于一个5M大小的针对于前端页面的数据库，相比于cookie可以节约带宽，只有在高版本的浏览器中才支持

**局限：**

1、各浏览器对localStorage的大小不统一

2、目前所有的浏览器中都会把localStorage的值类型限定为string类型，这个在对我们日常比较常见的JSON对象类型需要一些转换

3、localStorage在浏览器的隐私模式下面是不可读取的

4、localStorage本质上是对字符串的读取，如果存储内容多的话会消耗内存空间，会导致页面变卡

5、localStorage不能被爬虫抓取到

使用

判断浏览器是否支持

if(!window.localStorage){

alert("浏览器不支持localstorage");

return false;

}else{

//主逻辑业务

}

1、localStorage的写入（有三种方法）

var storage = window.localStorage;

//写入a字段

storage["a"] = 1;

//写入b字段

storage.b = 2;

//写入c字段

storage.setItem("c",3);

console.log(typeof storage["a"]); // string

console.log(typeof storage["b"]); // string

console.log(typeof storage["c"]); // string

2、localStorage的读取

var storage = window.localStorage;

//第一种方法读取

var a = storage.a;

console.log(a);

//第二种方法读取

var b = storage["b"];

console.log(b);

//第三种方法读取（官方推荐的方法）

var c = storage.getItem("c");

console.log(c);

3、localStorage的更改

var storage = window.localStorage;

storage["a"] = 1;

storage.a = 4;

console.log(storage.a); // 4

4、localStorage的删除

var storage = window.localStorage;

storage.a = 1;

storage.setItem("c",3);

console.log(storage);

storage.clear();

console.log(storage);

将localStorage中的某个键值对删除

var storage = window.localStorage;

storage.a=1;

storage.setItem("c",3);

console.log(storage); { a: 1, c: 3 }

storage.removeItem("a");

console.log(storage.a); // undefined

5、localStorage的键获取

使用key()方法，向其中出入索引即可获取对应的键

var storage=window.localStorage;

storage.a=1;

storage.setItem("c",3);

for(var i=0;i<storage.length;i++){

var key=storage.key(i);

console.log(key);

}

6、 一般我们会将JSON存入localStorage中，但是在localStorage会自动将localStorage转换成为字符串形式

这个时候我们可以使用JSON.stringify()这个方法，来将JSON转换成为JSON字符串

var storage=window.localStorage;

var data={

name:'xiecanyong',

sex:'man',

hobby:'program'

};

var d=JSON.stringify(data);

storage.setItem("data",d);

console.log(storage.data)

7、读取之后要将JSON字符串转换成为JSON对象，使用JSON.parse()方法

var storage=window.localStorage;

var data={

name:'xiecanyong',

sex:'man',

hobby:'program'

};

var d=JSON.stringify(data);

storage.setItem("data",d);

//将JSON字符串转换成为JSON对象输出(其他类型读取出来也要进行转换)

var jsonStr = storage.getItem("data");

var jsonObj = JSON.parse(json);

console.log(typeof jsonObj); //object