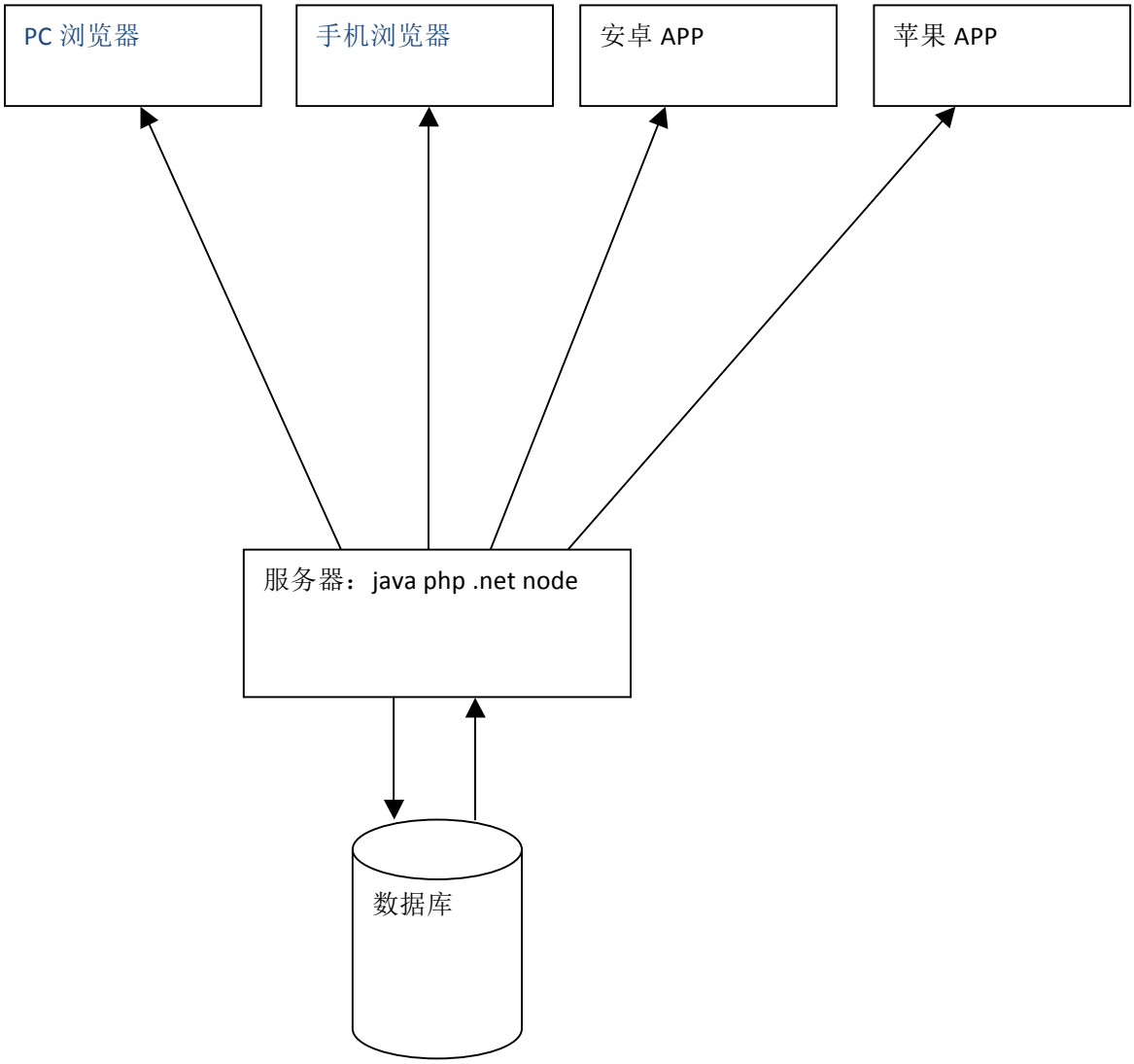
Html

软件架构



HTML（Hyper Text Markup Language）超文本标记语言：制作网页的语言，基于浏览器运行的，解释性的语言，不需要编译，语法不区分大小写，由 W3C 组织创造，目前的版本分为 html4.01 和 html5 两大版本

基本语法：<标签名称> </标签名称> 成对标签　<标签名称> 没有结束的单一标签

创建一个网页，扩展名.html，一般网站主页规范名称 index.html default.html

Html 的基本结构：

```
<html>
    <head>
        网页的文档的定义
    </head>
    <body>
        网页的文档的内容
    </body>
</html>
```

嵌套标签，内层标签要缩进（tab 键）

编码：每个国家都有自己的语言编码，网页需要显示的设置编码，推荐设置 UTF-8 这种 unicode 编码(全球统一编码)

DreamWeaver 是一个所见即所得的网页开发工具，简单的说，就是可以自动生成标签

文档类型（doctype）：作用是让浏览器知道当前的网页使用的是那种 html 文档语法，html4 系列，html5 系列，xhtml 系列，更好的解析 html 内容

html 属性：是对标签具体的设置，每个标签都有 0 到多个属性，设置：属性名称=”属性值”

meta 标签设置元数据，根据属性不同，功能也不同
<meta charset="utf-8"> 设置文档编码

Web 颜色表示法有三种：
16 进制数字表示法：#000000 纯黑 #ffffff 纯白 六位可以简写为三位数字，例如#0cc，换算为#00cccc
英文单词表示法，例如 red green blue yellow gray
RGB 表示法，CSS 支持

注释：让浏览器忽略执行 <!-- 注释文字内容 -->

实体字符：对特殊字符进行转义
< <
> >
空格
双引号 "
&符号 &

文字相关的标签：
<font> ：属性 size 字号 color 颜色 face 字体
<b>加粗 <i>倾斜 <u>下划线 加粗 倾斜 <sub>下标 <sup>上标

段落相关的标签：

h1~h6 标题文字

<p> 段落文字

 换行

超级链接：<a>标签 方便用户点击定向到一个 url，链接的载体可以是文字和图片

href 属性：链接的 url

target 属性：打开链接的方式 属性值 _self 当前窗口（默认） _blank 新窗口 _parent 父框架 _top 最外层框架

title 属性：鼠标椅上提示文字

锚点链接：链接到网页一个具体位置 <a href=”#锚点名称”>

定义锚点<a name=”锚点名称”>

定义锚点方式 2：在元素中用 id 属性定义锚点名称

图片：img 标签

src 属性：图片文件的路径

alt 属性：替代文字，在图片无法显示的时候会显示，另一个作用就是作为 SEO 搜索引擎优化

width 属性：图片宽度 像素或者百分比单位

height 属性：图片高度，同上

引用外部资源的路径问题

路径表示法：

物理路径：直接用驱动器下的目录表示，例如 c:\abc\123\a.jpg

绝对路径：引用 url 例如 http://www.sohu.com/img/Harvest.jpg

相对路径：

相对于当前位置：../代表上一级目录

相对于根目录：以/开头表示根目录，这种路径表示法必须基于服务器运行

水平线：<hr>标签

width 属性：宽度

align 属性：对齐方式，left center right

size 属性：高度

noshade 属性：单一属性，没有属性值，代表是否有阴影

color 属性：颜色

预格式化：<pre>标签 把源代码中的格式体现到浏览器中

自定义列表：<dl>标签代表列表 <dt>标签代表标题 <dd>标签代表内容

有序列表：<ol>代表列表 代表一个条目

无序列表：<ul>代表列表 同上

div 是排版布局中最常用的标签元素，本身是一个块级元素

span 是一个内联元素，和 a 是相同性质

块级元素（block）：每个元素必须占据一行，可以设置宽度高度，例如 div h1~h6 p hr

内联元素（inline）：多个元素可以在一行，不可使设置宽度高度，例如 span a

内联块级（inline-block）：多个元素可以在一行，可以设置宽度高度，例如 img 表单元素

网页排版布局的方式，主流方式有两种：

1. 表格排版

2. CSS+DIV

表格：基本的三个标签组成 <table>表格 <tr>行 <td>单元格 <th>标题单元格

<thead>表格头部区域 <tbody> 表格内容区域 <tfoot>表格底部区域

Table 标签的属性：border 边框的宽度 width 宽度 height 高度 bgcolor 背景色 cellpadding 填充（单元格内边距） cellspacing 间距（单元格外边距）

Tr 和 td 标签的属性：align 水平对齐 valign 垂直对齐：top middle bottom width 宽度 height 高度 bgcolor 背景色

合并单元格：colspan 单元格跨列数 rowspan 单元格跨行数 大量的合并单元格会使代码变得混乱，建议用表格嵌套布局来代替合并单元格

id 属性和 name 属性的区别：id 属性每个元素都有，name 属性只有少数的元素有，例如 form img frame，id 必须是网页中唯一的，不能重复，name 属性值在网页中允许重复。

表单：<form>标签 表单可以包含表单元素

id：通用属性，id 名称

name：表单名称

action：提交的 url 地址

method：提交数据的方式，get（默认）和 post

单行文本框：<input type=”text”>

size：字符宽度

maxlength：允许最多字符数

readonly：只读（单一属性）

disabled：禁用，和只读不同，禁用会使表单元素的数据无法提交

value：默认值

placeholder：输入提示（h5 属性）

密码框：<input type=”password”> 属性同文本框

文本区域：<textarea> 没有 value 属性，默认值是写在标签中的

cols：列宽

rows：行高

单选按钮：<input type=”radio”>

name：单选按钮同一组的 name 名称必须相同

checked：默认选中（单一属性）

value：设置单选的 value 值

label 标签可以辅助单选复选的文字绑定，唯一的属性就是 for 对应单选复选的 id

复选按钮：<input type=”checkbox”> 属性同单选按钮 一组复选按钮的 name 也必须相同

列表框：<select>和<option>

select 标签属性：size 显示的条目数量 multiple 是否允许多选

option 标签属性：value 设置值 selected 默认选中

文件域：<input type=”file”>：上传文件的表单必须要设置 enctype 属性值为 multipart/form-data（二进制数据），默认是 application/x-www-form-urlencoded（纯文本数据） 多文件选中需要设置 multiple 属性

隐藏域：<input type=”hidden”> 用户看不见的表单元素，一般设置一个需要提交但是不需要用户看到的数据

三种按钮：

提交按钮：<input type=”submit”>

重置按钮：<input type=”reset”>

普通按钮：<input type=”button”>

<button>标签也可以产生按钮

Fieldset 字段集：产生一个包围表单元素的 配合 legend 标题文字

框架概念：一个浏览器窗口，可以显示不同的网页，框架是对窗口进行分割，框架的网页数量是 n+1

框架集：<frameset>标签 包含一到多个<frame>

cols 属性：水平分割的设置

rows 属性：垂直分割的设置

frameborder 属性： 是否显示边框 yes no , 1 0

framespacing 属性：边距大小，IE 特有

框架：<frame>标签 一个 frame 就是一个单独的分割窗口

src 属性：引用的网页路径

noresize 属性：不允许用户调整大小

scrolling 属性：滚动条设置 auto 默认 no 无滚动条 yes 永远出现滚动条

框架页本身没有 body 标签，没有自己的内容，是引用其他网页的内容

浮动框架：<iframe>标签，可以灵活的潜入到网页的任何位置，不需要分割窗口

多媒体：<embed>标签 可以在网页中潜入视频和音频 html4 中，浏览器本身是没有播放音频视频的能力，是需要借助系统的插件播放器来播放

滚动效果：<marquee>标签

CSS

Css（Cascade Style Sheet）级联式样式表，html 本身的语法特点决定只能决定页面显示的内容，但是 html 的样式功能极其薄弱，所以网页的样式格式基本都是由 css 语法控制的。
使用 css 控制网页的样式，优点：样式选择多，可以统一控制网页的格式。

目前有两个版本：css2.1 css3

CSS 的使用方式

1. 在<head>中加入<style>标签（嵌入式）
2. 外联 css 文件（外联式）用<link>标签在 html 中引用
3. 直接在标签中用 style 属性设置样式（行内样式）

定义样式：选择器 {样式 1：值;样式 2：值;……}

选择器的分类：

1. **元素选择器，标签名称作为选择器 例如 p h1 table**
2. **类选择器，用.语法定义类名，然后在元素中用 class 属性应用**
3. **Id 选择器，用#语法定义，指定 id 的元素会自动应用样式**
4. 通配选择器，用*定义，选择所有元素
5. 后代选择器，语法：E F，E 和 F 是元素，类，id 选择器任意情况 例如 ul li，#div1 a，.p1 .span1
6. 父子选择器，语法：E>F，例如 ul>li ， #div1>a
7. 群组选择器，语法：E,F 例如 p,h1,h2
8. 属性选择器：语法：E[属性] E[属性=值] 例如 input[size] input[type=text]
9. 伪类选择器：
   语法：
   a) E:link 未访问过的链接状态
   b) E:visited 访问过时的样式
   c) E:hover 鼠标悬停时的样式
   d) E:active 鼠标按下时的样式（激活）
   e) E:focus 获得焦点是的样式
10. 伪元素选择器：语法 E:after E:before 通过 css 添加页面内容，必须配合 content 属性

选择器优先级：id 选择器>类选择器>元素选择器 行内样式>内嵌样式>外联样式

选择器的权重：
元素选择器 1
类选择器 10
Id 选择器 100
行内样式 1000
!important 权重最高！

伪类选择器，例如:link :hover 等等权重是 1
属性选择器，例如[href] 权重是 10
在权重完全相同情况下，下面声明的优先级比上面声明的高。

字体相关样式：
font-family：字体，可以设置多个，以第一个好用的为准，安全字体：系统一定安装的字体
font-size：字号，单位可以 px 和百分比，其他还可以是 em rem，webkit 内核浏览器不支持小于 12px 的字号，需要做特殊设置
color ：文字颜色
font-weight：加粗 bold 加粗 normal 取消加粗
font-style：倾斜 italic 倾斜 normal 正常
text-decoration：文字修饰 none 无 underline 下划线 overline 上划线 line-through 删除线
line-height：行高，单位可以是数字，百分比，和固定像素，（百分比和数字都是以当前字号为基数做换算）
text-indent：文字缩进，可以为负数
text-transform：大小写转换
font：字号行号字体的合并写法
white-space：强制不换行 nowrap
word-wrap:强制换行 break-word
text-align：水平对齐 left center right

盒子模型：一个完整的盒子模型是由四部分组成，内容 padding（内边距） border（边框） margin（外边距）

padding：内边距，可以上下左右单独设置，也可以写出合并写法 一个值：padding：上下左右，两个值：padding:上下 左右 三个值：padding：上 左右 下 四个值：padding：上 右 下 左

margin：外边距，也可以上下左右单独设置，也可以写合并写法，同 padding，可以写负数，也可以写出 auto 表示浏览器自动计算

垂直 margin 失效问题的解决办法：
1. 父元素有内边距或者有内容
2. 父元素有 border
3. 父元素 position 设置 absolute 或者 fixed，或者 float 浮动，脱离文档输出流

垂直 margin 合并的问题：垂直的 margin，会合并为一个 margin，取两者之间的最大值

border：边框 上下左右可以单独设置 三个样式：宽度，颜色，线型可以单独设置
outline：外边框，是在 border 之外的边框

区块相关样式：

width：宽度 像素或者百分比
max-width:最大宽度
min-width:最小宽度
height:行高 同上
max-height：最大高度
min-height：最小高度

overflow：溢出容器 visible 默认，正常显示 hidden 超出部分隐藏 scroll 固定有滚动条 auto 溢出后出现滚动条

盒子模型的怪异模式：
在标准模式的盒子模型中，padding 和 border 都是向外扩展空间，并不会占据内容本身的空间
在怪异模式的盒子模型中，padding 和 border 是占据内容本身的空间，并不会向外扩展
产生怪异模式的情况：
1. 不写 doctype 文档类型并且在低版本的 IE 的浏览器下
2. 在 CSS3 中用 box-sizing 属性设置值为 border-box

定位相关的样式：
display：决定元素显示的特征 none 不显示 inline 内联元素特征 inline-block 内联块特征 block 块元素特征

显示隐藏元素
display 属性:none 隐藏 其他值是显示，不占位隐藏
visibility 属性：visible 显示 hidden 隐藏，占位隐藏

排版定位的原则：从上到下，从左到右，从外到内

float：浮动元素，none 不浮动 left 左浮动 right 右浮动
浮动元素会脱离正常文档输出流，float 设计的初衷是文字环绕图片，内联元素应用了 float，默认就是块级元素特征

clear：清除浮动 left 不允许上边有左浮动元素 right 不允许上边有右浮动元素 both 不允许上边有左或右浮动元素

浮动元素会导致父元素**高度塌陷**，解决的方法：
1. 父容器设置固定高度
2. 最后一个浮动子元素的后面，加上一个任意块级元素应用 clear 清除浮动样式
3. 父元素应用 overflow:hidden 样式，在 IE6 下需要用 zoom:1 样式做兼容
4. 用伪元素动态的在父元素的最后输出一个空内容，在空内容上应用清除浮动样式

position：设置元素的定位方式，取值：
static：正常的文档输出流，无法使用 left top right bottom 属性（默认）
absolute：绝对定位，可以使用 left top right bottom 进行定位，脱离文档输出流，absolute

元素在设置 left top right bottom 之前位置不会相对于原来的位置有任何变化（位置跟随）
如果有父元素，且父元素设置了 position:fixed 或 absolute 或 relative，相对于父元素的原点定位，如果不是，则相对于网页原点定位
fixed：永远相对于视口定位，脱离文档输出流
relative：相对定位，正常文档输出流，利用 left top right bottom 相对于原理的位置进行偏移

盒子的叠放次序问题：
1. 相同的定位下，默认就是下面叠放次序比上面高
2. 设置 position 叠放次序（非 static）比默认高
3. 设置 z-index 值越大，叠放次序越高，z-index 设置为负数，叠放次序要低于正常文档输出流
4. 设置 position 叠放次序比 float 浮动高

背景的相关样式：
background-color：背景颜色
background-image：背景图片
background-repeat：平铺的方式 repeat 默认平铺 no-repeat 不平铺 repeat-x 横向平铺 repeat-y 纵向平铺
background-attachment：背景是否固定 fixed 为固定
background-position：设置背景图片的位置，
　　百分比：偏移量=（容器的宽度-图片的宽度）*百分比　高度也是同样道理 可以为负数
　　像素：直接设置偏移量，也可以为负数
　　固定位置：center left right top bottom
background：背景多个样式的合并写法

雪碧图（CssSprite）：把多个图片图标合成一个图片，利用 css 的背景图片位置偏移单独显示某一个图标，主要好处就是减少网络连接的请求次数，提高效率，网站优化。

代码重构：重新组织 html 和 css，减少冗余代码，让代码结构更加健壮。

border-collapse：相邻边框是否合并 collapse 合并

样式的继承：子元素直接应用父元素的样式设置，有一些样式默认就可以被子元素继承，例如 font-size color line-height
显式继承：inherit

引用外联样式表的第二种的方式
<style>
　　@import “css 文件路径”;
</style>
如果同时用<link>和@import，<link>的加载比@import

clip：裁剪，必须是 postion:absolute 元素才可以应用

网站结构：html 画面 images（图片文件） css（样式文件） js（程序脚本）

reset.css 文件：统一重写浏览器的默认元素样式

CSS3.0

是 css2.1 的升级版本，多出了很多 css 样式，现在已经得到了绝大多数的浏览器的支持，IE8 只支持其中一小部分，在移动端支持良好，因为移动端是安卓，苹果手机占大众化，所以谷歌内核（webkit）的浏览器是最多的。

浏览器的私有前缀：

谷歌浏览器 –webkit
火狐浏览器 –moz
欧朋浏览器 –o
IE 浏览器 -ms

例如 -webkit-box-shadow

overflow-x：水平溢出设置
overflow-y：垂直溢出设置

border-radius：圆角 单位像素和百分比
单独设置 border-top-left-radius border-top-right-radius border-bottom-left-radius border-bottom-right-radius

box-shadow：阴影

box-reflect：倒影 兼容性比较差，目前只有 webkit 内核支持

background-origin：背景图片占据的盒子模型 padding-box（默认）在 padding 开始显示背景 border-box 在 border 区域开始显示背景 content-box 在内容区域开始显示背景

background-clip：背景裁剪 属性值同上

background-size：设置图片的大小 可以是像素，也可以是百分比 cover 宽度或高度自适应容器，超出的部分隐藏 contain 保证背景能够完整显示，但是容器可能出现空白区

rgba 表示法：最后的 a 就表示颜色透明度取值 0~1

opacity：设置网页元素的透明度 取值 0.0~1.0
IE 兼容写法：filter:alpha(opactiy=值) 值 0~100 之间

calc：css3 新的计算宽度和高度的一个函数，可以做百分比和像素的混合运算，例如 calc(100% - 100px) 注意！运算符两边有空格！

tab-size：在 pre 标签中一个 tab 的空格数量
text-shadow：文字阴影
text-overflow：文字溢出显示省略号 ellipsis

column-count：文字分栏

transition：css 样式过渡，但是不是所有的 css 样式都能做成过渡效果
transition-property 参与过渡的样式名称，如果全都参与，设置为 all
transition-duration 过渡时间，秒为单位，例如 3.5s
transition-timing-function 过渡的动画类型
transition-delay 动画延迟执行的时间，秒为单位

transform：元素的变换效果
平移：translate translateX translateY 单位可以是像素或者百分比（自身的百分比，不是父元素）
旋转:rotate(60deg) 默认按 z 轴旋转（2D 旋转）transform-origin 配置旋转的原点
缩放：scale scaleX scaleY 值是原始大小的倍率
扭曲：skew skewX skewY

transform-style：preserve-3d 产生 3d 立体效果
perspective：远近度，值越小立体效果越强，必须要加在父元素上，单位像素
backface-visibility:hidden 元素旋转到背面不可视

新增选择器：
E:not 不是某个选择器
E:first-child 父元素的第一个子元素
E:last-child 父元素的最后一个子元素
E:only-child 父元素仅有的一个子元素
E:nth-child(n) 匹配父元素的第 n 个子元素

E:first-of-type 匹配同类型中的第一个同级兄弟元素
E:last-of-type 匹配同类型中的最后一个同级兄弟元素
E:only-of-type 匹配同类型中的唯一的一个同级兄弟元素
E:nth-of-type(n) 匹配同类型中的第 n 个同级兄弟元素

E:empty 匹配没有任何子元素（包括 text 节点）的元素

E:checked 选中状态
E:enabled 可用状态
E:disabled 禁用状态
E:selection 选中状态

E:first-letter 段落中的首字
E:first-line 段落中的第一行

E[att] 选择具有 att 属性的 E 元素
E[att="val"]选择具有 att 属性且属性值等于 val 的 E 元素
E[att~="val"]选择具有 att 属性且属性值为一用空格分隔的字词列表
E[att^="val"]选择具有 att 属性且属性值为以 val 开头的字符串的 E 元素
E[att*="val"]选择具有 att 属性且属性值为包含 val 的字符串的 E 元素
E[att$="val"]选择具有 att 属性且属性值为以 val 结尾的字符串的 E 元素

E~F 选择 E 元素后面的所有兄弟元素 F

filter:blur 模糊效果

网络字体：字体文件放到网站中，用户打开网页的时候自动下载引用这个字体
@font-face 自定义字体，将字体文件定义为 css 中的字体
主要用于字体图标

border-image：用图片设置边框线
border-image-source 设置图片路径
border-image-slice 图片分割
border-image-width 边框宽度
border-image-outset 外扩宽度
border-image-repeat 重复的设置

@keyframes 自定义动画
animation 调用自定义动画
animation-name 调用的动画名称
animation-duration 动画执行的时间
animation-timing-function 动画执行的类型，默认 ease
animation-delay 动画延迟时间
animation-iteration-count 动画执行次数 infinite 无限次
animation-direction 是否有反向动画 alternate 为有反向动画，默认没有
animation-fill-mode 动画执行结束后的状态
animation-play-state 动画是否暂停 running 运动 pause 暂停

less 和 sass

css 中有可能会大量的编写重复性的样式设置，例如重复的属性值，重复的样式，当我们后期维护的时候，就比较困难，一次改变外观，需要修改多处 css。

Css 中没有变量的概念（常量，就是不可变的量，变量，就是可变的量，变量的取名规范驼峰写法，xxxXxxXxx，多个单词组成，第一个单词全小写，第二个开始首字母大写，例如 studentCode）

Less 和 sass 就是为了解决以上问题的一种工具语言，这个语言模拟了变量的概念，必须通过工具去自动维护 css 文件

Less 和 sass 都是 css 预处理器，less 的文件扩展名是.less，sass 的文件扩展名是.scss。
单行注释：// 不会出现到 css 文件中 没有多行注释专用语法

Less 定义变量使用@符号 Sass 定义变量使用$符号，变量可以直接做数学表达式运算，定义到样式外的是全局变量，定义到样式里的是局部变量，局部变量优先级高于全局变量。

JavaScript

JavaScript 由网景公司开发，后来由于浏览器大战，出现了多个版本的 JavaScript，例如 IE 的 Jscript，所以不同的浏览器会有兼容性问题。

JavaScript 通过浏览器执行，是一种即时编译的脚本语言，是解释性语言（不需要事先编译好，一边编译一边执行的语言）

面向过程的语言：程序的最小单位是函数，没有对象，类，属性，方法的概念，例如 C 语言
面向对象的语言：程序的最小单位是类，没有函数的概念，都是对象，属性，方法，例如 Java，c++，c#

JavaScript 是面向过程，基于对象的。

Web 前端的三层结构：html（结构）css（表现）JavaScript（行为）

目前的标准的名称是 EcmaScript，最新版本是 EcmaScript6.0。

JavaScript 的语法严格区分大小写，每条语句用分号结束。

严格模式：在代码中加入 'use strict'; 如果代码中出现不规范的语句，直接报错

使用方式：

内嵌式：在网页的 head 或者 body 中，加入<script>标签
外联式：在<script src=”外部 js 文件”>

注意：

外联的 css 文件，如果用到了外部资源，相对路径的位置以 css 文件位置为准
外联的 js 文件，如果用到了外部资源，相对路径的位置以应用了 js 的 html 文件位置为

准

document.write(数据) 输出数据到页面
window.alert(数据) 弹出对话框 window 对象可以省略，直接调用 alert 方法

单行注释：//
多行注释：/* */

控制台输出语句: console.log(数据) IE 不支持，会直接报错

变量用 var 声明，必须是字母，下弧线，$符号开头，后面可以是字母，下划线，$符号，数字，不可以是关键字（程序中有特殊含义的，例如 var）
例如 stuId $font12 _color
变量声明语法：var 变量; var 变量=值; var 变量 1，变量 2=值，……;

Js 的数据类型是弱类型，不是像 java 那种强类型，变量必须显示的声明数据类型
强类型举例： int i = 10; String s = “abcd“; double d = 3.14;
弱类型举例：var i = 10; var s = “abcd”; var d = 3.14;

JS 的数据类型划分：

基本数据类型（原始数据类型）

undefined 未定义

null 空对象（代表空指针）

number 数字型 例如 100 3.14

string 字符型（字符串）常量需要用双引号或者单引号括起 例如 “中国” ‘abcd’ 空字符串”” ‘’

boolean 布尔值 就两个取值 true 代表真 false 代表假

引用数据类型

object 对象类型

function 函数类型

返回数据类型的函数 typeof

NaN 不是一个单独的数据类型，是一个 number 的特殊值，代表计算结果是非数字（Not a Number）

Asc 码：用 0~127 的数字代表键盘的 128 个字符 例如 a~z 97~122，A~Z 65~90，0~9 48~57
Unicode 码用 0~65535 代表全球统一编码（世界上所有官方语言） unicode 的前 128 位就是 asc 码

运算符：

字符串拼接符：+ 运算符两边有一个是字符串，就做拼接运算

数学运算符：+ - * / %取模 -取负 ++自增 --自减

++ -- 前置，先做自增自减运算，再过其他运算，后置，先做其他运算，再做自增自

减运算

赋值运算符： = += -= *= /= %=

比较运算符：>大于 <小于 >= 大于等于 <= 小于等于 ==等于 !=不等于 ===严格等于 !==严格不等于 比较运算返回的值都是布尔型 true 和 false

严格等于：先判断两个数据类型是否一致，如果不一致，直接 false，如果一致，再进行比较

在 js 的判断条件中以下值代表假：false 0 空字符串 null undefined，其他的都代表真

逻辑运算符：优先级 ! > && > ||

&&逻辑与：同真则真，有一个假，就为假

||逻辑或：有一个真就为真，同假则假

! 逻辑非：真变假 假变真

逻辑与的短路运算：表达式 1 为假，不执行表达式 2

逻辑或的短路运算：表达式 1 为真，不执行表达式 2

单目运算符： ++ -- - !

双目运算符：> < >= <= == != && ||

三目运算符：表达式?返回值 1:返回值 2 表达式为真返回值 1，为假返回值 2

浮点运算：根据 IEEE 标准特点，浮点数运算一定会有误差

window. prompt 输入对话框

parseInt(数据)：可以把其他数据转换为整数数字型

isNaN(数据)： 判读是否是非数字

转义符号：反斜杠 \ 例如 \" \' \\

特殊转义：\n 换行符 \r 回车符 \t 制表位

JavaScript 是基于对象，面向过程的语言，语言中既有对象，方法，也有函数的概念。

在 JS 里，方法函数都是 function

function 可以理解为是一个定义好的功能，一次声明，多次调用

声明 function 基本语法

function 自定义名称（形参 1，形参 2，……） {

语句块代码

[return [返回值]];

}

调用 function

function 名称(实参 1，实参 2，……);

return 表示退出当前函数，回到调用者那里继续执行（即时没有 return 语句，函数执行完毕

也会回到调用者那里继续向下执行）

return 返回值，表示退出当前函数，并且把函数的返回值，返回给调用者，然后继续向下值

递归调用：函数调用自己本身，如果没有设置正确的退出条件，很容易形成无限递归（死递归）

document.getElementById(元素的 id)：通过 id 得到元素的对象

流程控制-条件语句：只有条件为真才会执行特定语句，条件为假再执行其他语句

If 语句语法 1：

```
if (条件表达式) {
    条件为真执行的语句块
}
```

If 语句语法 2：

```
if (条件表达式) {
    条件为真执行的语句块
} else {
    条件为假执行的语句块
}
```

If 语句语法 3：

```
if (条件表达式 1) {
    条件 1 为真执行的语句块
} else if (条件表达式 2) {
    条件 2 为真执行的语句块
} else if (条件表达式 3) {
    条件 3 为真执行的语句块
}……
 else {
    以上条件都为假执行的语句块
}
```

如果有一个条件为真，就会执行对应的语句块，然后退出整个 if 语句块，没有机会执行其他的条件判断

if 里的执行语句块如果只有一句，可以省略花括号

switch 条件判断：和 if 的区别就是，只能做等值判断

```
switch(表达式) {
    c ase  值 1:
        执行语句;
```

```
        [break;]
    c ase  值 2:
        执行语句;
        [break;]
    ……
    default:
        执行默认语句;
        [break;]
}
```

break 如果不写，当条件为真的时候，执行语句后，会无条件的继续向下执行，可以用 break 语句直接跳出整个 switch 语句

循环语句：在满足条件的情况下，反复执行特定的代码
一个完整的循环的组成：初始计数值，条件，迭代，循环代码

while 循环：只要条件为真就一直循环执行

```
whiile (条件表达式) {
    循环执行的代码
}
```

do…while 循环：类似 while 循环，但是是先执行一次循环体代码，然后再判断循环条件

```
do {
    循环执行的代码
} while (条件表达式);
```

for 循环：典型的计数循环，通过一个计数变量来控制循环的次数

```
for (计数变量初始化;条件表达式;迭代) {
    循环执行的代码
}
```

break：跳出整个循环继续向下运行
continue：跳出当前循环（continue 下面的语句不执行），进行下一次循环

for…in 循环：遍历对象的成员或者遍历数组的下标

```
for (变量  in  对象) {
    循环执行代码
}
```

代码调试：给需要单步运行的代码行加上断点，然后用单步运行进行调试

变量的作用域：变量的使用范围，
    分为两种作用域：
        全局变量：属于 window 对象（全局对象，是作用域中最大的对象），在整个 js 程

序中都可以使用，声明在 function 的外部，尽可能少用全局变量，会造成全局污染。

局部变量：属于某个 function 范围，仅限于在 function 内部使用

JS 中没有块级作用域这一说，最小的作用域就是一个 function，声明在 function 里面的变量包括参数都是属于局部变量。

变量如果没有加 var 声明，即时写在 function 内部，也算全局变量。

作用域链：当在某个 function 使用某个的变量时候，首先先查看当前 function 作用域中是否有这个变量，如果有，直接使用，如果没有，就会像上层（外层）作用域查找，直至全局作用域。

生命周期：变量出现在内存开始，到从内存中销毁为止，中间的过程，就是一个变量的生命周期，一个局部变量的生命周期从函数调用开始，到函数执行完毕，全局变量的生命周期是整个 js 程序结束。

面向对象编程

类：**抽象**上的定义，不具体代表任何一个实例
对象：必须是一个**具体**的对象实例

类和对象是一对多关系，一个类可以创建多个对象实例，通过类创建对象的过程叫做实例化

声明类：

```
function 类名() {
    属性定义
    方法定义
}
```

类名的规范是大驼峰：XxxXxXxxx 每一个单词首字母大写

通过类创建对象实例

var 变量 = new 类名(); 这是 function 的**构造函数**调用，调用构造函数的作用就是创建某个类的对象实例

对象的成员：

属性：作用是存储对象的数据，在类中用 this 关键字来定义属性

方法：作用是实现对象的功能，在类中用 this 关键字来定义方法

function 本身就是一个对象，它的数据类型是 function
创建一个 function 有三种写法：
写法 1：函数声明

```
function 自定义名称() {
    语句块
}
```

写法 2：函数表达式

```
var 名称 =  function() {
    语句块
}
```

写法 3：创建 Function 类的对象

```
var 名称 = new Function('参数 1','参数 2',……,'函数语句块');
```

函数声明和函数表达式最大的区别是，函数声明带有声明提升，即使函数声明在后面，js 引擎把函数在执行之前放到代码的最前面，但是函数表达式没有声明提升。

匿名函数：没有名字的函数，例如 function(xx,xx){}，本身就是一个函数对象

匿名自执行:

```
(
    function(xxx,xx) {

    }
)(xxx,xxx);
```

this 关键字：代表一个对象，不同的情况下，this 代表的对象也不同

函数级调用 function：this 永远代表 window 全局对象
构造函数调用 function：this 永远代表当前创建的对象
对象实例方法调用 function：this 永远代表当前的对象
静态方法调用 function：this 永远代表当前类的 function 对象

内存分布：

栈区：只存储局部变量（也包括全局变量）
堆区：只存储对象的实例，每个对象实例在堆中都会自动产生一个唯一的内存地址

对象之间赋值赋的是引用地址，基本类型赋值赋的是值本身。

私有方法：在类中声明的函数，就是只能在类内部调用，在外部调用不了的功能
公有方法：在类中声明的正常方法，在类的内部可以调用，也可以在类的外部使用

JS 的对象时可以随时动态添加和删除对象的成员（属性和方法）
在对象中添加属性或者方法
对象.属性=xxx 对象.方法名=function() {}
删除对象中的某个属性后者方法
delete 对象.属性 delete 对象.方法

Object 类：是 JS 语言自带的一个类，是所有类的父类（超类），类中的属性和方法会被其他的类所继承，所有说，任何类都具有 Object 类的成员

字面量对象：等同于创建一个 Object 类的对象，利用特定的语法，设置属性和方法

{
    属性名称：属性值,
    属性名称：属性值，
    ……
    方法名称：function(){},
    方法名称：function(){},
    ……
}
对象实例成员（对象实例的属性和方法）：是属于具体的某个对象实例，需要通过对象调用
静态成员（类成员）：是属于类本身的，不属于任何一个具体的对象实例，需要通过类调用
在类中定义静态属性：类名.属性 = 属性值　　类名.方法名 = function(){}
静态成员在 js 中只能通过类访问，不能通过对象访问，静态方法中不能访问对象实例成员，但是可以调用静态成员。

访问对象的成员的两种方式：
    对象.成员
    对象['成员名称']：一般用在属性或者方法是动态调用，而不是固定属性和方法名称的情况下

instanceof 运算符： 对象 instanceof 类 ，如果对象是这个类创建的，返回 true，否则为 false

with 语句：可以绑定一个默认对象，可以在调用对象成员的时候，省略对象前缀
with (默认对象) {
    直接访问对象的成员
}
with 效率极差，开发中不推荐使用。

Array 数组：数组是对象引用类型，在 js 中的类名叫做 Array，可以存储相同类型的一组数据，一个数组对象可以存放数组元素，数组元素是用下标访问，起始下标是从 0 开始，js 的数组是可变长数组，可以任意的添加新元素

创建数组对象的方式：
1. var 变量 = new Array(); 基本语法
2. var 变量 = new Array(值 1，值 2，……); 直接初始化元素值
3. var 变量 = [值 1，值 2，……]; 等价于上面语句，是字面量写法

数组对象的属性方法：
length 属性：返回数组的元素个数
push 方法：在数组末尾添加元素
pop 方法：删除数组末尾元素
unshift 方法：在数组头部添加元素
shift 方法：删除数组头部的元素
splice 方法：删除数组指定的元素
concat 方法：合并另一个数组

join 方法：按照指定的分隔符（默认逗号）将数组元素拼接为一个字符串
reverse 方法：颠倒数组中元素的顺序
slice 方法：根据起始和终止下标截取返回数组的一部分，下标可以为负数，代表倒数第 n 位
sort 方法：两种用法：
　　无参：默认按照数组中的元素的 asc 码来排序
　　传参：传入一个 function，在这个 function 中定义排序规则，这个排序规则的 function 需要传入两个参数，代表比较的两个数值，如果是按照升序排序，参数 1>参数 2，返回正整数，参数 1<参数 2，返回负整数，参数 1==参数 2，返回 0

数组排序算法：**冒泡排序**（下沉法，上浮法） 选择排序　插入排序　希尔排序　快速排序
冒泡排序下沉法算法核心思想：每次冒泡都会把一个最大的值沉到最底下

数组查找算法：线性查找法　二分查找法（折半查找法）：数组必须先排序

多维数组：数组的元素本身又是一个数组对象，例如二维数组，三维数组，……

String 字符串对象
常用属性和方法
charAt 方法：字符串 string 的第 n 个字符
charCodeAt 方法：字符串 string 的第 n 个字符的 Unicode 编码
fromCharCode 方法：指定编码的字符的新字符串
concat 方法：拼接字符串，不如用+号方便
indexOf 方法：查找返回子字符串的下标，如果没有返回-1
lastIndexOf 方法，同上，从后向前找
length 属性：返回字符串长度
substr 方法：返回指定下标范围的子字符串
substring 方法：类似 substr，但是第二个参数代表终止下标
toLowerCase, toUpperCase 方法：大小写转换
split 方法：按照分隔符，将一个字符串分割为一个数组

Math 数学对象
常用属性和方法：
floor 方法：返回一个数字的不大于本身的最大整数
round 方法：四舍五入取整
ceil 方法：进位取整，小数位有有效数字，就进位
random 方法：返回一个随机小数，一定是大于等于 0 小于 1 之间
产生任意两整数之间的随机整数（包括两整数本身）公式:
　　parseInt((大值-小值+1)*Math.random()+小值)

Date 对象
常用方法：
构造函数：new Date( )默认当前系统时间 new Date(毫秒数) new Date(“2013-7-18”) new Date(year, month, day, hours, minutes, seconds, ms)

getYear/setYear 方法：返回或设置年份 有兼容问题 推荐使用 getFullYear 方法
getMonth/setMonth 方法：返回或设置月份 0~11 表示 1~12 月
getDate/setDate 方法：返回或设置天数
getHours/setHours 方法：返回或设置小时
getMinutes/setMinutes 方法：返回或设置分钟
getSeconds/setSeconds 方法：返回或设置秒
getDay/setDay 方法：返回或设置星期几 0~6 表示周日到周六
getTime/setTime 方法:返回或设置当前日期对象毫秒数，这个毫秒数代表的是从 1970-1-1 0:0:0 到这个日期间隔的毫秒数

定时器
window.setTimeout：间隔多少毫秒执行一个程序
语法：setTimeout(函数对象或者语句字符串,毫秒数); 推荐传函数，不推荐传语句字符串，效率低
window. setInterval：同上，但是是每隔多少毫秒执行一次程序

setTimeout 和 setInterval 方法都有一个返回值 timerId，代表这个定时器的唯一标识，可以用来取消定时器（clearTimeout clearInterval）

document.write 方法：如果潜入在网页的输出流中，那么将会和网页输出流形成同一个输出流，如果网页输出流已经结束，重新执行 document.write 方法，会产生一个新的输出流，网页原始内容会被覆盖。

所有的容器对象都具有两个动态属性：
innerHTML 属性：代表一个容器中的 html 内容，可以设置，也可以返回
innerText 属性：代表一个容器中的 text 文本内容，可以设置，也可以返回
innerHTML 属性：类似 innerHTML，但是是包含元素本身

中文在 gbk 编码下一个字符占 2 字节，在 unicode 编码下是占 3 字节
浏览器提交表单数据的时候会自动把中文转换为 **url 编码**
Tomcat 服务器默认的网络请求编码是 iso-8859-1 西欧编码，post 请求发送中文可以在后台代码中手动设置请求编码，但是 get 请求服务器是写死 iso-8859-1 编码，无法手动设置

Get 请求和 post 请求的区别：
1.get 把数据追加在 url 后面，以?参数名=参数值&参数名=参数……形式，post 请求把数据放到请求正文中
2.get 由于受到 url 长度限制，发送数据量小，post 发送数据几乎无限制
3.get 明文发送数据，不安全，post 数据放到请求正文中，安全
4.get 发送中文困难，post 发送中文容易

常用内置函数：
parseInt：字符串转为整数

parseFloat：字符串转为浮点数
isNaN：判断是否是一个非数字
eval：动态执行 JavaScript 语句字符串
escape/ unescape：基本编码解码，是 unicode 编码，不是 url 编码
encodeURI/ decodeURI：url 编码解码，常用
encodeURIComponent/ decodeURIComponent：url 编码解码，和上面的不同的是，&=?也会被编码

JS 能够动态读取设置一切 HTML 属性 元素对象.属性 例如 t1.value
注意：有两个 html 属性不能直接直接用原始属性名做动态属性名
class 需要写成 className
for 需要写出 htmlFor
单一属性，例如 checked selected disabled 动态属性是布尔值 true 和 false

JS 能够动态读取设置一切 CSS 样式 元素对象.**style**.样式 例如 t1.style.backcolorColor
这种写法操作的是行内样式

Js 获得对象的方式：
document.getElementById 方法：通过元素的 id 返回元素对象
document.getElementsByName 方法，通过元素的 name 返回一组元素对象，返回的数据类型是一个类数组（不是真正的数组，只具有下标和 length 属性，没有其他的 api 方法）
document.getElementsByTagName 方法：通过元素的标签名称返回一组元素对象，返回类型也是一个类数组
document.getElementsByClassName 方法：通过元素应用的 class 类返回一组元素对象，返回类型是类数组，这个方法有兼容性问题，高版本浏览器支持

html 允许静态或者动态的设置**自定义属性，自定义属性就是 html 本身没有的属性，也**可以通过 js 动态读取和设置，例如 <p abc=”123”>，在 html5 中规定，只要是自定义属性，必须用 data-开头，例如 data-abc 比较规范

自定义属性最好用以下方法操作，保证兼容性：
设置属性值：对象.setAttribute(‘属性名’,属性值)
返回属性值：对象.getAttribute(‘属性名’)

事件
事件=事件源 + 事件类型 + 事件处理（事件回调函数）

注册事件的方式：
0 级事件：
行内写法：在标签中注册事件 例如 <p oncliclk=”xxxx”>，只能调用全局函数
代码写法：事件源对象.事件名称 = 事件回调，推荐使用，耦合度低，调用灵活
2 级事件：
W3C 标准写法：事件源对象.addEventListener(‘事件类型’,事件回调函数[,是否为捕获型])

IE 专用写法：事件源对象.attachEvent(‘事件类型’, 事件回调函数)

移除 2 级事件：

W3C 标准写法：事件源对象.removeEventListener(‘事件类型’,事件回调函数[,是否为捕获型])

IE 专用写法：事件源对象.detachEvent(‘事件类型’, 事件回调函数)

常用事件：
onload：当对象加载完毕 例如 window.onload 代表网页资源加载完毕，包括外部引用的资源也加载完毕才触发事件
onclick：单击
ondblclick：双击
onmouseover：鼠标移上，鼠标穿过子元素，会重新触发事件
onmouseout：鼠标离开，鼠标穿过子元素，会重新触发事件
onmousedown：鼠标按下
onmouseup：鼠标抬起
onmouseenter：类似 onmouseover，但是鼠标穿过子元素不会重复触发事件
onmouseleave：类似 onmouseout，但是鼠标穿过子元素不会重复触发事件
onmousemove：鼠标移动
onfocus：获得焦点
onblur：失去焦点
onchange:当内容被改变，一般用在列表框中
onselect：当内容被选中
onkeydown：键盘按下
onkeyup：键盘抬起

this 关键字在事件中：
行内事件里的语句中用 this，代表当前标签元素对象
代码事件的**回调函数**中用 this，代表当前事件源对象（是当前注册事件的那个对象）

事件对象：event 对象，本身代表当前事件的一些状态，不同的事件状态也是不一样
IE 和 w3c 标准标准获得事件对象的方式不一致，w3c 标准下：回调函数的参数就代表事件对象，IE 下：window.event 代表事件对象

事件对象常用属性和方法：
type 属性：事件类型
事件源对象：target 标准写法 srcElement IE 写法
clientX, clientY 属性：鼠标坐标位置
button 属性：返回鼠标的按键，不同的浏览器返回值不一致
keyCode 属性：返回键盘按键的 unicode 码
ctrlKey,altKey,shiftKey 属性：返回是否按下 ctrl,alt,shift 键

事件流：事件传播的顺序
捕获型：从父元素传播到子元素（IE 不支持）

冒泡型：从子元素传播到父元素（默认）

阻止事件冒泡：

W3C 标准写法：事件对象.stopPropagation()

IE 专用写法：事件对象.cancelBubble=true

阻止事件的默认行为：

W3C 标准写法：事件对象.preventDefault()

IE 专用写法：事件对象.returnValue=false

行内事件，直接在最后加上 return false；就可以阻止事件默认行为

代码事件，直接写上 return false 也可以组织事件的默认行为

事件委托：利用事件冒泡的原理，通过父元素来捕获子元素的事件加以处理

动态添加的元素注册事件方式：

方式 1：每次添加元素后直接注册事件

方式 2：采用事件委托机制

方式 3：直接用行内事件

滚轴事件：

非火狐浏览器：事件 onmousewheel，通过 wheelDelta 属性获得滚轴状态，向上滚轴值为正数，向下滚轴，值为负数

火狐浏览器：必须使用 2 级事件注册，事件名 DOMMouseScroll，通过 detail 属性获得滚轴状态，向上滚轴值为负数，向下滚轴，值为正数

手动产生事件行为：大多数的事件都可以通过程序产生，例如 click() focus() select()……

DHTML：dynamic html 动态 htmnl，js 提供了对某些元素的对应的 api，例如 Image 类 Option 类，innerHTML 属性等

DOM：document object model 文档对象模型，js 提供了通用的动态操作元素的 api，dom 编程思想认为网页中每一个元素都是一个对象，也是一个节点，根节点就是 document，唯一的子节点就是 html，html 有两个子节点，head 和 body，以此类推

DOM 主要就是用来创建，插入，修改，删除节点的

DOM 操作的常用属性和方法：

createElement 方法：在内存中创建一个元素节点

createTextNode 方法：在内存中创建一个文本节点

appendChild 方法:在父节点中添加子节点

setAttribute/getAttribute 方法：设置读取节点的属性

insertBefore 方法：有两个参数，参数 1 表示插入的节点，参数 2 表示某个节点对象，参数 1 插入到参数 2 节点之前，如果省略参数 1，功能和 appendChild 一致（IE 下好用）

removeChild 方法：通过父节点删除子节点，必须是直接子节点，不能间接子节点

replaceChild(新节点,旧节点)方法：替换节点

parentNode 属性：返回当前对象的父节点

childNodes 属性：返回当前对象的所有子节点，返回类数组，注意的是行内的文本也算子节点
firstElementChild 属性：返回第一个元素子节点（不包含文本节点）
lastElementChild 属性：返回最后一个元素子节点
nextElementSibling 属性：下一个兄弟元素节点
previousElementSibling 属性：上一个兄弟元素节点
childElementCount 属性：返回所有的子元素节点的个数

nodeType 属性：返回节点的类型，返回值是 9 代表 document 节点，1 代表元素节点，3 代表文本节点，8 代表注释节点，11 代表文档碎片节点
nodeName 属性：返回节点的名称（一般用于元素节点）
nodeValue 属性：返回节点的值（一般用于文本节点或注释节点）

querySelector(css 选择器)：根据选择器返回 dom 对象，只返回匹配的第一个对象，兼容到 IE8
querySelectorAll(css 选择器)：同上，但是返回所有匹配的元素对象，返回类数组，兼容到 IE8

表格相关的 dhtml 方法
表格对象.createCaption()：创建表格标题对象
表格对象. createTHead()：创建表格头部
表格对象. insertRow()：创建并插入行
行对象. insertCell()：创建并插入单元格
表格对象.rows：返回所有的行的类数组
行对象.cells：返回当前行的所有单元格的类数组

文档碎片：createDocumentFragment()创建一个临时的容器对象，可以在减少渲染次数的同时，不产生垃圾标签

BOM（Browser Object Model）浏览器对象模型：不属于任何脚本语法，属于浏览器本身的对象，可以被任何脚本调用

最大的对象是 window 对象，代表当前的浏览器窗口
常用属性和方法：
alert()：对话框
prompt()：输入框
confirm()：确认框，确定返回 true，取消返回 false
close()：关闭当前窗口
open()：开启新窗口
print()：调用浏览器打印
scrollTo(水平滚动条位置,纵向滚动条位置)：动态设置窗口的滚动条位置
showModalDialog(窗口参数)：模特窗口
document 对象：代表当前文档窗口
navigator 对象：代表当前客户端的信息 最重要的属性 userAgent 包含了客户端操作系统和浏览器信息

screen 对象：代表客户端的屏幕

height,width：垂直和水平的屏幕分辨率

dpi 像素密度：电子屏幕一平方英寸里的物理像素数量 ppi 打印机像素密度

history 对象：代表浏览历史，注意操作就是前进和后退

back()方法：后退按钮

forward()方法：前进按钮

go(参数)方法：前进(正数)或后退(负数)几个页面

location 对象：代表地址对象

reload()：刷新当前网页

replace()：替换当前文档内容（不可后退）

assign()：加载一个新的文档内容（可以后退）

hash 属性：可以设置或者读取锚点

href 属性：设置或者读取当前 url

search 属性：代表 url 的？号后面的请求参数

获得 body 对象：document.body

获得 html 对象：document.documentElement

元素的大小和位置：

视口的宽度和高度：

document.documentElement.clientWidth

document.documentElement.clientHeight

网页的实际宽度和高度：

document.body.scrollWidth

document.body.scrollHeight

网页元素的实际宽度和高度

元素对象. clientWidth 元素对象. clientHeight （内容+padding）

元素对象. offsetWidth 元素对象. offsetHeight（内容+padding+border）

网页元素距离父元素的偏移量：如果有父元素且父元素设置了 postion:relative absolute fixed，相对于父元素的偏移量，否则相对于网页原点的偏移量

元素对象. offsetLeft 元素对象. offsetTop

网页元素距离视口原点的偏移量：

元素对象. getBoundingClientRect() 返回对象，通过这个对象 top left right bottom 四个属性可以返回四个边界距离视口原点的偏移量

获得窗口滚动条卷去的距离

垂直滚动条 document.documentElement.scrollTop || document.body.scrollTop

水平滚动条 document.documentElement.scrollLeft || document.body.scrollLeft

鼠标相对于事件源的位置

事件对象. offsetX 事件对象. offsetY

表单元素相关操作：

获得表单元素对象

document.表单元素 name 名称

document.forms[下标]

document.getElementById('表单 id')

表单对象的常用属性方法

length 属性:返回表单元素的个数

elements 属性：返回所有表单元素的类数组

reset()方法：重置表单

submit()方法：提交表单

文本框对象：

value 属性：代表当前用户在网页中输入的值

defaultValue 属性：代表文本框的默认值

列表框对象：

value 属性：当前选择的 option 的 value 值

selectedIndex 属性：当前选择的 option 的下标，如果没有选择任何 option，返回-1

length 属性：当前 option 的数量

options 属性：所有 option 的集合类数组

Option 对象

动态增加 option：

var opt = new Option('option 文字','value 值');

select 对象.add(opt); //添加方法 1

select 对象. options[终止下标+1] = opt; //添加方法 2

修改 option：

select 对象. options[指定下标] = option 对象;

删除 option：

select 对象.remove(指定下标); //删除方法 1

select 对象. options[指定下标] = null;//删除方法 2

清空 option：

select 对象. options.length = 保留数量;

value 属性：value 值

text 属性：标签的文字

表单验证：验证表单中的输入数据是否合法，验证分为前端和后端

前端验证：用户体验好，不需要走网络请求，效率好，很容易被绕开验证

后端验证：需要每次验证，都发送网络请求，效率低，用户体验差，优点不容易被绕开验证，相对比较安全

表单提交事件 onsubmit

正则表达式：用一系列的特殊字符来做字符串的规则匹配

创建正则表达式对象的两种方式：

```
var 变量 = new RegExp('匹配模式','参数');  //正常创建正则表达式对象
var 变量 = /匹配模式/参数;  //字面量写法
```

参数 g：全局匹配，一般用在 replace 方法
参数 i：忽略大小写

正则的方法：

test()：判断字符串是否匹配模式

exec()：返回一个对象，包含三项数据，匹配的字符串，下标，原始字符串，找不到返回 null，匹配的字符串是按数组返回，如果有子表达式，每个子表达式匹配的内容也会加到数组中

String 中支持正则表达式的方法：
replace(要替换的字符串或者模式，替换为的字符串)方法：替换字符串，默认只会替换第一个匹配的，如果要全局替换，必须用正则表达式配合 g 参数
match()方法：返回一个数组，存储所有匹配的值，如果没有任何匹配，返回 null
search()方法：返回第一个匹配的内容的下标，如果没有匹配的，返回-1

正则表达式元字符

\ 转义字符

^ 表示开始 例如 /a/ 匹配 bab 但 /^a/ 匹配 ab 不匹配 bab

$ 表示结束

表示匹配某些字符的元字符

. 表示任意字符 包括中文

[abc] 表示 a 或 b 或 c 其中一个字符

[^abc] 必须不是其中任何一个字符

[a-z] 小写字符 [A-Z]大写字母 [a-zA-Z]所有字母 也可以是[c-e][X-Z]

[0-9] 数字

[a-zA-Z0-9] 字母和数字

[^a-z] [^A-Z][^0-9] 非字母，非数字

\s 空白（空格，制表位，换行，换页，回车）

\S 非空白

\d 数字 相当于[0-9]

\D 非数字 相当于[^0-9]

\w 字母或数字或下划线 [0-9][a-z][A-Z]_

\W 非字母数字下划线

表示匹配次数的元字符

* 0 次到多次

+ 1 次到多次

? 0 次或 1 次

{n} 正好 n 次

{n,} 至少 n 次

{n,m} n 次到 m 次

其他元字符

| 逻辑或 要打小括号

(表达式) 子表达式,改变优先级

? 非贪婪模式，一般用于 replace 方法，尽可能的少匹配

Cookie：通用的客户端数据存储技术，在浏览器端以纯文本的**键值对**形式存储数据，必须基于服务器运行实现（本地运行无法实现）

Cookie 分为两种形式：

会话级：存储在浏览器的会话进程中，生命周期和会话一致

硬盘级：永久存储在浏览器的缓存文件夹中，是一个文本文件，但是需要设置失效时间，否则默认就是会话级。失效时间必须以;expires=GMT 时间格式

一个域写入的 cookie 数据，只能被相同的域读取，不能被其他的域读取，例如百度无法读取搜狐写入的 cookie，搜狐无法读取新浪写入的 cookie，如果写入的是中文数据，需要进行编码，然后读取的时候进行解码。

Cookie 在浏览器中有容量限制，有条数限制，每次网络请求都会放到请求报头中一起发给服务器。

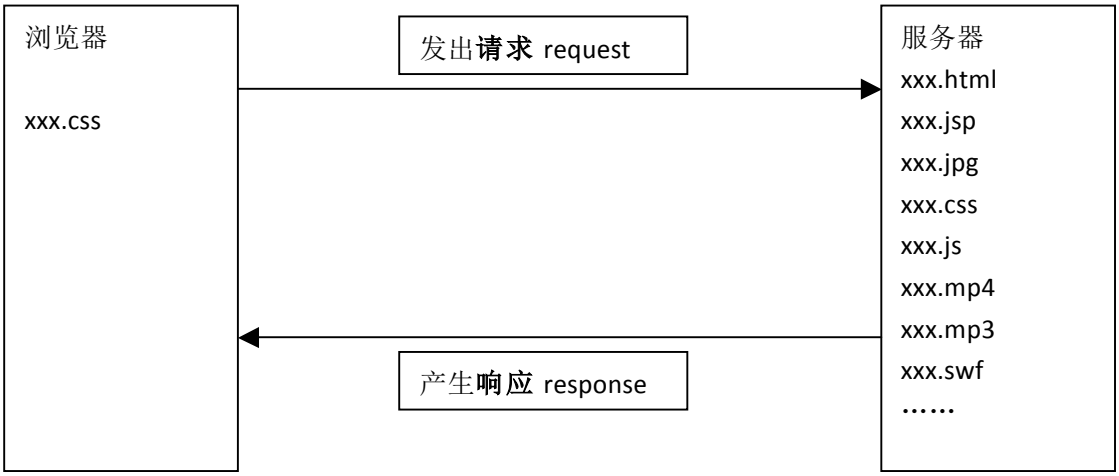
JS 中操作 cookie 的语句：document.cookie 属性

HTTP 网络通信协议

TIP/IP：传输控制协议/网际协议 有很多子协议 http ftp https

HTTP：超文本传输协议 简单说就是传输 html 数据的，HTTP 协议是无连接，无状态的，客户端发出**请求**，连接服务器，得到**响应**，立即断开。

请求 request-响应 response：必须由客户端主动发出请求，服务器才能产生响应，一般来说，请求是客户端向服务器端发送数据，而响应是服务器端向客户端发送数据



如果请求的是一个静态资源，服务端会直接响应数据到客户端，如果请求的是一个动态资源，比如 java,php,asp，服务器端会先在服务器端执行完毕这些动态内容，然后把执行的结果响

应到客户端。

HTTP 请求的三部分组成：

请求行：请求方法（GET POST PUT DELETE）请求的 URI 协议和版本 例如 http1.1

消息报头：客户端的一些相关信息，例如语言，浏览器，操作系统，压缩方式……

请求正文：GET 请求没有请求正文，POST 的请求正文存储了发送的数据

响应状态码：

200 请求响应

304 从浏览器本地缓存读取数据

404 资源找不到

405 无法请求对应的 get 和 post 处理程序

500 服务器程序运行错误

增删改查的四大操作（CRUD）：restful 风格

GET 查询

POST 增加

PUT 修改

DELETE 删除

数据交换标准

1. 字符串用特殊符号分割
2. CSV 格式
3. Xml 格式
4. Json 格式

Xml（Extensive Makeup Language）可扩展的标记语言，特点是自定义标签，而不是像 html 那种固定标签，主要用途做通用数据交换格式和配置文件，缺点是标签本身占用流量太多，解析 xml 数据比较困难

语法特点：

1. 文件头部声明 <?xml version="1.0" encoding="UTF-8"?>
2. 标签必须是成对的，如果是空标签，可以写成<xxxx/>形式的一个标签
3. 标签区分大小写
4. 文档中必须且只能有一个根元素
5. 属性值必须加双引号或者单引号
6. 如果有应用 dtd 或者 schema 规则文件，必须符合 dtd 或者 schema 规则
   a) Dtd 是早期规则语法
   b) Schema 是后期规则语法，和 dtd 语法并存

JSON：基于 JavaScript 语言的轻量级的数据交换格式(JavaScript Object Notiation)，最常见的就是和后台的数据通信

基本语法特点：就是和字面量的对象语法非常类似，用{}表示一个 json 对象，用[]表示 json

数组对象，但是属性名称需要加双引号，只能有属性，不能有方法

Json 字符串转换为 json 对象：JSON.parse() //ie8 或以上支持
Json 字符串转换为 json 对象：eval(‘(’ + json 字符串 + ‘)’) //ie6 或以上支持

Json 对象转换为 json 字符串：JSON.stringify() //ie8 或以上支持

函数高级扩展
函数本身就是一个对象，可以作为参数传递，也可以作为返回值返回
重载的概念：就是方法（函数）的名称相同，参数列表不同
在 JS 中没有重载这一说，重复的函数或者方法声明，以最后声明的为准

arguments 对象：在 function 内部可以获得一个 arguments 对象，通过这个对象就可以获得所有的**实参**数据，这个对象是一个类数组

获得形参的数量：函数对象.length

函数对象的 call 方法: 函数对象.call(上下文对象,可变长参数……)，改变函数的调用者的这个对象
函数对象的 apply 方法：函数对象.apply(上下文对象,参数数组……)，功能同上

callee 是 arguments 对象的一个对象，通过 callee 可以调用本身的函数，简单的说，callee 代表当前函数本身

caller 是函数对象的一个属性对象，永远代表调用了当前函数的那个函数对象

一个 function 中的变量，不论声明在声明位置，在实际运行期间，都会提升到函数的开始位置声明

闭包：一个正常的 function 调用，里面所有的局部变量在 function 调用结束之后，都会被销毁，但是如果局部变量**被内层作用域所引用**，即时外层函数执行完毕也不释放，这就是闭包，缺点是会造成内存泄露（内存使用完不及时释放，就叫内存泄露）

Error 异常处理：如果 js 代码出现运行错误，默认 js 引擎就会终止程序，错误行下面的代码是没有机会执行的。

手动异常处理：在发生异常之后，程序可以继续向下执行

```
try {
    可能会发生异常的代码
} catch(e) {
    异常处理代码（如果发生异常，执行的语句） e 就是当前的错误对象
}
```

手动抛出（发生）异常：手动创建错误对象，抛出给 js 引擎，让程序发生错误

throw 异常对象；

原型 prototype

每个类型（例如 Date,Error,Array,Number……包括自定义类型例如 Person）都有一个属性，这个属性的名字叫做 prototype，这个 prototype 指向一个对象，这个对象就是这个类的原型对象，里面默认有两个属性，constructor，__proto__，如果把属性或者方法定义到类的原型对象中，不管创建多少个对象，这些属性和方法仅存在一次，比较节省内存，一般来说，只会把方法定义到原型对象中。

constructor 永远代表当前的 function 对象

__proto__任何对象实例都拥有这个属性，这个属性指向这个对象的那个类的原型对象，不建议在程序中直接使用__proro__这个属性引用，很多浏览器不允许直接调用。

原型对象本身的类型是一个 Object 类的对象

原型链：基本原理就是对象的__proto__，当一个对象调用某个成员（属性，方法）的时候，首先先查找当前对象实例，没有再查找当前的原型对象，如果还没有，在继续向上层查找，会一直查找到 Object 类型，如果还没有，直接报错

in 语法：检测某个对象中（包括原型对象）是否拥有某个成员 ‘属性或者方法名称’in 对象

hasOwnProperty 方法：判断一个对象中（不包括原型对象）是否拥有某个成员

如果希望任何一个对象都能拥有某个方法，那么这个方法应该定义到 Object 类的原型中，但是这样做对团队开发会有影响。

作用域链：一个 function 中使用一个变量数据，如果当前作用域没有，就会向外层作用域查找，直至查找到全局作用域，要尽量减少对外层作用域的查找，比较耗费资源，尽量用传参来代替作用域查找。

继承：一个类（子类）从另一个类（父类）中获得属性和方法，优点就是避免重复定义同样的属性和方法。

isPrototypeOf 方法：判断当前对象是否继承了某个类的原型

propertyIsEnumerable 方法：判断对象的某个成员是否允许被遍历，默认的成员都是可以被遍历的，如果不想被遍历，需要对成员单独做特殊设置。

第三方技术：在原生的程序基础上扩展出来的技术，大多数都是由某些公司和个人开发的开源产品

如果一个第三方技术成了规模，已经完全代替原生的程序，就可以称作是一个框架，框架的主要目的就是简化开发

Jquery

Jquery 是一个 JavaScript 库，有些时候也成为 JS 框架，目的就是简化原生 JS 开发，集成了 JS 的常用功能，类似的框架还有 ExtJS，Dojo，Prototype，YUI，

Jquery 的版本：1.xx 兼容 IE6,7,8　2.xxx 去掉了冗余代码，不再支持 IE6,7,8 最新 3.xx

一般开发的时候用正常非压缩班，生产环境（上线）尽量使用压缩版 jquery.min.js

核心函数（全局函数）是 jQuery，和$是等价的，传入对应的参数，返回 Jquery 对象，用 jquery 对象才能调用 jquery 的 API

Jquery 对象不能直接调用原生对象的任何 API，例如 innerHTML click() focus()

Jquery 对象并不是原生 js 对象，它是封装了原生对象并且附加了一些功能，产生的一个新的对象实例

把 jq 对象转换为原生的 js 对象：jq 的 get 方法，返回值是一个类数组

Jq 的文档就绪事件

写法 1：$(document).ready(回调函数);

写法 2：$(回调函数);

window.onload 和$(document).ready 的区别：

window.onload：只能注册一次，必须当前 html 的 dom 全部加载并且所有的外部资源（图片，音频，视频）全都加载完毕才会触发

$(document).ready：注册多次，仅仅 html 的 dom 全部加载完毕就会触发，原理是 JS 的 document.onreadystatechange 事件

Jquery 的选择器语法和 css 选择器语法完全一致

获得选择器得到对象的个数：jq 对象.length　jq 对象.size()

通过下标访问元素：eq 返回 jq 对象 get 返回 js 对象

大多数的选择器语法都有对应的方法实现

操作属性

attr：如果属性不存在，读取属性，返回 undefined，适合非单一属性和自定义属性

prop：适合单一属性

Jquery 的循环遍历语句

$.each(对象或者数组,function(index, el) {});
JQ 对象.each(function(index, el) {});

同时阻止事件冒泡和事件默认行为：在事件回调函数中 return false;

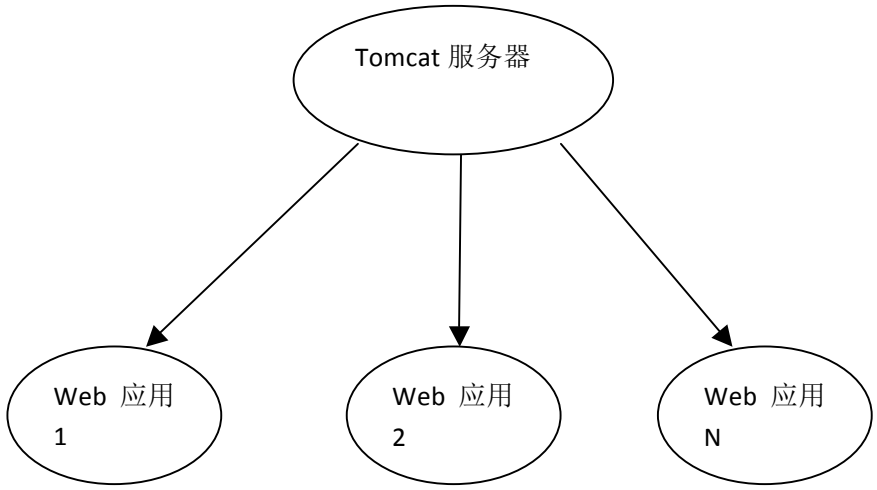
动态添加的元素注册事件问题的解决：
1.在动态添加的元素中直接写行内事件，但是回调函数必须是全局作用域
2.每次动态添加一个元素，就立即给这个元素注册事件，缺点是注册的事件太多
3.使用事件委托，用父元素处理子元素事件，但是需要判断是否是要处理的子元素

Jquery 的事件委托语句的历史
最早是 live 语句，后来是 delegate 语句，最后是 on 语句

Js 插件：就是第三方编写的 js 功能（广义上看，jquery 就是大插件），js 插件有上千个（轮播图，模态层，拖拽，上传，菜单，幻灯片，布局……），使用插件最基本的用法就是直接应用插件的 css 和 js，然后调用插件功能

按语法划分，分为 js 插件（纯 js 编写）和 jquery 插件（用 jquery 语法编写，需要依赖 jquery）

客户端和服务器端通信



关于相对于根目录的斜杠问题：
1. 如果是在服务器端执行，/代表的是当前 web 应用根目录
2. 如果是在客户端浏览器执行，/代表的是整个服务器根目录

浏览器本身就有请求服务器的功能，提交表单，点击链接，输入 url 回车都可以发出请求，每次请求都默认会跳转到对应的 url 画面，用户体验不友好

Ajax（Asynchronous JavaScript and XML）：异步的 javascript 请求，是无刷新请求，发出请求的时候浏览器不会发生任何跳转，JS 是依赖的 XMLHttpRequest 对象（ajax 核心对象）实现 ajax 请求

返回 XMLHttpRequest 对象

标准写法： new XMLHttpRequest();

IE 写法：new ActiveXObject("Microsoft.XMLHTTP");

$.get 和$.post 以及$.ajax 方法在 jquery1.5 之后都会返回一个对象，这个对象一般认为是 promise 对象，代表执行的状态，一般三种状态：成功 失败 执行中，可以根据这三个不同的状态，做不同的回调

同步和异步请求的区别：

同步：顺序执行，当一个程序流程执行完毕，才能执行下一个程序流程，例如一个 ajax 请求响应结束之后，才能执行其他操作

异步：允许并发执行，多个程序流程可以同时执行，互相没有任何影响（多线程）

$.Deferred() 工具方法可以返回一个延迟对象，通过延迟对象，可以得到 promise 对象，可以针对于不同的延迟操作作监听和回调。

大部分浏览器是不支持本地 ajax 操作的，也就是必须通过 http 服务器来发出 ajax 请求，默认情况下，只有火狐浏览器支持本地 ajax 请求

跨域请求：一个域名下的程序发出一个请求，请求的 url 是另一个域名下的资源，这就是跨域请求，所有的浏览器默认都不允许 ajax 跨域请求。

解决跨域问题的方案：

1. 强行改变浏览器设置，让浏览器允许跨域 例如，在谷歌浏览器中设置 --disable-web-security 启动参数 （意义不大）
2. 在服务器端增加允许跨域请求的响应报头，设置 Access-Control-Allow-Origin 为*
3. 利用 jsonp 技术，前端和后端配合实现跨域 ajax

   所谓的 jsonp 技术，就是利用浏览器并不禁止外联 js 是另一个域的 js，所以 jsonp 跨域

请求原理就是<script src="其他域的 js">，而被外联的 js 是一个函数调用语句，通过实参把数据传递给请求客户端

4. 利用后台代码发出跨域请求（java php python nodejs asp.net），然后再返回一个响应，用前端的 js 发出本域请求

关于浏览器请求的缓存问题：浏览器发出请求的时候，会把请求的结果自动复制一份放到客户端本地，这就是浏览器缓存，当下次发出相同的 url 请求的时候，并且如果服务器端资源最后修改日期和本地缓存的资源最后修改日期相同，就会走本地资源（304），如果是动态生成的内容就会出问题（修改日期不变，内容已经发生变化），一般在请求的后面追加一个不重复值的参数就可以，一般都是当前时间，这样永远不会走本地缓存。

Jquery 插件分为全局对象插件和局部对象插件

全局对象插件：对应$.extend，调用是用$全局对象调用

局部对象插件：对象$.fn.extend，调用是用 jquery 的 dom 对象调用

利用 ajax 请求其他画面内容显示到当前网页，需要注意其他画面内容绝对不能有 html 结构（head body html style script），否则很容易和主画面样式和代码冲突，最佳的选择应该用 iframe 去显示其他画面，iframe 是独立的浏览器窗口。

如果 iframe 画面中需要操作主画面，需要通过 window.parent 获得主画面的 window 对象

HTML5

Html 的两大版本：

Html4.01 对应时代的 css 版本 css2.1

Html5 对应时代的 css 版本 css3.0

Html5 在 html5 的基础之上，增加了一些标签，增加一些 API（6 万多个），不兼容低版本浏览器，在手机浏览器上支持相对良好

语义化标签：例如 table img ，但是使用最多的排版标签 div 没做语义化，html 中针对于 div 新增同样功能但是带语义的标签

article 文章

section 区块

aside 侧边栏

header 头部

footer 底部

nav 导航

hgroup 标题组

低版本的 IE 浏览器不兼容新增标签，需要用第三方 js 来实现兼容，例如 IE8

新增表单元素：在移动端浏览器，会根据不同表单元素，弹出不同的虚拟键盘
搜索框 search
列表提示 datalist
数字框 number
滑动条 range
颜色框 color
电话框 tel
网址输入 url
日期框 date
时间框 time
日期时间框 datetime-local

度量 meter
进度条 progress

多媒体

在 html4 中播放多媒体（音频，视频）需要第三方插件支持（例如 QQ 影音，暴风影音，……），浏览器本身没有任何播放能力，使用的 embed

在 html5 中，浏览器本身就有播放音频视频的能力（高版本支持 H5 的），音频 audio 标签，视频 video

Audio 标签：音频播放
html 属性：
controls 显示控制面板
autoplay 自动播放
loop 循环播放
src 文件路径

Video 标签：视频播放 html 属性同上

音频和视频的 API:

相关方法
play()：播放
pause()：暂停

相关属性
paused：是否当前是暂停状态
currentTime：设置或者返回当前播放的进度时长
playbackRate：播放速度，默认是 1
muted：是否静音
volume：音量 0~1

duration：返回播放总长度

相关事件

play 播放
pause 暂停
ended 播放结束

BASE64 图片：把二进制图片转换为 base64 编码的文本，然后在 img 标签中使用，优点可以减少外部图片的文件的请求次数，一般用于的小的图片文件

画面 Canvas 和 SVG 图形图像

Canvas 是位图图像，一整幅画布，由像素点组成，放大失真，不能单独调整画布中的元素

SVG 是矢量图图像，每个元素都是单独的 dom 节点，是使用 xml 作为节点，可以单独调整设置。

拖放

draggable="true" 允许元素被拖放
ondragstart 当元素开始被拖动
ondragover 当拖动到元素上方
ondrop 在元素中释放鼠标（放下被拖动的元素）
ondragenter 鼠标拖放对象进入到拖放区域
ondragleave 鼠标拖放对象离开拖放区域

事件对象中有一个属性 dataTransfer，这个对象存储传输一些数据 setData（）设置数据 getData 读取

文件操作

Html5 为我们提供一种和本地文件（客户端文件）的方式，一些文件操作 API

操作文件：最常见的方式用表单中的文件域选取文件，默认只能选择一个文件，如果需要多选，要增加 multiple 属性，如果要控制选取文件的类型，需要增加 accept 属性，设置 MIME 类型，通过 onchange 事件来检测是否选择完毕

另一种方式，就是利用 html5 拖放获得文件信息。

读取文件内容：

FileReader 类：文件读取对象，异步读取文件内容

主要方法：

readAsDataURL () 读取二进制内容
readAsText() 读取纯文本内容

地理定位：HTML5 中唯一能够调用硬件的功能，可以通过 PC 的 IP 地址（至今未实现），或者手机移动端的 GPS 模块（最常用的），来得到用户的经纬度坐标

通过 navigator.geolocation 对象的 getCurrentPosition 方法来获得用户位置

WEB 本地存储：为了代替 Cookie 实现客户端数据存储，相对于 Cookie 的区别是，不会在每次请求的时候，把数据自动发送到服务器端，节省流量。

localStorage 方式：永久性存储，没有失效时间

sessionStorage 方式：会话级存储，会话关闭就销毁

和 Cookie 一样，WEB 本地存储的数据由哪个域写入，就只能由哪个域读取

相关的 API:
length 属性：返回数据的条数
setItem()：设置或增加数据
removeItem()：删除指定 key 的数据
getItem()：返回指定 key 的值
key()：返回指定下标的 key

应用缓存：利用 html5 设定某些文件缓存，需要通过单独的配置文件进行配置
如果缓存的文件进行了修改，必须修改配置文件，否则文件在浏览器端不更新

WebWorker 多线程
CPU 在同一时刻只能运行一个程序（单核），一个应用程序可以由多个独立的进程组成，一个进程可以由多个独立的线程组成，线程是独立的程序的最小单位，可以并发执行。

JS 默认没有真正意义上的多线程，setTimeout setInverval 只是模拟了多线程，并不能真正并发执行，如果 JS 有任何代码在执行中，定时器语言必须等待 JS 代码执行结束才能执行。

WebWorker 技术是真正意义上的多线程，可以在其他 JS 脚本运行期间并发同时执行，不影响页面效率，需要基于服务器运行，且运行的代码必须外联的 js 代码，一般我们会把比较耗费时间的代码放到 WebWorker 里面运行。

需要注意：WebWorker 代码不能直接操作 dom，需要把值回传给主程序（调用 postMessage 方法），通过主程序的回调函数（onmessage）来操作 dom。

推送（push）：服务器在客户端没有主动发出请求的情况下，向客户端发送消息
常见的 WEB 推送技术：

1. 长连接技术，推翻 http 协议的无状态的特征，不是请求响应结束之后立即断开连接，而是客户端浏览器和服务器端一直保持连接，缺点就是服务器压力太大。
2. HTML5 的服务器发送事件（server-sent event），属于单向服务器发送数据到客户端，注意响应的数据格式必须是 data:xxxx，否则客户端无法接收。
3. WebSocket 双向通信的推送，需要服务器本身支持，客户端需要使用 WS 协议发送请求
4. **Ajax 轮询实现假推送**：用定时器每隔一段时间发送一个 ajax 请求，得到响应，兼容性最好，缺点是耗费很多无用的 ajax 请求浪费资源，响应的及时性不如真正的推送技术。

WEB SQL ：第一代客户端浏览器数据库，目前已经被淘汰，和正常服务器端数据库类似

IndexedDB：索引数据库，和 WEB SQL 数据库相比，用 key 和 value 方式存储数据，和本地存储非常类似，操作比较繁琐，用的较少

BootStrap

BootStrap 是一个 css 框架，内置了大量的写好的样式，有两个核心功能：栅格布局和响应式布局，2.x 版本和 3.x 版本语法差别非常，最新的是 4.x
BootStrap 也提供了很多 js 功能，例如轮播图，模态层，按钮提示等，但是功能比较弱

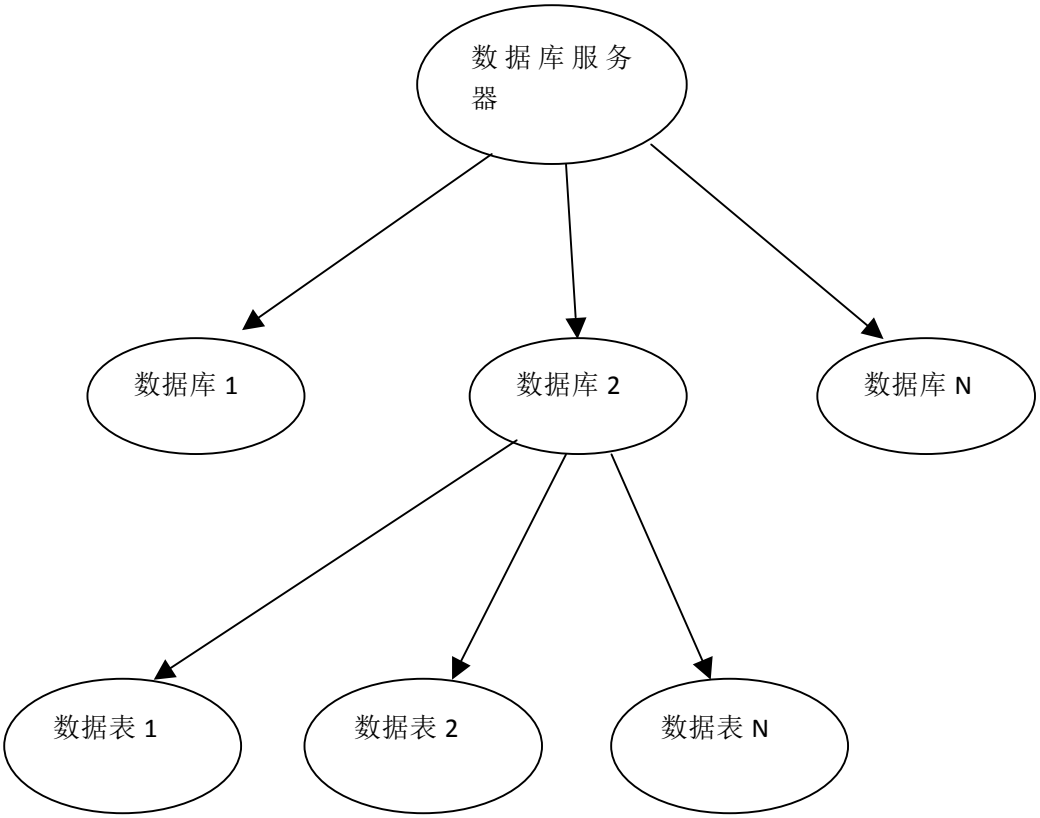
布局最外层容器
.container 固定宽度 1170px 970px 750px 和视口等宽通栏
.container-fluid 通栏宽度

栅格布局：把整体容器分成 12 列，然后控制子元素的列数，形成布局
分为 lg md sm xs 四个临界点宽度，只有大于等于临界点宽度，才能维持栅格布局，否则会自动堆叠排列

项目架构图

数据库 database，简称 db，数据库管理系统 dbms

数据库常见厂商：甲骨文 Oracle IBM DB2 微软 Sql Server 甲骨文 Mysql



数据表的组成：列（字段） 行（记录） 列值（字段值）

数据表设计原则：每个表必须有一个 id 列，表示唯一值（在表中不能重复），这个列一般都设置为主键列（自动限制重复值），作为其他表引用的列值

自增列：自动产生列值，不需要用户手动输入，是一个流水号，一般主键列的 id 都是自增列。

数据操作：

查询操作：

select * from 表名; 查询一个表中的所有列
select 列名 1,列名 2,…… from 表名; 查询一个表中某些列
select * from 表名 where 条件表达式; 按照条件过滤数据
select * from 表名 order by 排序字段 [desc]; 排序 desc 表示降序

插入数据操作：

insert into 表名 [(列 1，列 2，……)] values (值 1，值 2，……);
列列表可以省略，默认就是所有列
如果列值是自动生成，需要用 default 关键字代替

修改数据操作：

update 表名 set 列名=列值,列名=列值,…… where 条件表达式

删除数据操作：

delete from 表名 where 条件表达式;

软件开发的前后端结合方式分为两种：

一、传统开发，由后端代码（java php .net）直接生成数据画面，利用 jsp asp 动态网页技术，在网页中直接嵌入后端代码，缺点是耦合度太高，页面制作人员必须得前后端都写。

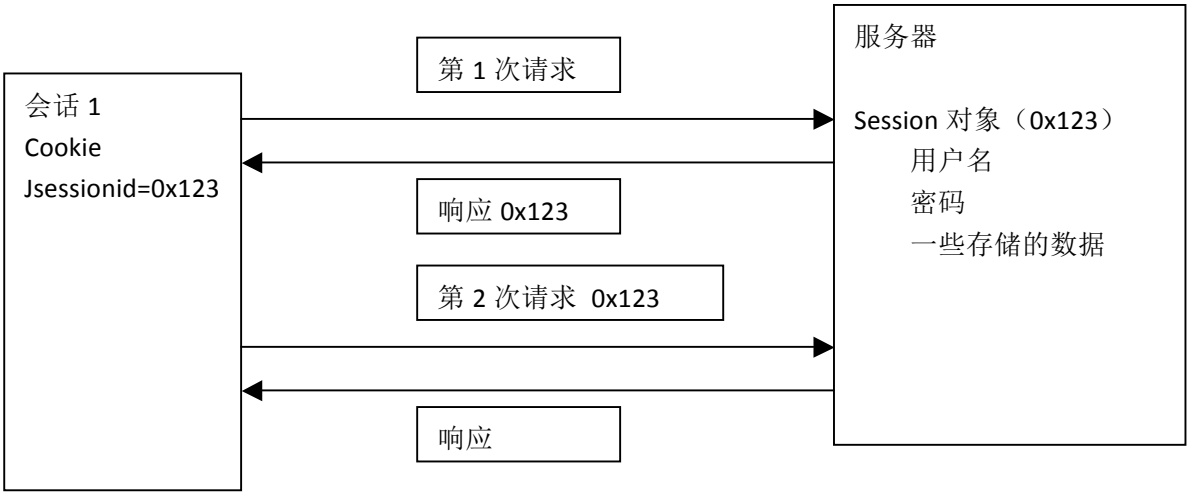
二、分离式开发，前后端的代码完全分离，前端用 ajax 对后端进行数据访问

Session 机制：session 是服务器端存储数据的技术，用于会话级存储

当浏览器会话第一次发出请求的时候，服务器端会在内存中创建一个对象，这个对象有一个唯一的 id（称为 sessionId），服务器会产生响应的时候把这个 sessionId 通过会话级 Cookie 形式响应给客户端浏览器，当第二次请求的时候客户端浏览器会把这个 sessionId 发送到服务器，服务器判断如果这个 sessionId 是存在的，则不会创建新的 session 对象，而是使用原理的 session 的对象。

从会话最后一次发出请求开始，如果超过 XX 分钟（一般是 30 分钟），没有发出任何请求，服务器端的对象 session 对象自动销毁

和 Cookie 的区别是：Cookie 是存储在客户端浏览器中，Session 存储在服务器端，只是把 sessionId 以会话级 Cookie 的形式存储在客户端。



APP 开发的三种形式：

1.纯原生开发（NativeApp）：利用 Java 开发安卓，利用 ObjectC，Swift 开发苹果，缺点不能跨平台（需要做安卓版和 IOS 版），开发周期过长，UI 开发过于繁琐，优点用户体验流畅，可以调用手机原生功能（硬件加速 GPU，摄像头，通讯录，麦克风，重力感应，蓝牙传输，GPS，SD 卡读写）

2.纯 WEB 开发（WebApp）：利用 HTML5，CSS3，JavaScript 开发网页的 APP，一般需要通过手机浏览器打开，其实本质上是手机网站，优点跨平台，开发周期快，不需要升级 APP，缺点，用户体验差，尤其是低端安卓手机（随着手机硬件的提示，将来不是问题），无法调用手机原生功能，费流量（因为网页都是在服务器端，每次打开界面都需要下载）

3 混合开发(HybridApp)：在 APP 中嵌入浏览器组件（安卓 WebView IOS UiWebView），通过这个组件直接在 app 中访问网页，网页中的 UI 组件可以通过 JS 接口访问原生的安卓和 IOS 功能

移动端开发框架

纯 UI 框架：bootstrap JqueryMobile mui ionic

调用原生功能混合开发框架：

Phonegap（cordorva） js 调用框架封装好的原生功能

React Native js 转换为原生代码

Html5plus 国产混合开发框架 开发工具 hbuilder

ApiCloud 云平台直接生成 app 再下载到本地

Appcan 国产混合开发框架

Weex 阿里出品混合开发框架

其他：zepto.js 移动端的 jquery，去除了一些移动端用不到的功能，减小体积，增加了移动端的事件，例如触摸，滑动，长按

PC 端 UI 框架：jquery ui jquery easy ui

AngularJS

angularJS 是谷歌出品的一个基于 MVVM 架构的 JS 框架，比较有特点的地方：MVC、模块化、自动化双向数据绑定、语义化标签（自定义指令）、依赖注入，页面路由等等。

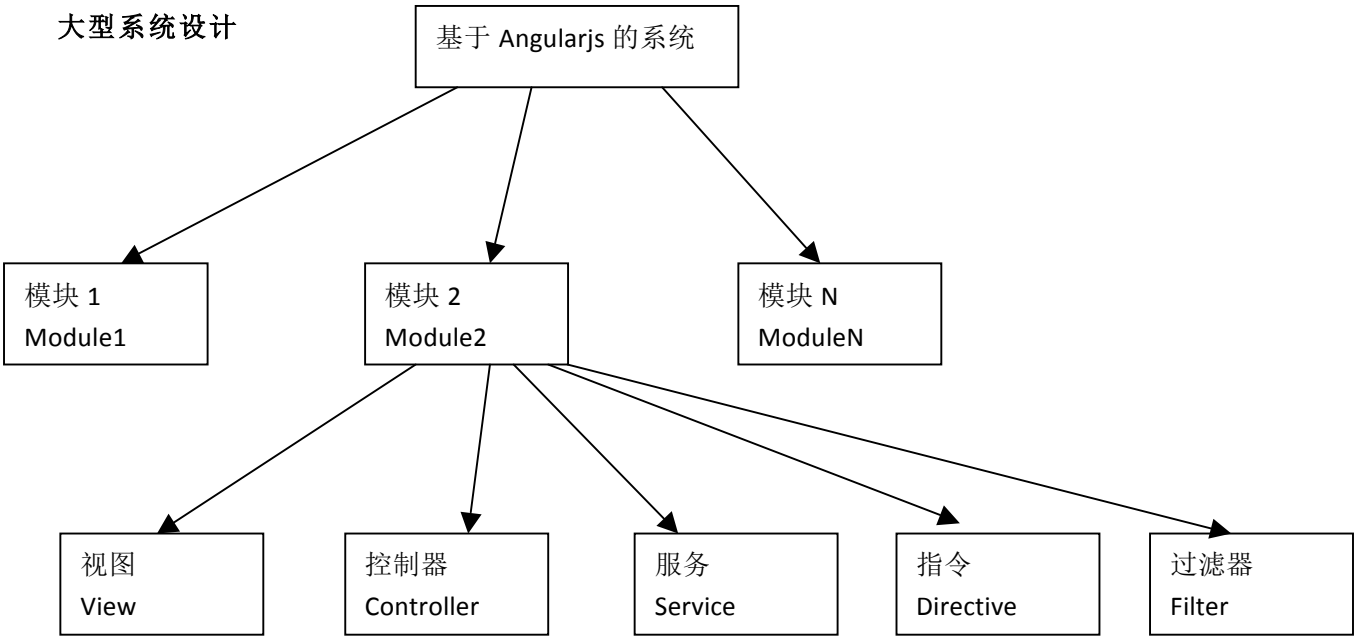
AngularJS 版本分为 1.x 和 2.x，语法几乎完全不一样，2.x 主要特点是适合移动端开发，语法变得简单。

类似的框架：knockout（最早期） backbone react.js（facebook 出品） vue.js avalon（国产框架） ember.js

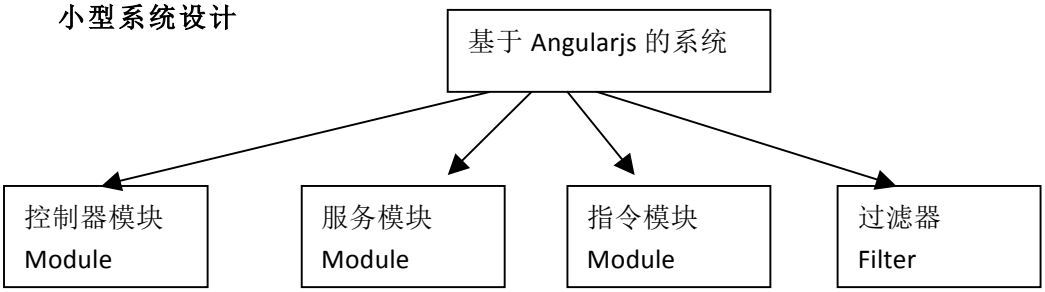
Ng 中一切都是基于数据模型（model）！

Jquery 适合做页面的动态效果，AngularJS 更适合做增删改查的数据操作。

大型系统设计



小型系统设计



视图不是 angularjs 的 js 代码一部分，它就是网页的 html 代码。

在页面的 dom 范围中应用 angulajs 的功能，必须在指定范围应用 **ng-app** 指令，一般都是放到最外层的 html 标签（是一个 html 的自定义属性）

数据绑定：
用{{表达式}} ，表达式可以是变量，属性，带返回值的函数

ng-model 指令：双向绑定表单元素的数据

控制器 Contorller：在老版本中，可以有全局控制器，控制器可以不属于任何模块，新版本的控制器必须要基于某个模块。

创建 module 模块：angular.module("模块名",[依赖的其他模块名称]);

在 html 中通过 ng-app 指令设定使用的模块名称

创建控制器：模块对象.controller('模块名称',函数对象($scope));

$scope 代表当前控制器的作用域范围对象，在这里设置的数据模型，才能被视图使用

在视图中，使用 ng-controller 指令指定使用的控制器对象，尽量不要使用控制器嵌套（ng-controller），会使作用域混乱

根作用域，作用域范围包含了所有控制器，是全局数据模型，需要通过模块对象的 run 方法初始化全局的数据，$rootScope 代表全局作用域范围对象

依赖注入：

依赖：一个功能 A 调用了另个一个功能 B，就可以理解为 A 依赖了 B

注入：功能 A 直接获得了功能 B 的对象

在控制器中，经常需要注入一些对象，比如$scope,$http，但是这些对象的参数名称不能变化，为了防止代码压缩之后，参数名发生变化，ng 提供了两种注入对象的方式

把视图看做一个模板，模板中的动态数据由控制器提供，在视图中通过指令操作视图的数据

ng-repeat 指令：在视图中遍历集合或对象，循环生成 dom 结构，$index 返回当前下标

ng-disabled 指令：动态设置元素的 disabled 属性

ng-bind 指令：避免在视图绑定控制器数据之前，用户看到{{语法}}，通过一个额外 html 标签设置 ng-bind 来输出数据

ng-init 指令：初始化数据，给数据赋值

脏检查：angularjs 会对数据模型进行监控，当数据模型发生变化的时候，会及时更新视图的数据，这个过程就是脏检查。

一般来说，直接调用 ng 的指令，默认就会做脏检查，如果 ng 没有自动做脏检查，需要手动调用$apply 方法做脏检查。

$interval 和$timeout 对象是 ng 自带的定时器，默认就会触发脏检查。

通过$watch 方法检测数据模型的数据变化，触发回调函数，可以得到修改前的值和修改后的值，也可以强制给数据模型赋值

$watch 方法，如果检测的数据模型是一个对象类型，对象中的属性值变化默认不会触发回调函数，如果希望能触发，必须多加一个参数 true

ng-model-options 指令：设置模型数据变化的时机，设置{updateOn:'blur'} 表示只有失去焦点才会进行数据更新（变化）

ng-show 指令：动态显示元素

ng-hide 指令：动态隐藏元素

在 NG 中，使用服务作为 MVC 中的 M 层，创建服务的方式：

1. 利用$provide.provider 方法注册服务（语法不常用）
2. 利用$provide.factory 方法注册服务
3. 利用$provide.service 方法注册服务

factory 相对于 service 方法注册服务，区别在于 factory 可以返回任何类型，甚至是基本类型，而 service 只能返回对象类型

多个控制器共享相同的数据，有两种方案：

1. 嵌套控制器（不推荐，会使作用域的数据混乱）
2. 利用服务，让多个控制器注入同一个服务对象（推荐使用）

在模块中定义一些常量（只赋一次值，不会在程序运行中变化）

用 value 方法定义**常量**
用 constant 方法定义**常量**

config 方法：初始化模块的配置，定义 config 方法的两种方式：

方式 1：在创建 module 的第三个参数
方式 2：直接用模块对象.config 方法

config 只能注入 constant 常量，无法注入 value 定义的常量，config 中唯一能注入的服务对象就是 provider 定义的服务，其他的 factory 和 service 定义的服务无法注入

run 方法：程序启动以后，页面对用户可用之前执行，在 config 方法之后执行，可以注入 provider，factory，service 注册的服务，也可以注入 value,constant 设置的常量

angular.bootstrap 方法：和那个 css 框架 bootstrap 没有半毛钱关系，这是用来动态应用模块的语句，可以代替 ng-app 设置，不能重复应用模块。

过滤器 Filter：主要作用格式化数据和过滤数据，格式：表达式|过滤器名称
数字格式化过滤器：number
货币格式化过滤器：currency
日期格式化过滤器：date
大小写格式化过滤器：lowercase uppercase

过滤数据：

limitTo：过滤数组的前几位或后几位
filter：进行关键字模糊匹配过滤
orderBy：按照属性排序

可以在代码中使用内置的过滤器，但是需要注入$filter 服务对象

自定义过滤器：独立在控制器，服务之外的单独对象，用模块对象.filter 方法创建

ng-class 指令：动态设置应用样式类

angularjs 发出异步 ajax 请求（ng 没有同步请求），必须注入$http 服务，ng 的 ajax 请求默认都是 promise 风格的，请求都会返回一个 promise 对象，然后通过这个对象这调用成功和失败的回调函数。

PayLoad 请求和 FormData 请求的区别：

PayLoad 不会把数据按照键值对的形式发送给服务器，而是产生一个完整的固定字符串直接发送给服务器，需要服务器自己做解析。

AngualrJs 的 post 请求默认就是 payload 形式，如果想按照 FormData 发送请求，需要对模块做额外的配置

SPA 单页 Web 应用（single page web application，SPA），特点是只有一个主页，所有的二级页面都是通过 ajax 动态请求响应，设置到主页面的 DIV 中，需要注意的是要把所有二级页面的 css，javascript 提前在主画面加载。

页面路由：ajax+历史记录，进行动态页面切换

Angularjs 提供默认路由和第三方路由两种方式，自带的默认路由没有嵌套视图功能，一般我们使用都是第三方路由

ng-view 指令：指定某个容器为显示路由页面的容器

url 重写形式传递参数的两种方式：

1. url?参数名称=参数值&参数名称=参数值
2. /xxx/xxx/xxx　　rest 风格的 url

内置指令：

ng-bind-template：等价于 ng-bind，但是需要写{{xxx}}，可以写多个表达式

ng-init 指令：初始化数据，给数据赋值，在双层嵌套循环中非常有用

ng-repeat 指令：遍历一个对象或者数组，内置值 $index 遍历的下标 $first 是否第一个元素 $last 是否最后一个元素 $middle 是否最后一个元素

ng-cloak 指令：等价于 ng-bind 的功能，避免闪烁，但是还是可以使用{{}}绑定数据

ng-non-bindable 指令：转义{{}}语法变为普通文本

ng-bind-html 指令：不转义 html 语法，直接在浏览器执行，需要 angular-sanitize.js 插件支持和依赖 ngSanitize 模块

几乎所有的普通的 html 属性，都可以写成 ng 指令，例如 ng-style ng-href ng-src 等等，或者 ng-attr-xxxx 写法

大部分的 js 事件，都可以作为 ng 的指令，例如 ng-click ng-focus　ng-mouseover 等等，$event 代表当前事件对象

ng-show ng-hide 指令：显示和隐藏

ng-class 指令：应用样式类
ng-class-even：偶数应用样式
ng-class-odd：奇数应用样式

ng-if 指令：类似 if 语句，只有为 true 才执行，如果 false 就不执行

ng-switch 指令：类似 switch 语句，判断等值条件，执行内容

ng-copy 指令：当复制内容触发事件
ng-cut 指令：当剪切内容触发事件
ng-paste 指令：当粘贴内容触发事件

ng-include 指令：包含其他文件内容，需要基于服务器或者本地 ajax 测试

ng-open 指令：折叠指令，配合 details 标签使用

自定义指令：通过模块对象. directive 方法，开发人员可以自定义指令，自定义指令不要用 ng 开头，很容易和内置的指令搞混，自定义指令的主要作用就是编写一些可复用的组件。

在 AngularJS 内置了一个 jquery，称作 JQLite，能够实现最基本的 JQ 功能

自定义指令分为四种情况：
- **E 元素**
- C 样式
- M 注释
- **A 属性**

创建自定义指令的两种方式：
- 方式 1：模块的配置中，使用 $compileProvider 创建
- 方式 2：直接使用模块对象.directive 方法

指令在创建的时候，命名是驼峰写法 xxXxxXxx，使用的时候是匈牙利命名法 xxx-xxx-xx

指令的具体设置：
restrict：指令的类型，四种类型**E 元素**　C 样式　M 注释　**A 属性，**可以写一个到多个

template：模板内容

replace：如果为 true，模板内容直接替换掉指令本身，如果为 false，保留原始指令内容，模板内容必须有且只能有一个根标签

templateUrl：动态引入外部文件内容作为模板内容，但是模板内容必须有且只能有一个根标签

transclude：是否保留原始，ng-transclude 指令：配合 template 或者 templateUrl 使用，代表保留下来的原始内容

compile 方法：在指令执行的时候回调一次（尽量不要调用，否则 link 方法会失效）

link 方法：在指令执行的时候回调多次，一般我们在 link 方法里直接操作 dom

controller：属于指令自己的控制器

controllerAs：给控制器起个名字，在控制器中用 this 创建的方法，可以在视图中调用

require：依赖的其他指令，有三种依赖的形式

1. 直接写指令名，默认是在当前指令中查找
2. ^指令名，在父指令中找
3. ?指令名，表示找不到指令，创建指令本身不会抛异常

scope:控制器的独立作用域的设置，如果不设置，和父控制器共享作用域

= 和父作用域的数据做双向数据绑定

@和父作用域的数据做单向数据绑定

&把父作用域的数据封装为一个函数，通过函数的返回值获得数据

表单和表单验证：

表单元素包括表单本身一定要加上 name(不是 id)

表单验证相关的属性

xxx. $dirty 如果表单或者输入框有使用到则为 True

xxx. $pristine 如果表单或者输入框没有使用则为 True

xxx. $valid 这一项表单元素是否验证通过

xxx. $invalid 这一项表单元素是否验证未通过

xxx.$error 表单元素验证是否成功的数据模型

相关的表单验证的样式：

has-error：错误样式，表示验证失败

相关的验证规则：

required：必填

ng-minlength：最少几位字符

ng-maxlength：最多几位字符

ng-pattern：正则表达式验证

type=email 邮箱验证

type=url url 验证

type=number 数字验证（不怎么好用） min 最小值 max 最大值

实现其他的验证，需要自定义验证，写一个自定义指令

自定义表单验证，需要自定义一个指令，指令中要检测 view 到 model 和 model 到 view 的过程

$parsers 对象代表 view 传值给 model
$formatters 对象 model 传值给 view

如果非表单元素，例如 div，p 等需要做数据双向绑定，是不能直接使用 ng-model，需要通过自定义指令，手动的绑定数据。

Angularjs 内置的常用工具函数：
angular.uppercase()，angular.lowercase() 大小写转换
angular.isArray() 判断是否数组
angular.extend()把一个对象的成员加入到另一个对象
angular.equals() 判断两个对象的内容是否相等，==默认判断的是引用地址是否相等
angular.toJson() json 对象转为 json 字符串
angular.fromJson() json 字符串转为 json 对象
angular.copy() 克隆对象

其他的服务对象
$log 日志服务，专门用来操作日志
$location 地址栏服务，专门用来操作 url
$cacheFactory 缓存服务，把一些数据放到内存，其他地方可以访问
$sce 实现不转义 html

常用的 angularjs 的插件，例如 angular-animate.js 动画插件，angular-resource.js http 的 ajax 插件，angular-sanitize.js 转义 html 插件，angular-route.js 路由插件，例如第三方的 angular-ui-router.js 路由插件，ng-table 表格插件等

ng 的插件的 js 版本一定要和 ng 的版本保持一致，否则可能会出各种问题！

版本控制工具

常见版本控制工具：
VSS 微软出品 不允许同一时刻多人 check out（签出，检出）
cvs 和 vss 相比多了版本的概念，允许多人同时 checkout
svn 集成了 vss 和 cvs 的优点，允许多人 checkout，也可以不允许多人 checkout（加锁）
git 分布式版本控制系统，目前用的最多的是和 github 配合使用