**复习**

**0. svn**

**1. 定位参照于谁块来定位**

**没有定位 :包含块**

**相对 : 元素本来的位置**

**绝对 :包含块**

**如果最近的祖先元素中存在定位元素，则这个定位元素就是包含块**

**如果没有，包含块为初始包含块**

**固定 : 视口**

**2. 什么是初始包含块**

**是一个视窗大小的矩形，不等于视窗**

**3. left top right bottom width height**

**默认值为auto**

**margin padding**

**默认值 0**

**boder-width**

**如果不存在border-style**

**4. 百分比参照于谁**

**width margin padding:包含块的width**

**height:包含块的height**

**left:包含块的width**

**top :包含块的height**

**5.浮动**

**浮动提升半层**

**6.三列布局**

**两边固定，当中自适应**

**中间列要完整的显示**

**中间列要优先加载**

**定位**

**浮动**

**圣杯**

**双飞翼**

**7.margin为负值（margin不影响元素的位置）**

**负值:将元素的边界往里收**

**正值:将元素的边界往外扩**

**8.伪等高布局**

**9.fixed**

**怎么使用绝对定位来模拟固定定位**

**1.禁止系统滚动条**

**2.将滚动条加给body**

**3.让body的尺寸变为视口的尺寸**

* **圣杯布局**

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

<!--

1.两边固定 当中自适应

2.当中列要完整显示

3.当中列要优先加载

-->

<!--

浮动: 搭建完整的布局框架

margin 为赋值:调整旁边两列的位置(使三列布局到一行上)

使用相对定位:调整旁边两列的位置（使两列位置调整到两头）

-->

<style type="text/css">

\*{

margin: 0;

padding: 0;

}

body{

min-width: 600px;

}

#content{

padding: 0 200px;

}

#header,#footer{

height: 20px;

text-align: center;

border: 1px solid deeppink;

background: gray;

}

#content .middle{

float: left;

width: 100%;

background: pink;

/\*padding: 0 200px;\*/

}

#content .left{

position: relative;

left: -200px;

margin-left: -100%;

float: left;

width: 200px;

background: yellow;

}

#content .right{

position: relative;

right: -200px;

margin-left: -200px;

float: left;

width: 200px;

background: yellow;

}

.clearfix{

\*zoom: 1;

}

.clearfix:after{

content: "";

display: block;

clear: both;

}

</style>

</head>

<body>

<div id="header">header</div>

<div id="content" class="clearfix">

<div class="middle">

<h4>middle</h4>

</div>

<div class="left">left</div>

<div class="right">right</div>

</div>

<div id="footer">footer</div>

</body>

</html>

* 双飞翼布局

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title></title>

<!--两组实现的对比:

1.俩种布局方式都是把主列放在文档流最前面，使主列优先加载。

2.两种布局方式在实现上也有相同之处，都是让三列浮动，然后通过负外边距形成三列布局。

3.两种布局方式的不同之处在于如何处理中间主列的位置：

圣杯布局是利用父容器的左、右内边距+两个从列相对定位；

双飞翼布局是把主列嵌套在一个新的父级块中利用主列的左、右外边距进行布局调整

-->

<style type="text/css">

\*{

margin: 0;

padding: 0;

}

body{

min-width: 600px;

}

/\*头部 脚部样式\*/

#header,#footer{

border: 1px solid;

background: gray;

text-align: center;

}

/\*三列的伪等高布局\*/

#content .middle,#content .left,#content .right{

/\*padding-bottom:10000px ;

margin-bottom: -10000px;\*/

height: 50px;

line-height: 50px;

float: left;

}

/\*双飞翼布局\*/

#content{

overflow: hidden;

}

#content .middle{

width: 100%;

background: deeppink;

}

#content .middle .m\_inner{

padding: 0 200px;

}

#content .left,#content .right{

background: pink;

width: 200px;

text-align: center;

}

#content .left{

margin-left: -100%;

}

#content .right{

margin-left: -200px;

}

</style>

</head>

<body>

<div id="header">

<h4>header</h4>

</div>

<div id="content">

<div class="middle">

<div class="m\_inner">

middle

</div>

</div>

<div class="left">left</div>

<div class="right">right</div>

</div>

<div id="footer">

<h4>footer</h4>

</div>

</body>

</html>

CSS3

Day1

1.Cascading style sheets

2.样式表的组成

规则

选择器+声明块

声明

CSS合法的属性名+属性值

3.浏览器渲染样式表的顺序

从右往左

4.选择器

基本选择器及其扩展

\* . # 后代 组合（#test.pink）

> + ~ 分组（，结合符）

属性选择器

存在与值 属性选择器

[attr] [attr="val"] [attr~="val"](只认空格)

子串值 属性选择器

^ $ \* |(val val-)

伪类与伪元素选择器

链接伪类

:link :visited :target(css实现选项卡)

动态伪类

:hover :active(lvha)

表单伪类

:disabled :enabled :checked(自定义单选按钮) :focus

结构性伪类

ele:nth-child(index)

ele:nth-of-type(index) 以元素为中心

区别：

1.nth-child找到第index个子元素 这个子元素必须满足ele的规则

nth-of-type找到底index个ele子元素

2.nth-of-type以元素为中心

注意：

index可以是变量n（只能是n 0到正无穷）

odd：奇数

even：偶数

伪元素

::after

::before

CSS声明的优先级

层叠

先按来源进行刷选

如果来源相同，按选择器的特殊性继续刷选

选择器的特殊性如果相同，按顺序继续刷选

5.自定义字体

@font-face

字体图标

1.制作一套矢量图

2.将矢量图与字符进行绑定

3.使用工具或者站点生成一套字体

4.最终使用

字体兼容处理网站

https://www.fontsquirrel.com/tools/webfont-generator

icomoon字体图标

https://icomoon.io/#home

6.文本新增样式

文本阴影

怎么溢出显示省略号

white-space=no-wrap

overflow=hidden

text-overflow=ellipsis

包裹区域必须不能让子元素撑开

* Day2

1.盒模型新增样式

box-shadow

关键字(内 外阴影)

length(x轴的偏移量)

length(y轴的偏移量)

length(模糊程度)

length(阴影面积)

color(阴影颜色)

text-shadow

length(x轴的偏移量)

length(y轴的偏移量)

length(模糊程度)

color(阴影颜色)

2.倒影（webkit内核 文字描边 背景镂空）

渐变倒影

3.box-sizing

border-box：代表元素上设置的width和height表示的是border-box尺寸

content-box：代表元素上设置的width和height表示的是content-box尺寸

4.层级

a.浮动提升半层，只有在浮动的情况下，才需要考虑元素分两层

定位元素提一层

相对定位会在文档流你有残留

b.z-index为1怎么都会比a高;z-index为-1怎么都会比a低

5.包含块

初始包含块：和视窗大小位置尺寸一样的矩形

滚动系统滚动条会不会影响初始包含块的位置？

会

禁止系统滚动条

html,body{

height:100%;

overflow:hidden

}

怎么解决ie6下固定定位失效的问题？

用绝对定位来模拟固定定位

1.禁止系统滚动条

2.将滚动条作用在最外层的包裹器上或者在body上

3.因为移动包裹器或者body身上的滚动条并不会影响初始包含块的位置

所以一个按照初始包含块定位的元素就不会产生移动

6.边框图片、渐变

7.背景

css2

background-color 平铺整个border-box

background-image 默认从padding-box开始绘制，从border-box开始剪裁

css3中有多背景，默认绘制时尺寸是自己的位图像素

background-repeat

控制平铺与否

background-position

控制背景图片在背景区域中的位置

px

百分比

参照于背景区域减去背景图片的位图像素值

background-attachment

scroll：默认值，背景图不会随着元素滚动条的滚动而滚动

fixed：背景图铺在视口中固定定位了

css3

background-origin

background-clip

background-size

图片是自适应的

8.如何实现一张图片的垂直水平居中

body:after{

content: "";

display: inline-block;

height: 100%;

vertical-align: middle;

}

img{

vertical-align: middle;

}

* Day3

1.过渡

transition-property

指定过渡动画的属性（并不是所有的属性都可以动画）

transition-duration

指定过渡动画的时间（0也要带单位）

transition-timing-function

指定过渡动画的形式（贝塞尔）

transition-delay

指定过渡动画的延迟

transition

第一个可以被解析成时间的值会赋给transition-duration

transtionend事件（DOM2）

在每个属性完成过渡时都会触发这个事件

当属性值的列表长度不一致时

跟时间有关的重复列表

transition-timing-function使用默认值

2.2D变换（transform）

rotate

旋转

translate

平移

skew

斜切

scale

缩放

变换组合!

顺序是从右往左的，变换的底层其实就是矩阵的运算

基点的变换

transform-origin

* day4

百分比都是参照于谁？

元素垂直水平居中方案

已知高宽

1.position=absolute,lrtb=0,margin=auto,包含块一定的是容器。（绝对定位盒模型的特性）

left+right+w+p+m = 包含块的宽度

top+bottom+h+p+m = 包含块的高度

2.position=absolute,lt=50%,包含块一定的是容器,marginT/L为负的自身的一半。

3.position=absolute,lt=50%,包含块一定的是容器,transform:tranlate3d(-50%,-50%,0)

4.flex

未知高宽

1.position=absolute,lt=50%,包含块一定的是容器,transform:tranlate3d(-50%,-50%,0)

2.flex

img元素如何元素垂直水平居中

1.基线

1.3D变换

perspective，灭点

景深的写法：

2种写法。一种css属性，一种是transform属性的一个变换函数（必须在第一位）

景深的作用：

构建3D舞台，决定用户的肉眼距离屏幕的水平距离！使3D变换的元素具有近大远小的感觉

景深和灭点的关系

景深越大，灭点越远，元素变形越小

景深越小，灭点越近，元素变形越大

景深基点

景深叠加

景深是一个不可继承属性，但他可以作用于所有的后代元素

transform-style

transform-style的作用

构建3D舞台，使3d舞台有层次感

transform-style是一个不可继承属性，他只作用于子元素

backface-visibility

隐藏元素背面

一个元素分两面，但并不意味元素有厚度。在一个状态下，元素只能展现自己的一面

2.css3动画基础

animation-name：代表关键帧的名字

关键帧：

@keyframes animiationName{

keyframes-selector{

css-style;

}

}

animation-duration：一个动画周期的时长

animation-timing-function：作用于一个关键帧周期而非整个动画周期

animation-delay：代表动画的延迟（这是一个动画外的属性）

animation-iteration-count：循环关键帧的次数！（只会管理动画内的属性，动画的延迟不会被循环）

animation-direction：改变关键帧的执行方向，还会影响animation-timing-function的形式

animation-fill-mode：管理所有动画外的状态！

什么是动画外的状态

from之前

animation-delay

to之后

值

none : 动画外的状态保持在动画之前的位置

backwards ：from之前的状态与from的状态保持一致

forwards ：to之后的状态与to的状态保持一致

both ：动画外的状态与from和to的状态保持一致

animation-play-state：管理动画的运行和停止

关键帧：

@keyframes animiationName{

keyframes-selector{

css-style;

}

}

keyframes-selector可以是关键帧form（0%）和to（100%）

可以是百分比

代表的是时间的百分比（时间点）

* Day5

###steps(num,[start/end])

num:拆成多少步（当吗num为12时，整个动画最好有13帧）

start：看不见第一帧

end：看不见最后一帧

transform只能使用在块级元素上！！！

###flex捋一捋

0.flex frog练习

http://flexboxfroggy.com/

1.flex基础点

---什么是容器，什么是项目，什么是主轴，什么是侧轴？

---项目永远排列在主轴的正方向上

---flex分新旧两个版本

-webkit-box

-webkit-flex / flex

2.注意点

---我们都知道新版本的flex要比老版本的flex强大很多，有没有必要学习老版本的flex？

移动端开发者必须要关注老版本的flex

因为很多移动端设备内核低只老版本的flex

---老版本的box通过两个属性四个属性值控制了主轴的位置和方向

新版本的flex通过一个属性四个属性值控制了主轴的位置和方向

3.老版本

容器

容器的布局方向

-webkit-box-orient:horizontal/vertical

控制主轴是哪一根

horizontal：x轴

vertical ：y轴

容器的排列方向

-webkit-box-direction：normal/reverse

控制主轴的方向

normal：从左往右（正方向）

reverse：从右往左（反方向）

富裕空间的管理

只决定富裕空间的位置，不会给项目区分配空间

主轴

-webkit-box-pack

主轴是x轴

start：在右边

end: 在左边

center：在两边

justify：在项目之间

主轴是y轴

start：在下边

end：在上边

center：在两边

justify：在项目之间

侧轴

-webkit-box-algin

侧轴是x轴

start：在右边

end： 在左边

center：在两边

侧轴是y轴

start：在下边

end： 在上边

center：在两边

项目

弹性空间管理

-webkit-box-flex：弹性因子（默认值为0）

4.新版本

容器

容器的布局方向

容器的排列方向

flex-direction

控制主轴是哪一根，控制主轴的方向

row; 从左往右的x轴

row-reverse;从右往左的x轴

column; 从上往下的y轴

column-reverse;从下往上的y轴

富裕空间的管理

只决定富裕空间的位置，不会给项目区分配空间

主轴

justify-content

flex-start： 在主轴的正方向

flex-end: 在主轴的反方向

center： 在两边

space-between： 在项目之间

space-around： 在项目两边

侧轴

align-items

flex-start：在侧轴的正方向

flex-end： 在侧轴的反方向

center： 在两边

base#ne 基线对齐

stretch 等高布局（项目没有高度）

项目

弹性空间管理

flex-grow：弹性因子（默认值为0）

5.新版本新增的

容器

flex-wrap：控制的是侧轴的方向

nowrap 不换行

wrap 侧轴方向由上而下 （flex-shrink失效）

wrap-reverse 侧轴方向由下而上 （flex-shrink失效）

align-content：多行/列时侧轴富裕空间的管理（把多行/列看成一个整体）

flex-flow：flex-direction和flex-wrap的简写属性

本质上控制了主轴和侧轴分别是哪一根，以及他们的方向

项目

order:控制项目的排列顺序

align-self：项目自身富裕空间的管理

flex-shrink：收缩因子（默认值为1）

flex-basis：伸缩规则计算的基准值（默认拿width或height的值）

6.伸缩规则

flex-basis（默认值为auto）

flex-grow（默认值为0）

可用空间 = (容器大小 - 所有相邻项目flex-basis的总和)

可扩展空间 = (可用空间/所有相邻项目flex-grow的总和)

每项伸缩大小 = (伸缩基准值flex-basis + (可扩展空间 x flex-grow值))

flex-shrink（默认值为1）

--.计算收缩因子与基准值乘的总和

var a = 每一项flex-shrink\*flex-basis之和

--.计算收缩因数

收缩因数=（项目的收缩因子\*项目基准值）/第一步计算总和

var b = (flex-shrink\*flex-basis)/a

--.移除空间的计算

移除空间= 项目收缩因数 x 负溢出的空间

var c = b \* 溢出的空间

7.侧轴富裕空间的管理

单行单列

align-items

align-self（优先级高）

多行多列

align-content

* Day6

###flex捋一捋

0.flex frog练习

http://flexboxfroggy.com/

1.flex基础点

---什么是容器，什么是项目，什么是主轴，什么是侧轴？

---项目永远排列在主轴的正方向上

---flex分新旧两个版本

-webkit-box

-webkit-flex / flex

2.注意点

---我们都知道新版本的flex要比老版本的flex强大很多，有没有必要学习老版本的flex？

移动端开发者必须要关注老版本的flex

因为很多移动端设备内核低只老版本的flex

---老版本的box通过两个属性四个属性值控制了主轴的位置和方向

新版本的flex通过一个属性四个属性值控制了主轴的位置和方向

3.老版本

容器

容器的布局方向

-webkit-box-orient:horizontal/vertical

控制主轴是哪一根

horizontal：x轴

vertical ：y轴

容器的排列方向

-webkit-box-direction：normal/reverse

控制主轴的方向

normal：从左往右（正方向）

reverse：从右往左（反方向）

富裕空间的管理

只决定富裕空间的位置，不会给项目区分配空间

主轴

-webkit-box-pack

主轴是x轴

start：在右边

end: 在左边

center：在两边

justify：在项目之间

主轴是y轴

start：在下边

end：在上边

center：在两边

justify：在项目之间

侧轴

-webkit-box-algin

侧轴是x轴

start：在右边

end： 在左边

center：在两边

侧轴是y轴

start：在下边

end： 在上边

center：在两边

项目

弹性空间管理

-webkit-box-flex：弹性因子（默认值为0）

4.新版本

容器

容器的布局方向

容器的排列方向

flex-direction

控制主轴是哪一根，控制主轴的方向

row; 从左往右的x轴

row-reverse;从右往左的x轴

column; 从上往下的y轴

column-reverse;从下往上的y轴

富裕空间的管理

只决定富裕空间的位置，不会给项目区分配空间

主轴

justify-content

flex-start： 在主轴的正方向

flex-end: 在主轴的反方向

center： 在两边

space-between： 在项目之间

space-around： 在项目两边

侧轴

align-items

flex-start：在侧轴的正方向

flex-end： 在侧轴的反方向

center： 在两边

base#ne 基线对齐

stretch 等高布局（项目没有高度）

项目

弹性空间管理

flex-grow：弹性因子（默认值为0）

5.新版本新增的

容器

flex-wrap：控制的是侧轴的方向

nowrap 不换行

wrap 侧轴方向由上而下 （flex-shrink失效）

wrap-reverse 侧轴方向由下而上 （flex-shrink失效）

align-content：多行/列时侧轴富裕空间的管理（把多行/列看成一个整体）

flex-flow：flex-direction和flex-wrap的简写属性

本质上控制了主轴和侧轴分别是哪一根，以及他们的方向

项目

order:控制项目的排列顺序

align-self：项目自身富裕空间的管理

flex-shrink：收缩因子（默认值为1）

flex-basis：伸缩规则计算的基准值（默认拿width或height的值）

6.伸缩规则

flex-basis（默认值为auto）

flex-grow（默认值为0）

可用空间 = (容器大小 - 所有相邻项目flex-basis的总和)

可扩展空间 = (可用空间/所有相邻项目flex-grow的总和)

每项伸缩大小 = (伸缩基准值flex-basis + (可扩展空间 x flex-grow值))

flex-shrink（默认值为1）

--.计算收缩因子与基准值乘的总和

var a = 每一项flex-shrink\*flex-basis之和

--.计算收缩因数

收缩因数=（项目的收缩因子\*项目基准值）/第一步计算总和

var b = (flex-shrink\*flex-basis)/a

--.移除空间的计算

移除空间= 项目收缩因数 x 负溢出的空间

var c = b \* 溢出的空间

7.侧轴富裕空间的管理

单行单列

align-items

align-self（优先级高）

多行多列

align-content

8.flex的简写属性

flex:1 (flex-basis:0% flex-grow:1 flex-shrink:1)

等分布局

###响应式布局核心 CSS3媒体查询选择器

@media 媒体类型 and(关键字) 带条件的媒体属性 and 带条件的媒体属性 {

//规则

}

媒体类型

all

screen

print

媒体属性

width：浏览器的窗口尺寸

device-width：设备尺寸

device-pixel-ratio（必须加webkit前缀）：像素比

---以上三个媒体属性可加 min 和 max 前缀

min：最小值为谁

max：最大值为谁

横竖屏切换

orientation：landscape（横屏）

orientation：portrait （竖屏）

关键字

only：处理浏览器尽量问题

and：连接一条查询规则

,：连接多条查询规则

not：取反

###多列布局

###规范

HTML

什么叫html5? 是一个强大的技术集（html5 ~ html+css+js）

CSS

什么是css3？

css3其实就是html5的一部分，而且现代前端中已经没有版本的概念，都是级别

JS

ECMASCRIPT

DOM

BOM（没有规范，window）

###预处理器（less）

变量

@

变量的延迟加载

变量是块级作用域

嵌套

&:平级

混合

什么是混合？

将一系列的规则集引入另一个规则集中（ctrl c+ctrl v）

混合的定义在less规则有明确的指定，使用.的形式来定义

普通混合（编译到原生css中的）

不带输出的混合（加双括号）

带参数的混合

带默认值的混合

匹配模式

arguments

计算

加减乘除 计算的一方带单位就可以

继承

* Day7

1.less的继承

#test{

&:extend(.father)

}

#test:extend(.father){

}

继承实质上将.father选择器和#test组合成一个选择器，

声明块使用.father的

all:继承所有和.father相关的声明块

切记父类不能定义成混合（继承不灵活性能高 混合灵活性能低）

2.less的避免编译

~"不会被编译的内容"

变量在less中属于块级作用域，延迟加载

3.bootstrap栅格系统&源码分析

流体容器

width:auto

两侧15px padding

固定容器

阈值

xs（小于768px） width：auto

sm（大于等于768px） width：720+槽宽

md（大于等于992px） width：940+槽宽

lg（大于等于1200） width：1140+槽宽

两侧15px padding

行

两侧-15px margin

列

公共样式

两侧有15px 的padding

相对定位

float

width

1~12

left

right

0~12

0：auto

margin-left

0~12

4.列排序

注意阈值上样式的设置不能跳跃

* Day8

1.width：auto width：100%

2.响应式页面

定制化：修改bootstrap的源文件

将bootstrap入口less文件作为混合使用

3.attr&prop

---什么是attribute？

html的预定义和自定义属性

---什么是property？

js对象身上的直接属性

---什么是布尔值属性，什么是非布尔值属性？

property所对应的属性值是否布尔类型

---attribute和property的同步关系

非布尔值属性

实时同步

布尔值属性

没有动过property

attribute会同步property

property不会同步attribute

动过property

attribute不会同步property

property不会同步attribute

---浏览器认谁，用户操作的是谁

property

---在jQuery中的体现

attr()

prop()

---总结

布尔值属性最好使用prop方法

非布尔值属性attr方法

---html5中有关的属性

classlist：相对于class的property(className)

add

remove

toggle

dataset:自定义属性（限制 data-x-y）的property

4.html5导学

html5的意义，到底什么是html5，html5的优势

h4与h5的对比

编码

渲染模式

mine类型

语义化标签

header

footer

section

nav

###CSS选择器 CSS3选择器规范地址： <https://www.w3.org/TR/2011/REC-css3-selectors-20110929/> CSS3选择最新选择器规范: <https://www.w3.org/TR/selectors>

!---问题---!  
1.css的全称是什么？  
2.样式表的组成  
3.浏览器读取编译css的顺序？  
​  
1.基本选择器  
/\*通配符选择器\*/ \* { margin: 0; padding: 0; border: none; }  
/\*元素选择器\*/ body { background: #eee; }  
/\*类选择器\*/ .list { list-style: square; }  
/\*ID选择器\*/ #list { width: 500px; margin: 0 auto; }  
/\*后代选择器\*/ .list li { margin-top: 10px; background: #abcdef; }

2.基本选择器扩展  
/\*子元素选择器\*/ #wrap > .inner {color: pink;}  
也可称为直接后代选择器,此类选择器只能匹配到直接后代，不能匹配到深层次的后代元素  
/\*相邻兄弟选择器\*/ #wrap #first + .inner {color: #f00;}  
它只会匹配紧跟着的兄弟元素  
/\*通用兄弟选择器\*/ #wrap #first ~ div { border: 1px solid;}  
它会匹配所有的兄弟元素(不需要紧跟)  
/\*选择器分组\*/ h1,h2,h3{color: pink;}    
此处的逗号我们称之为结合符

3.属性选择器  
/\*存在和值属性选择器\*/   
[attr]：该选择器选择包含 attr 属性的所有元素，不论 attr 的值为何。  
[attr=val]：该选择器仅选择 attr 属性被赋值为 val 的所有元素。  
[attr~=val]：表示带有以 attr 命名的属性的元素，并且该属性是一个以空格作为分隔的值列表，其中至少一个值为val。  
  
/\*子串值属性选择器\*/  
[attr|=val] : 选择attr属性的值是val（包括val）或以val-开头的元素。  
[attr^=val] : 选择attr属性的值以val开头（包括val）的元素。  
[attr$=val] : 选择attr属性的值以val结尾（包括val）的元素。  
[attr\*=val] : 选择attr属性的值中包含字符串val的元素。

4.伪类与伪元素选择器  
/\*链接伪类\*/ 注意:link，:visited，:target是作用于链接元素的！  
:link 表示作为超链接，并指向一个未访问的地址的所有锚  
:visited 表示作为超链接，并指向一个已访问的地址的所有锚  
:target 代表一个特殊的元素，它的id是URI的片段标识符  
/\*动态伪类\*/ 注意:hover，:active基本可以作用于所有的元素！  
:hover 表示悬浮到元素上  
:active 表示匹配被用户激活的元素（点击按住时）  
  
由于a标签的:link和:visited可以覆盖了所有a标签的状态，所以当:link，:visited，:hover，:active同时出现在a标签  
身上时 :link和:visited不能放在最后！！！  
  
隐私与:visited选择器  
只有下列的属性才能被应用到已访问链接：  
color  
background-color  
border-color  
/\*表单相关伪类\*/  
:enabled 匹配可编辑的表单  
:disable 匹配被禁用的表单  
:checked 匹配被选中的表单  
:focus 匹配获焦的表单  
  
/\*结构性伪类\*/  
index的值从1开始计数！！！！  
index可以为变量n(只能是n)  
index可以为even odd  
#wrap ele:nth-child(index) 表示匹配#wrap中第index的子元素 这个子元素必须是ele  
#wrap ele:nth-of-type(index) 表示匹配#wrap中第index的ele子元素  
除此之外:nth-child和:nth-of-type有一个很重要的区别！！  
nth-of-type以元素为中心！！！  
  
:nth-child(index)系列   
:first-child  
:last-child  
:nth-last-child(index)  
:only-child (相对于:first-child:last-child 或者 :nth-child(1):nth-last-child(1))  
:nth-of-type(index)系列  
:first-of-type  
:last-of-type  
:nth-last-type(index)  
:only-of-type (相对于:first-of-type:last-of-type 或者 :nth-of-type(1):nth-last-of-type(1))  
  
:not   
:empty(内容必须是空的，有空格都不行，有attr没关系)  
/\*伪元素\*/  
::after  
::before  
::firstLetter  
::firstLine  
::selection

5.css声明的优先级  
选择器的特殊性  
选择器的特殊性由选择器本身的组件确定，特殊性值表述为4个部分，如   0,0,0,0  
一个选择器的具体特殊性如下确定：  
      1.对于选择器中给定的ID属性值，加 0,1,0,0  
      2.对于选择器中给定的各个类属性，属性选择，或伪类，加 0,0,1,0  
      3.对于选择器中的给定的各个元素和伪元素，加0,0,0,1  
      4.通配符选择器的特殊性为0,0,0,0  
      5.结合符对选择器特殊性没有一点贡献  
      6.内联声明的特殊性都是1,0,0,0  
      7.继承没有特殊性  
  
特殊性 1,0,0,0 大于所有以0开头的特殊性(不进位)  
选择器的特殊性最终都会授予给其对应的声明  
如果多个规则与同一个元素匹配，而且有些声明互相冲突时，特殊性越大的越占优势  
  
注意：id选择器和属性选择器  
    div[id="test"]（0,0,1,1） 和 #test（0,1,0,0）     
重要声明  
有时某个声明比较重要，超过了所有其他声明，css2.1就称之为重要声明  
并允许在这些声明的结束分号之前插入 !important 来标志  
必须要准确的放置 !important 否则声明无效。   
!important 总是要放在声明的最后，即分号的前面  
  
标志为 !important的声明并没有特殊的特殊性值，不过要与非重要声明分开考虑。  
实际上所有的重要声明会被浏览器分为一组，重要声明的冲突会在其内部解决  
非重要声明也会被分为一组，非重要声明的冲突也会在其内部解决  
如果一个重要声明与非重要声明冲突，胜出的总是重要声明  
继承  
继承没有特殊性，甚至连0特殊性都没有  
0特殊性要比无特殊性来的强  
来源  
css样式的来源大致有三种  
创作人员  
读者  
用户代理     
  
权重：  
  读者的重要声明  
  创作人员的重要声明  
  创作人员的正常声明  
  读者的正常声明  
  用户代理的声明  
层叠  
1.找出所有相关的规则，这些规则都包含一个选择器  
  2.计算声明的优先级  
              先按来源排序  
              在按选择器的特殊性排序  
              最终按顺序

###css3媒体查询 css3媒体查询是响应式方案核心 ###媒体类型 all 所有媒体（默认值） screen 彩色屏幕 print 打印预览

  projection     手持设备  
  tv                   电视  
  braille           盲文触觉设备  
  embossed     盲文打印机  
  speech       “听觉”类似的媒体设备  
  tty                 不适用像素的设备

###媒体属性 width （可加max min前缀） height （可加max min前缀） device-width （可加max min前缀） device-pixel-ratio（可加max min前缀，需要加webkit前缀） orientation portrait竖屏 landscape横屏

###操作符，关键字 (only,and,(，or),not)

only： 防止老旧的浏览器 不支持带媒体属性的查询而应用到给定的样式.

@media only screen and (min-width:800px ){ 规则；

规则

}

@media screen and (min-width:800px ){

规则；

规则

}

在老款的浏览器下

@media only ---> 因为没有only这种设备 规则被忽略

@media screen ---> 因为有screen这种设备而且老浏览器会忽略带媒体属性的查询

  建议在每次抒写media query的时候带上only  
    
  and:  
  连接媒体属性 、连接媒体类型  
  对于所有的连接选项都要匹配成功才能应用规则  
    
  or(,) : 和and相似  
  对于所有的连接选项只要匹配成功一个就能应用规则  
    
  not:取反

###less

less是一种动态样式语言，属于css预处理器的范畴，它扩展了 CSS 语言，

增加了变量、Mixin、函数等特性，使 CSS 更易维护和扩展

LESS 既可以在 客户端 上运行 ，也可以借助Node.js在服务端运行。

less的中文官网：http://lesscss.cn/

bootstrap中less教程：http://www.bootcss.com/p/lesscss/

###Less编译工具

koala 官网:www.koala-app.com

###less中的注释

以//开头的注释，不会被编译到css文件中

以/\*\*/包裹的注释会被编译到css文件中

###less中的变量

使用@来申明一个变量：@pink：pink;

1.作为普通属性值只来使用：直接使用@pink

2.作为选择器和属性名：#@{selector的值}的形式

3.作为URL：@{url}

4.变量的延迟加载

###less中的嵌套规则

1.基本嵌套规则

2.&的使用

###less中的混合

混合就是将一系列属性从一个规则集引入到另一个规则集的方式

1.普通混合

2.不带输出的混合

3.带参数的混合

4.带参数并且有默认值的混合

5.带多个参数的混合

6.命名参数

7.匹配模式

8.arguments变量

###less运算

在less中可以进行加减乘除的运算

###less避免编译

###less继承

性能比混合高

灵活度比混合低

###bootstrap

简洁、直观、强悍的前端开发框架，让web开发更迅速、简单

中文网 ： http://www.bootcss.com/

英文网 : http://getbootstrap.com/

###容器

流体布局容器

容器的width为auto，只是两边加了15px的padding。

固定布局

容器的width会随设备分辨率的不同而生产变化

分辨率阈值

w >=1200 容器的width为1170 左右padding为15 （注意是borderBox）

1200>w >=992 容器的width为970 左右padding为15 （注意是borderBox）

992 > w >=768 容器的width为750 左右padding为15 （注意是borderBox）

768 > w >=992 容器的width为auto 左右padding为15 （注意是borderBox）

###栅格系统

col-lg-x

col-md-x

col-sm-x

col-xs-x

x默认拥有12个等级

###列偏移

调整的是margin-left，分13个等级（0到12）

0时为0%

###列排序

push的时候调整的是left，分13个等级（0到12）

0时为auto

pull的时候调整的是right，分13个等级（0到12）

0时为auto

###响应式工具

###容器与栅格盒模型设计的精妙之处

container 提供了一个15px的padding

row 是 column 直接存在的容器，row 默认最多可有12个 column，

同时作为都是左浮动的 column 的 wrapper，自带 clearfix 的性质。

同时 row 还有一个很特殊的地方，就是左右各有 －15px 的 margin，

为了抵消 container 中15px的 padding

每个column 也会有15px的水平方向的 padding，colunmn 只能在 row 中生存，

由于 row 的 margin 为－15px，那么位于两边的 column 就碰到了 container 的边界。

但是 colunmn 本身又有 15px 的 padding 使得它其中的内容并不会碰到 container，

同时 不同column的内容之间就有了30px的槽

目的是为了确保列与列之间有30px的槽，列与容器之间有15px的槽

​

​

​

​

​

<!DOCTYPE html>

<html lang="zh-CN">

<head>

<meta charset="utf-8">

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

<meta name="viewport" content="width=device-width, initial-scale=1,user-scalable=no">

<title>1208</title>

<link href="css/bootstrap.min.css" rel="stylesheet">

<!--[if lt IE 9]>

<script src="js/html5shiv.js"></script>

<script src="js/respond.min.js"></script>

<![endif]-->

<style type="text/css">

img{

width: 100%;

}

.damu-thumbnail img{

width: auto;

}

.damu-threeC{

margin-top:50px ;

}

.damu-hr{

margin: 50px auto;

}

.damu-hr > hr{

border:1px solid deeppink;

}

.damu-carousel{

margin-top: 50px;

}

.damu-carousel,.damu-carousel .item{

height: 500px;

overflow: hidden;

}

.carousel-caption,{

bottom: 50%;

}

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

.carousel-caption{

bottom: 5%;

}

.damu-carousel,.damu-carousel .item{

height: 200px;

}

}

</style>

</head>

<body>

<!--导航-->

<nav class="navbar navbar-inverse navbar-fixed-top">

<div class="container">

<div class="navbar-header">

<button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1" aria-expanded="false">

<span class="sr-only">Toggle navigation</span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

<a class="navbar-brand" href="#">现代浏览器博物馆</a>

</div>

<div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">

<ul class="nav navbar-nav">

<li class="active"><a href="#">综述 <span class="sr-only">(current)</span></a></li>

<li><a href="#">简述</a></li>

<li class="dropdown">

<a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" aria-haspopup="true" aria-expanded="false">特点 <span class="caret"></span></a>

<ul class="dropdown-menu" id="damu-nav">

<li><a href="#chrome">Chrome</a></li>

<li><a href="#firefox">Firefox</a></li>

<li><a href="#safari">Safari</a></li>

<li><a href="#opera">Opear</a></li>

<li><a href="#ie">IE</a></li>

</ul>

</li>

<li><a href="javascript:;" data-target="#myModal" data-toggle="modal">关于</a></li>

</ul>

</div>

</div>

</nav>

<!--轮播-->

<div id="carousel-example-generic" class="carousel slide damu-carousel" data-ride="carousel" data-interval="1000">

<ol class="carousel-indicators">

<li data-target="#carousel-example-generic" data-slide-to="0" class="active"></li>

<li data-target="#carousel-example-generic" data-slide-to="1"></li>

<li data-target="#carousel-example-generic" data-slide-to="2"></li>

<li data-target="#carousel-example-generic" data-slide-to="3"></li>

<li data-target="#carousel-example-generic" data-slide-to="4"></li>

</ol>

<!-- Wrapper for slides -->

<div class="carousel-inner" role="listbox">

<div class="item active">

<img src="images/chrome-big.jpg" >

<div class="carousel-caption">

<h2>chrome</h2>

</div>

</div>

<div class="item">

<img src="images/firefox-big.jpg" >

<div class="carousel-caption">

<h2>firefox</h2>

</div>

</div>

<div class="item">

<img src="images/safari-big.jpg" >

<div class="carousel-caption">

<h2>safari</h2>

</div>

</div>

<div class="item">

<img src="images/opera-big.jpg" >

<div class="carousel-caption">

<h2>opera</h2>

</div>

</div>

<div class="item">

<img src="images/ie-big.jpg" >

<div class="carousel-caption">

<h2>ie</h2>

</div>

</div>

</div>

<!-- Controls -->

<a class="left carousel-control" href="#carousel-example-generic" role="button" data-slide="prev">

<span class="glyphicon glyphicon-chevron-left" aria-hidden="true"></span>

<span class="sr-only">Previous</span>

</a>

<a class="right carousel-control" href="#carousel-example-generic" role="button" data-slide="next">

<span class="glyphicon glyphicon-chevron-right" aria-hidden="true"></span>

<span class="sr-only">Next</span>

</a>

</div>

<!--三列布局-->

<div class="container damu-threeC">

<div class="row">

<div class="col-md-4 damu-thumbnail text-center">

<img src="images/chrome-logo-small.jpg">

<div class="caption">

<h3>chrome</h3>

<p>Google Chrome，又称Google浏览器，是一个由Google（谷歌）公司开发的网页浏览器。</p>

<p><a href="#" class="btn btn-default" role="button">Button</a></p>

</div>

</div>

<div class="col-md-4 damu-thumbnail text-center">

<img src="images/firefox-logo-small.jpg">

<div class="caption">

<h3>firefox</h3>

<p>Mozilla Firefox，中文名通常称为“火狐”或“火狐浏览器”，是一个开源网页浏览器。。</p>

<p><a href="#" class="btn btn-default" role="button">Button</a></p>

</div>

</div>

<div class="col-md-4 damu-thumbnail text-center">

<img src="images/safari-logo-small.jpg">

<div class="caption">

<h3>safari</h3>

<p>Safari，是苹果计算机的最新操作系统Mac OS X中的浏览器。</p>

<p><a href="#" class="btn btn-default" role="button">Button</a></p>

</div>

</div>

</div>

</div>

<!--线-->

<div class="container damu-hr">

<hr />

</div>

<!--标签页-->

<div class="container" id="damu-tab">

<ul class="nav nav-tabs" role="tablist" id="damu-list">

<li role="presentation" class="active"><a href="#chrome" aria-controls="chrome" role="tab" data-toggle="tab">chrome</a></li>

<li role="presentation"><a href="#firefox" aria-controls="firefox" role="tab" data-toggle="tab">firefox</a></li>

<li role="presentation"><a href="#safari" aria-controls="safari" role="tab" data-toggle="tab">safari</a></li>

<li role="presentation"><a href="#opera" aria-controls="opera" role="tab" data-toggle="tab">opera</a></li>

<li role="presentation"><a href="#ie" aria-controls="ie" role="tab" data-toggle="tab">ie</a></li>

</ul>

<div class="tab-content">

<div role="tabpanel" class="tab-pane active" id="chrome">

<div class="container-fluid">

<div class="row">

<div class="col-md-7">

<h3>IE 你懂的</h3>

Internet Explorer，原称Microsoft Internet Explorer(6版本以前)和Windows Internet Explorer(7，8，9，10版本)， 简称IE，是美国微软公司推出的一款网页浏览器。它采用的排版引擎(俗称内核)为Trident。

</div>

<div class="col-md-5">

<img src="images/chrome-logo.jpg" />

</div>

</div>

</div>

</div>

<div role="tabpanel" class="tab-pane" id="firefox">

<div class="container-fluid">

<div class="row">

<div class="col-md-5">

<img src="images/firefox-logo.jpg" />

</div>

<div class="col-md-7">

<h3>IE 你懂的</h3>

Internet Explorer，原称Microsoft Internet Explorer(6版本以前)和Windows Internet Explorer(7，8，9，10版本)， 简称IE，是美国微软公司推出的一款网页浏览器。它采用的排版引擎(俗称内核)为Trident。

</div>

</div>

</div>

</div>

<div role="tabpanel" class="tab-pane" id="safari">

<div class="container-fluid">

<div class="row">

<div class="col-md-7">

<h3>IE 你懂的</h3>

Internet Explorer，原称Microsoft Internet Explorer(6版本以前)和Windows Internet Explorer(7，8，9，10版本)， 简称IE，是美国微软公司推出的一款网页浏览器。它采用的排版引擎(俗称内核)为Trident。

</div>

<div class="col-md-5">

<img src="images/safari-logo.jpg" />

</div>

</div>

</div>

</div>

<div role="tabpanel" class="tab-pane" id="opera">

<div class="container-fluid">

<div class="row">

<div class="col-md-5">

<img src="images/opera-logo.jpg" />

</div>

<div class="col-md-7">

<h3>IE 你懂的</h3>

Internet Explorer，原称Microsoft Internet Explorer(6版本以前)和Windows Internet Explorer(7，8，9，10版本)， 简称IE，是美国微软公司推出的一款网页浏览器。它采用的排版引擎(俗称内核)为Trident。

</div>

</div>

</div>

</div>

<div role="tabpanel" class="tab-pane" id="ie">

<div class="container-fluid">

<div class="row">

<div class="col-md-7">

<h3>IE 你懂的</h3>

Internet Explorer，原称Microsoft Internet Explorer(6版本以前)和Windows Internet Explorer(7，8，9，10版本)， 简称IE，是美国微软公司推出的一款网页浏览器。它采用的排版引擎(俗称内核)为Trident。

</div>

<div class="col-md-5">

<img src="images/ie-logo.jpg" />

</div>

</div>

</div>

</div>

</div>

</div>

<!--版权-->

<div class="container">

&copy;尚硅谷

</div>

<!--模态框-->

<div class="modal fade" tabindex="-1" role="dialog" id="myModal">

<div class="modal-dialog" role="document">

<div class="modal-content">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal" aria-label="Close"><span aria-hidden="true">&times;</span></button>

<h4 class="modal-title">模态框</h4>

</div>

<div class="modal-body">

<p>我是一个模态框</p>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-default" data-dismiss="modal">Close</button>

</div>

</div>

</div>

</div>

<script src="js/jquery.min.js"></script>

<script src="js/bootstrap.min.js"></script>

</body>

<script type="text/javascript">

$(function(){

//1.链式调用 2.读写二合一 3.隐式迭代 4.编码函数化

$('.carousel').carousel({

interval: 2000,

pause:null,

wrap:false

})

$('#test').click(function(){

$('.carousel').carousel('cycle');

})

$('#test2').click(function(){

$('.carousel').carousel('pause')

})

$("#damu-nav > li > a").click(function(e){

var href = $(this).attr("href");

console.log("#damu-list > li > a[href="+(href)+"]");

$("#damu-list > li > a[href="+(href)+"]").tab('show');

$(document).scrollTop($("#damu-tab").offset().top);

e.preventDefault();

})

})

</script>

</html>

​