目录

[说明 4](#_Toc519667072)

[前端 5](#_Toc519667073)

[一、学习体系 5](#_Toc519667074)

[1.HTML 5](#_Toc519667075)

[**语法** 5](#_Toc519667076)

[**HTML用法** 6](#_Toc519667077)

[**HTML常见标签** 6](#_Toc519667078)

[2.CSS 14](#_Toc519667079)

[内联样式 14](#_Toc519667080)

[内部样式 15](#_Toc519667081)

[外部样式 15](#_Toc519667082)

[层叠样式表的规则特性 15](#_Toc519667083)

[选择器 16](#_Toc519667084)

[css常用样式属性 18](#_Toc519667085)

[课堂练习-学子商城首页的中间部分-简易版 21](#_Toc519667086)

[控制元素的显示方式display 24](#_Toc519667087)

[定位 25](#_Toc519667088)

[3.JavaScript 27](#_Toc519667089)

[变量 27](#_Toc519667090)

[数据类型 27](#_Toc519667091)

[运算符 29](#_Toc519667092)

[流程控制语句 31](#_Toc519667093)

[内置对象 34](#_Toc519667094)

[extra 36](#_Toc519667095)

[DOM编程 43](#_Toc519667096)

[节点（Node） 44](#_Toc519667097)

[DOM Core 44](#_Toc519667098)

[HTML DOM 50](#_Toc519667099)

[extra 54](#_Toc519667100)

[BOM编程 54](#_Toc519667101)

[window对象 54](#_Toc519667102)

[Location对象 56](#_Toc519667103)

[Screen对象 57](#_Toc519667104)

[History对象 57](#_Toc519667105)

[Navigator对象 58](#_Toc519667106)

[JavaScript事件属性 58](#_Toc519667107)

[extra 60](#_Toc519667108)

[JSON 60](#_Toc519667109)

[案例（DOM） 61](#_Toc519667110)

[一、邮件全选 61](#_Toc519667111)

[二、轮播图 64](#_Toc519667112)

[三、列表二级联动 66](#_Toc519667113)

[四、页面遮罩 68](#_Toc519667114)

[五、换肤 69](#_Toc519667115)

[六、按钮倒计时 70](#_Toc519667116)

[extra 70](#_Toc519667117)

[4.jQuery-js框架 71](#_Toc519667118)

[jQuery介绍 71](#_Toc519667119)

[使用jQuery 71](#_Toc519667120)

[jQuery对象（重点） 71](#_Toc519667121)

[二、浏览器介绍 79](#_Toc519667122)

[sublime插件安装 80](#_Toc519667123)

# 说明

部分蓝色字体为举例

# 前端

[李然----g-liran@tedu.cn](mailto:李然----g-liran@tedu.cn)

## 一、学习体系

# 1.HTML

用来勾勒出网页的结构和内容-骨架

html保存在服务器，运行在浏览器

html--超文本标记语言

### **语法**

1. **标签**

<标签></标签>

ex:<div></div>

1. **属性**

<标签 属性1=’值1’ 属性2=’值2’ …></标签>

ex:<span id=’s1’ style=’..’></span>

1. **标签与标签之间的嵌套关系**

<div>

<span id=’s1’></span>

</div>

1. **标签分类**
   1. 闭合标签/双标签

<body>内容<body1>

* 1. 非闭合标签/单标签

<标签>或

<标签/>

<meta charset=’utf-8’>

\*\*单标签没有内容这个概念

总结：学习html就是学习固定标签，固定属性以及标签之间的嵌套关系

### **HTML用法**

html经历了5个版本

html4之前版本很宽泛，宽松

html4非常严谨

html5是最新也是最流行的版本，比html4宽泛

总结：doctype用于约束HTML文档内部的标签，属性和嵌套关系

不写doctype，同一篇代码会在不同浏览器中有巨大的显示差异



### **HTML常见标签**

1. **文本标签--跟论文格式很相似**

1.1标题 <hn></hn> n = 1~6

ex:<h1></h1>

<h2></h2>

…

<h6></h6>

总结：h1最大，h6最小

字体较粗

垂直方向，上下都有空间

1.2段落 <p></p>

垂直方向，上下都有空间

1. **课堂案例**

使用hn标签和p标签，写一份简易的简历

属性align=‘center’可以使用在标题标签中

1. **列表标签--使文本显得有条理**

3.1无序列表

u—unordered

l—list

i--item

<ul>

<li></li>

<li></li>

…

</ul>

3.2有序列表

<ol>

<li></li>

<li></li>

…

</ol>

1. **分区**

<div></div>

div看不见，功能是为了逻辑分区

<div id='header'></div>

<div id='nav'></div>

…

H5添加了很多和div一样，但是更容易读容易理解的标签

<header></header>

<nav></nav>

<footer></footer>

\*块级分区 div在这个区域独占一行

\*行级（行内）分区

<span></span>

如果只是一行文字内部的区域进行分区设置属性或者修改样式，此时使用行内分区

行内分区不会换行，会和其他文字混合在同一行上

1. **元素（标签）的显示方式**
   1. 行级元素（行内）

可以和其他行级元素位于同一行显示

span

* 1. 块级元素

自己独占一行

hn p li ul ol div

1. **其他行内元素**

i/em 斜体

b/strong 粗体

u 下划线

del/s 删除线

1. **实体引用，实体替换符号**

http://www.w3school.com.cn/html/html\_entities.asp

行内元素的空格折叠现象

写代码的时候，不管连写多少个空格和回车，都显示成一个空格

<br>或<br/>🡪回车

&nbsp；🡪空格

<>🡪&lt（<） &gt（>） &amp(&) &#39（'） &apos（”）

1. **图片标签**

单标签，最重要的属性是src

<img src=’图片路径’>

支持图片格式：png jpg gif

图片路径的写法

主机、路径、文件名 绝对路径

在一个页面中 显示其他网站中的图片，这种行为成为图片盗链，通常使用绝对路径，称提供网页图片的网站称为图床，节省了自己服务器的存储空间，坏处是当图床网站将图片路径改变，自己的网站就显示不出来了

<http://cdn.tmooc.cn/tmooc-web/imgad/2018/6/29//1A69E1CFF81D4D4AA90AAAC08720E75F.png>

相对路径🡪参照物🡪当前.html文件

img的其他属性：为行级元素，自带了width和height属性，通常只写一个尺寸属性，另外一个自适应

title：鼠标悬停在图片上，显示的文本

alt：找不到图片的时候显示的文本

1. **超链接a标签**

用于跳转页面

链接到网页、图片、下载

<a href="01\_img.html">跳转到本机Troye链接</a>

<a href="img/troye1.jpg">图片链接</a>

<a href="http://www.tmooc.c n/">跳转到网页</a>

a标签的属性

target：有两个取值：\_self,\_blank默认为\_self

\_self：链接页面在当前选项卡打开

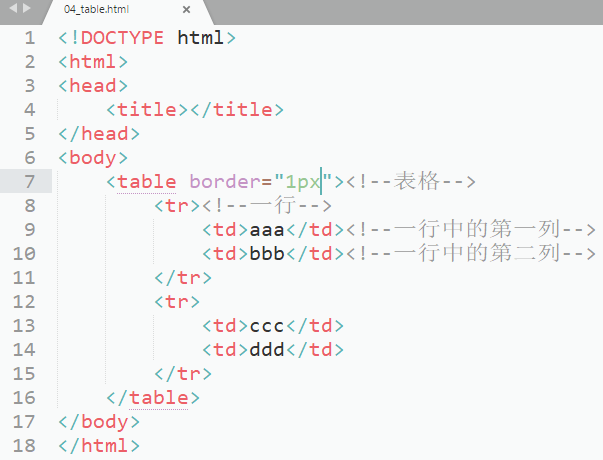
\_blank：链接页面在新的选项卡打开

\*\*两个页面之间的跳转，是超链接

一个页面内部的位置跳转，是锚点

H5锚点的用法：  
在需要跳转的位置标签内添加id属性，比如id=‘abc’，在a标签内的href属性中添加值href=‘abc’，这样点击a标签，页面就会自动跳转到锚点的位置，另：href=‘#’会让页面直接跳转到顶部

1. **网页中的表格标签**





table的常用属性

width：宽度

height：高度

border：边框宽度

algin：表格对齐页面

cellpadding：边框与内容之间的距离

cellspacing：边框之间的距离

colspan：跨列，一个单元格占据多列

<td colspan=’2’></dt>从左往右跨两列

rowspan：跨行，一行单元格占据多行

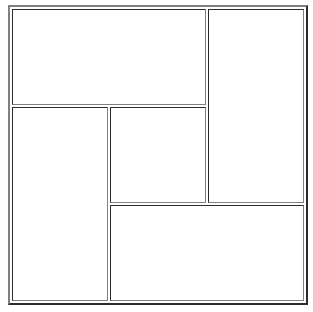
<td rowspan=’2’></dt>从上往下占2行，由于占据了下面单元格的位置，下面只用写n-1列，都减少一个td

1. 表分组

<thead></thead>

<tbody></tbody>

<tfoot></tfoot>

🡪

1. **表单标签**

用于在页面中显示一个可以收集用户填写的数据的控件

总结：想通过网页吧用户填入的信息发送给服务器处理

* 1. 所有的标签必须在form表单中放置
  2. form表单中的标签必须有name属性才可以传
  3. 使用submit将数据提交给服务器

form表单的主要属性

action：表单提交数据的url

method：表单提交数据的方法（get，post）

enctype：表单数据的编码格式

**1.文本框和密码框**

<input type=’text’ name=’…’> 文本框

<input type=’password’ name=’…’> 密码框

value属性：值

placeholder：输入文本提示

maxlength：限制输入的最大字符数

redonly：只读

lable标签：与span类似，但只用在表单中

**2.radio：单选框**

<input type="radio" name="gender">

<input type="radio" name="gender">

\*\*name属性决定一组的单选

\*\*单选框只有有value属性才能传值

\*\*checked=”checked”属性表示默认选中的单选框

<input type="radio" name="gender" value="男" checked="checked"><label>男</label>



**3.checkbox：多选框**

\*\*name属性，决定了一组多选

\*\*有value属性才可以提交值

\*\*checked=”checked”默认选中



**4.submit 按钮：提交**

<input type="submit" value="提交">

**5.button按钮**

<input type="button" value="click" onclick="alert('fdadg')">

**6.重置按钮：reset，将表单内的内容清空重置**

<input type="reset" name="重置">

**7.隐藏域**

<input type="hidden" name="id" value="123456789">

在form表单中存在，但是对用户隐藏，包含着不给用户看的信息，可以提交给服务器

**8.文件选择器-上传文件用**

<input type="file" name="up" value="选择文件">

属性multiple：多重的，可以选择多个文件，不写则只能选择一个

**9.文本域**

<textarea>大量文本</textarea>

rows cols描述的是文本区域内可见的宽度和高度，原意为行数和每一行显示的字符数，但由于系统编码不同，浏览器解析也不同，导致显示字符数差异很大

**10.下拉列表**

<select>

<option></option>

<option></option>

……

</select>

总结：

1.当option没有value属性时，select标签的value为选中的option的内容

2.当option有value属性时，select标签的value为选中的option的value值

**表单标签总结：**

1.只有含有name属性的表单标签才可以提交数据

2.radio和checkbox必须设置value属性才可以提交数据

3.text的value是直接设置值，placeholder是设置提示

4.radio和check默认选中为checked

5.select默认选中为selected

# 2.CSS

用来美化网页-皮肤-化妆

### 内联样式

<span style="color:#f00;font-size:30px">内联样式的使用</span

在标签的style属性中写样式

特点

内联样式使用不多

1.不能重用

2.内联样式优先级最高

项目代码完全不使用内联

只有在学习和调试过程中

### 内部样式

在head标签中，写style标签

在style标签内部写样式

选择器{

属性：值;

}

样式代码可以重用,但只能在本html中重用

在项目使用不多

### 外部样式

创建一个.css文件

在html文件中，head标签中引用这个css文件

<link href="03.css" rel="stylesheet">

### 层叠样式表的规则特性

1.层叠效果

多个样式作用在同一个标签上

如果样式属性不重复

那么这些样式都会产生效果

2.继承

父级元素的样式会继承给子元素

子元素可以重新写样式，

不使用父级元素的样式

3.优先级

优先级最低的 浏览器默认样式

优先级最高的 内联样式

内部样式和外部样式--就近优先

以被影响标签为基准，哪个样式离这个标签近

就使用哪个样式的效果

### 选择器

选择器的作用，为了选出当前页面中符合要求的一个或者多个标签

样式属性的作用是给选出来的标签设置样式

选择器{样式属性：值;}

* 1. 通用选择器

\*{}：尽量少用，写通用样式的时候使用

* 1. 标签选择器

选出当前页面中标签名称符合要求的标签

ex：div{} span{} P{}

使用范围；设置某个标签的通用样式

* 1. id选择器

唯一不重复

#p2{

border: 2px solid #b00;

}

id选择器 #id值

同一个html中，id值是惟一的，不可重复

同一个html中使用一个id选择器，只能找到一个标签

* 1. 类选择器（class选择器）

如果编写html标签时，需要让某些标签应用一类样式，就使用类选择器，标签的属性class为类名

<div class="d1"></div>

.d1{

width: 200px;

height:100px;

background-color: #006699;

border: 1px solid #f00;

}

多类选择器：当一个标签想匹配多个样式的时候，使用多类选择器

<div class="w200 h50 bg-c"></div>

* 1. 分组选择器

把一系列选择器当成一组，设置共有样式

div,p,h1,ul,li{

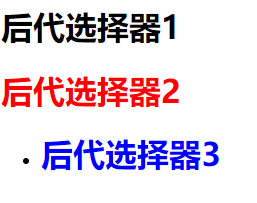
background-color: #f00;

}

逗号分隔，可以使用标签选择器，id选择器，class选择器混合编写，一般用于设置公共样式

一般项目中分组选择器写在最开始的部分

* 1. 后代选择器（派生选择器）



div内部只要存在h1，就可以定位到h1

* 1. 子代选择器

div>h1{

background-color: pink;

}

h1必须是div的儿子才可以匹配

* 1. 伪类选择器

标签名：hover{}：设置鼠标悬停时的样式

### css常用样式属性

#### 布局 相关属性

1.width：宽

2.height：高

单位：px cm 100%

3.margin：设置外边距

margin：10px，同时设置四方外边距

margin；10px 20px，上下为10，左右为20

margin；0px auto，块级元素水平居中

margin：10px 20px 30px 40px；上、右、下、左

margin-top：10px；上边距

margin-bottom：10px；下边距

margin-left：10px；左边距

margin-right：10px；右边距

4.padding：10px；设置4个方向的内边距

padding：10px 20px；上下为10，左右为20

padding：10px 20px 30px 40px；上、右、下、左

padding-top：10px；上内边距

padding-bottom：10px；下内边距

padding-left：10px；左内边距

padding-right：10px；右内边距-

改变内边距，视觉效果，元素大小改变

\*\*\*块级元素和行级元素的区别\*\*\*

块级元素可以设置宽高

行级元素，除了自带宽高属性的标签，其他元素宽高设置无效

margin-top，margin-bottom对行级标签无效

3.盒子模型（框结构）

css渲染页面时，使用盒子模型

对元素之间的留白，元素背景填充范围，元素大小的控制规则，就是盒子模

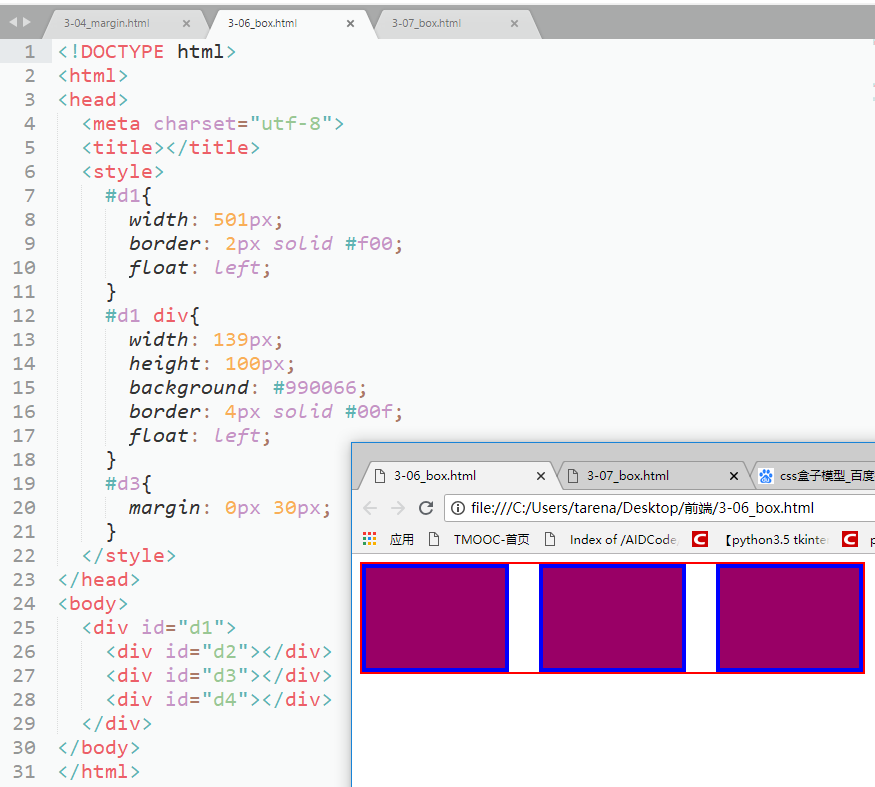
型

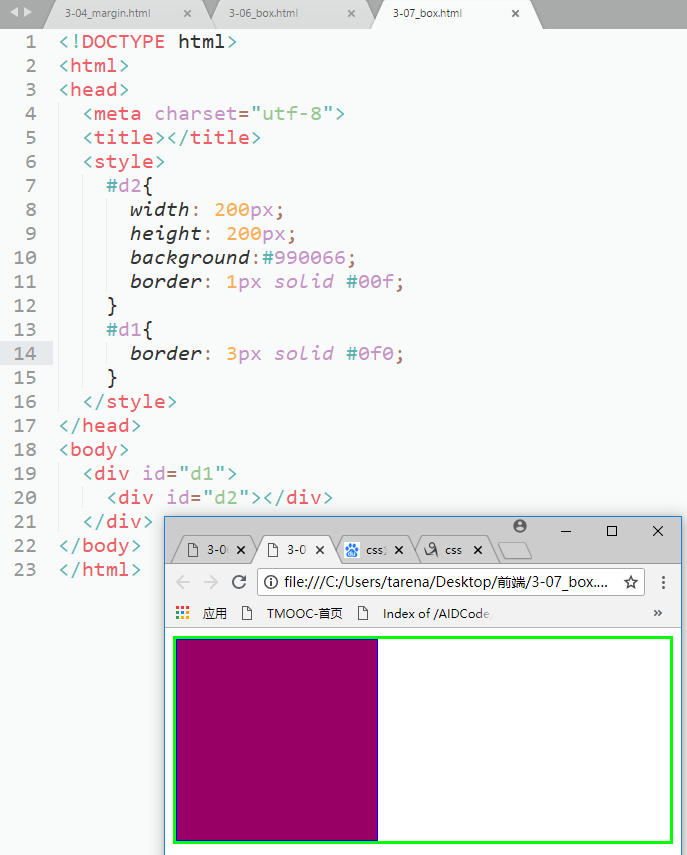
主要关注宽度

左外边距+左边框+左内边距+内容区域+右内边距+右边框+右外边距

总结：div默认占据的宽度是父容器的内容区域100%

div默认的高，是内部内容的高，如果内部没有内容，或没有高，则高度为0





#### 背景属性

1.background-color：合法的颜色值

background：合法的颜色值

合法的颜色值：

1.颜色的单词：red、pink…

2.#ffffff：6位16进制字符串

3.#aabbcc—>#abc

4.rgb（255,255,255）；

5.rgba（255,255,255,0~1）a为不透明度

2.背景图片

background-image：url（”图片路径”）

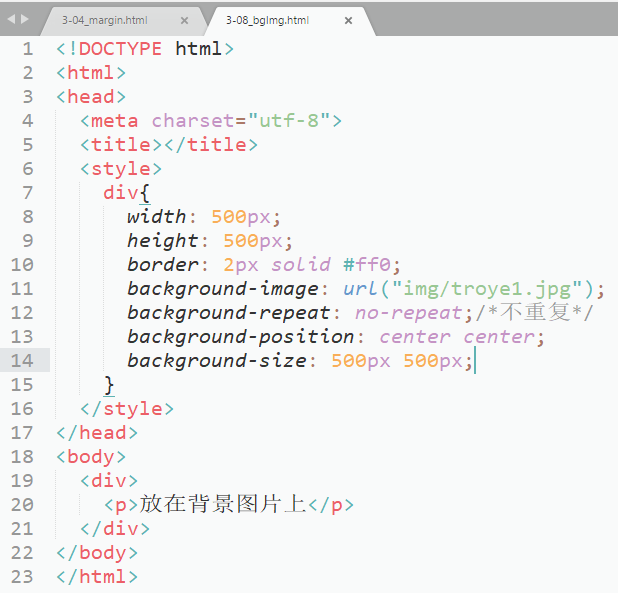
background-repeat：平铺方式

no-repeat repeat repeat-x repeat-y

不重复 默认，重复 水平平铺 垂直平铺

background-position: center center;（left，right，top，bottom可选，也可直接写数值，如100px）

background-size: 500px 500px;设置背景图片的大小



### 课堂练习-学子商城首页的中间部分-简易版

1、计算标签宽高

2、margin+padding

写html和css的节奏

* + - 1. 确认html代码的结构
      2. 通过盒子模型，计算标签所占的位置
      3. 通过定为和浮动确定元素位置
      4. 写css代码
         1. 宽高
         2. 背景
         3. 文本样式、对齐
         4. 微调

#### 文字属性

1. font-size

单位：px cm em

1. font-family

font-family：“微软雅黑”、“黑体”、“文泉驿正黑”

1. 字体粗细

font-weight：lighter（细）、normal（正常）、bold（粗体）、bolder（大粗体）

1. font-style：字体样式

font-style：italic（斜体）、normal

#### 文本属性

1. color：文本颜色
2. 文本的对齐方式：

text-align：left center right

1. 行高属性

line-height：

div中文本默认情况下会垂直处于行高中间的位置，如果希望文本在div中垂直居中，只需要把行高属性与div的高度设置一致

\*\*\*但如果字体过多，折行之后行高效果会在每一行生效

1. 文本装饰

text-decoration：none underline（下划线） line-through（删除线） overline（上划线）

#### 边框属性border

border：宽度 样式 颜色

border-width:

border-color:

border-style:solid（实线） dotted（点） double（双实线） dashed（虚线）

border-top: 2px solid #f00;

border-top-color：

border-top-width：

border-top-style：

border-left同上top

border-bottom同上top

border-right同上top

border:none;去掉边框，文本框也可

边框的圆角

border-radius：3px；（圆角半径）

#### 常见复杂属性

1. 溢出overflow

overflow：visiable（溢出属性可见）

hidden（不可见）

scroll（溢出部分变成滚动条）

div的bug：当两层div嵌套，内层div与外层div的内容区域的上线齐平（无边框的情况下），内层div如果有margin-top属性，外层div的margin-top会跟随内层的margin-top值显示

解决方案：需在外层div中加上overflow：hidden样式



### 控制元素的显示方式display

display：block；（让元素按照块级元素的规则显示）

inline；（让元素按照行级元素规则显示（一般不用））

inline-block；（行内块）

none；隐藏标签

总结：

1.行级元素特点：可以与其他行级元素共占一行，不能改变元素的height和width值（自带宽高属性的除外，如img）；行级元素的大小由内容撑开；可以使用padding、margin-left和margi-right，margin-top和margin-bottom不起作用

2.块级元素特点：独占一行；若不设置width，默认充满父级元素的宽度，若不设置height，靠内容撑开；可以正常地设置宽高；可以设置margin和padding的各个属性

3.行内块特点（不建议初学时使用）：与其他的行级元素和其他的行内块共用一行；可以设置宽高；可以设置margin和padding的所有属性；

使用行内块需要注意很多细节

1.每一个行内块都需要设置宽度

2.如果html代码中有空格，行内块会产生消除不了的空隙

### 定位

1. 默认定位

文档流定位🡪默认文档流

1. 浮动

float：left；元素浮动起来，往左放

right；元素浮动起来，往右放

clear: both;当前元素不会被任何浮动元素遮挡

总结

1.一旦元素设置了浮动，当前元素将脱离默认文档流，在默认文档流之上进行渲染，原来的位置将被其他元素占用

2.元素只会在当前行浮动（注意判断当前行的位置）

3.浮动最重要的作用就是块级元素横向显示

1. 高度坍塌

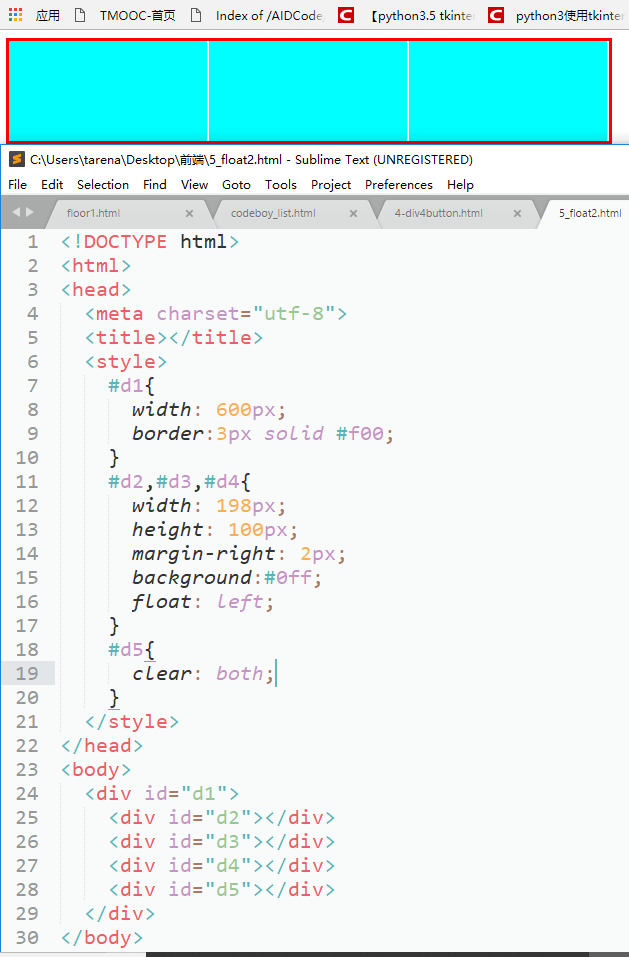
1.原理：外部div没写高度，靠内部高度撑起，而内部div都浮动了，脱离默认文档流，外部div内部认为内部没有元素了，高度为0

2.解决方案

1.给外部div加上高度，但需要提前知道高度是多少

2.给 外部div加浮动属性，但会影响外部的结构

3.在内部浮动的div下面加一个div，样式全无，仅加一条clear：both；（原理：没有内容，所以高为0，没有写宽高，所以用户不可见，所以宽为父元素的宽度，且有clear：both；样式，那么这个div就一直在浮动元素的下方显示，则父元素的高度自动匹配这个div的位置高度）



1. css中定位

position：relative；相对定位

absolute：绝对定位

1.relative：相对定位

1.定位之后，原来的位置不被占用

2.top/right/bottom/left为距离原来位置的距离

3.如果不写top/right/bottom/left，效果相当于没有定位

2.absolute：绝对定位

1. 定位之后，原来的位置会被占用

2.如果子元素有position：absolute，会向上级元素查找有position的元素，如果有一级元素带有position属性，就以这个元素的左上角为参照定位，如果没有，就以body左上角定位

3.一般情况下，absolute上级的标签会使用relative来定位，很少使用absolute

3.fixed；

# 3.JavaScript

吴华：wuhua@tedu.cn

让网页呈现出动态的数据和效果

是一种运行在客户端的解释性脚本语言

编译型：将程序的源代码编译成二进制机器语言，形成编译文件（.dll或.exe），以后再执行时可以直接调用变异的结果，优点是一次编译，多次执行

解释型：在每次执行时都需要进行编译、执行的过程，而且需要编译器进行编译、执行的操作

解释型脚本是程序执行的一种方式

说明：

JS严格区分大小写

每行JS代码必须以分号结尾

文件扩展名为js

1.创建JS文件并写入相关代码

2.在网页文件内调用相关的js文件

CSS文件合并js文件合并

1.为了减少http请求次数，减轻服务器的请求压力

2.前端框架（如天猫的Kissy）可以实现合并

### 变量

指存储在计算机内存中有名称的信息，其值在程序运行期间可以发生变化

声明变量：var 变量名称；或 var 变量名称=值；

### 数据类型

1. 字符型（string）
2. 数值型（number），包括整型和浮点类型，可以带有符号位，可以存储八进制、十进制、十六进制的数值，八进制的表示方法：0[0-7]+，十六进制的表示方法：0[x|X][0-9a-fA-F]+
3. 布尔类型（Boolean），只能存储true或false

\*\*\*在控制台输出结果console.log(typeof(age))

1. 未定义（undefined），值也只为undefined

\*\*\*以下变量的值为undefined

* 1. 声明但没有赋值的变量 （var n；console.log（typeof（n）））
  2. 声明但赋值为undefined的变量（var n=undefined；console.log（typeof（n）））、
  3. 引用对象不存在的属性时（var obj={}；console.log（typeof（obj.username）））

1. 对象（object），属性与方法的集合体

{

method：function（[parameter,…]）{

……

}

}



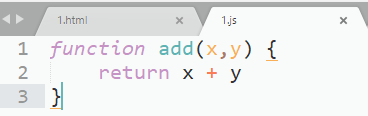
1. null（空对象）,其作用是为了断开继承链；类型为对象，值为null

var n=null;

console.log(typeof(n));---->object

console.log(n); ---->null

1. function（自定义函数），可以重复执行的代码段；



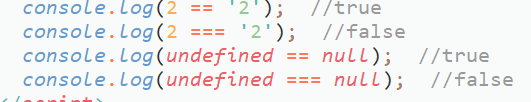


### 运算符

按类型分

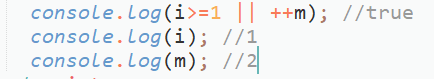
1. 字符运算符：+
2. 数字运算符：+（正号）、-（符号）、\* / % + -
3. 比较运算符：> >= < <= !=（不等于） ==（等于） ===（全等） !==

全等（===）：值相等、数据类型相同



1. 逻辑运算符：! && ||

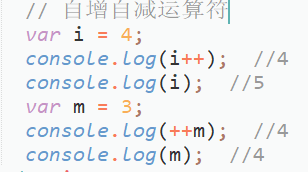
短路现象：在进行&&或||运算时，只要第一个表达式的值可以确定整个表达式结果时，后续表达式将忽略执行；



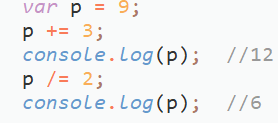
1. 自增/自减运算符：

i++ i--，后缀形式：先使用，后加减

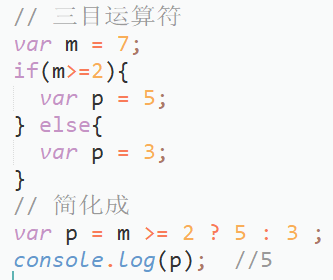
++i --i，前缀形式：先加减，后使用



1. 赋值运算符：= += -= \*= /= %=



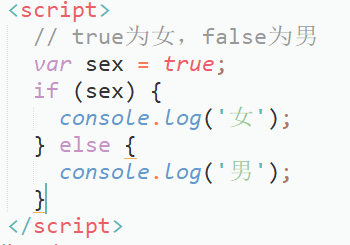
按参加运算的操作数的数量

1. 单目运算符：只有一个操作数，如负数（-），逻辑非（!）
2. 双目运算符：有两个操作数
3. 三目运算符：expression ? value1 : value2

运算符优先级：一二三赋值，

算数比较逻辑

### 流程控制语句

1. if语句

1.if（condition1）{

….条件1成立

}else{

…条件1不成立

}

2.if else语句

if（condition1）{

….条件1成立

}else if（condition2）{

…条件1不成立

}else{

…条件2不成立

}

1. switch语句

switch（expression）{

case 值1：

…

break；

case 值2：

…

break；

default

…

break

}

说明：

* 1. if语句的条件表达式只能是布尔型（可能会存在数据类型的自动转换）

switch语句的表达式可以为任意数据类型（布尔除外）

* 1. if语句每次执行时都将对变量进行取值操作，而switch只进行一次取值操作
  2. switch语句进行的是全等（===）运算

1. for语句

for（expr1；expr2；expr3）{

……

}

说明：

expr1，在循环之前无条件执行一次，一般用于变量的初始化

expr2，在循环前执行一次，以判断能否进入循环体；

expr3，在循环后执行一次，一般用于步长的计算；

for语句的表达式可以省略任意一个或全部；

1. while语句

while（条件表达式）{

……

}；

说明

* 1. 死循环：指条件永远为真的循环语句
  2. for语句和while语句同属于“当型循环（先判断，后执行）”，所以可以互换；

1. do…while循环

do{

…

…

} while（条件表达式）；

说明

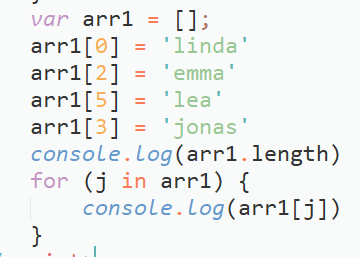
* 1. do…while语句为先执行后判断；
  2. do…while语句至少执行一次，而当型循环（for、while）至少执行零次；

1. for…in…语句

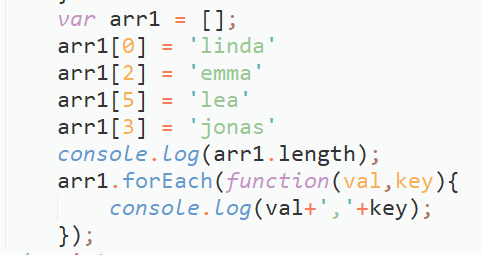
for （key in array）{

……

}



1. forEach（）语句



1. break语句：跳出本次的switch及循环语句
2. continue语句：跳过当前的循环，进行下一次循环

### 内置对象

1. String对象
   1. 构建String对象
      1. 直接量方式：’string’或”string”
      2. 构造函数：new String（'string’）
   2. 属性

length

描述：返回字符串的长度

语法：int ,object.length

* 1. 方法
     1. toLowerCase（）

描述：将字符转换为小写字母

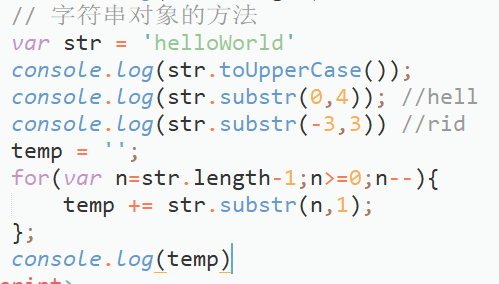
语法：str.toLowerCase（）

* + 1. toUpperCase（）

描述：将字符转换为大写字母

语法：string object.toUpperCase（）

* + 1. substr（）

-描述：截取字符串，取length位

语法：string object.substr（start[,length]）

说明：

1.字符串从0开始编号

2.如果省略length参数，则返回start到字符串结尾之间的字符

3.如果start参数为负数，则起始值从结尾倒数

* + 1. substring

描述：截取字符串

语法：string object.substring（int start[, int end]）

说明：

1.返回的字符串包含起始位，不包含结束位

2.如果省略end参数 ，则返回到字符串结尾之间的字符

* + 1. indexOf

描述：返回字符串第一次出现的位置，如果没有则返回-1

用法：string object IndexOf（string）

console.log(str.indexOf('a')); //1

console.log(str.indexOf('k')); //-1

* + 1. lastIndexOf

描述：返回字符串最后一次出现的位置，如果没有则返回-1

string object lastIndexOf（string）



* + 1. split（）

描述：使用分隔符拆分字符串

语法：

array object.split（delimiter）

array object.split（RegExp）

## extra

UUID:通用唯一识别码，其形态是8-4-4-4-12，其作用是为保证文件名的唯一性

webp格式是Google推出的一种图像格式，其仅被Google Chrome支持，其特点是系相比jpeg文件来说，字节数更小；

京东商品展示，采用的是动态图片缩放的奇数

火狐MDN里有很多前端知识

思维导图软件：xmind

* + 1. replace（）

描述：字符串替换

语法：

string object.replace（string search，string replcement）

\*\*仅替换第一次出现的字符

string object.replace（string RegExg，string replcement）

* 1. ES6新增方法
     1. trim（）

描述：删除第一个之前及最后一个字符之后的空白（去掉两边空白字符）

语法：string object.trim（void）

* + 1. startsWith（）

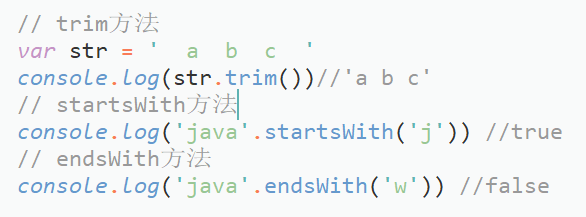
描述：检测字符是否以指定的字符开头

语法：bool object.startsWith（string）

* + 1. endsWith（）

描述：检测字符是否以指定的字符结尾

语法：bool object.endsWith（string）



1. Math对象

Math对象是一个静态对象，

* 1. 属性

Math.PI：圆周率

Math.SQRT2：2的平方根

* 1. 方法
     1. Math.ceil（）
     2. Math.floor（）
     3. Math.pow（）
     4. Math.sqrt（）
     5. Math.min（）
     6. Math.max（）
     7. Math.random（）产生0~1之间的随机数

Math.random（void）；

* + 1. Math.round（）四舍五入，仅保留到整数位

1. Date对象
   1. 构建日期对象

new Date（）

* 1. 方法
     1. getYear（）

描述：获取年份（从公元1900年至今的年份）

语法：int object.getYear（）

* + 1. getFullYear（）

描述：获取年份

语法：int object.getFullYear（）

* + 1. getMonth（）

描述：获取月份（取值范围为0-11）

语法：int object.getMonth（）

* + 1. getDate（）

描述：获取日期（xx号）

语法：int object.getDate（）

* + 1. getDay（）

描述：获取星期的第几天（0为星期天）

语法：int object.getDay（）

* + 1. getHours（）
    2. getMinutes（）
    3. getSeconds（）
    4. getMilliseconds（）获取毫秒
    5. getTime（）返回从公元1970年到现在的毫秒数（时间戳）



* + 1. toUTCString（）/toGMTString（）

描述：将日期转换成UTC/GMT格式

语法：object.toUTCString（）/ object.toGMTString（）



* 1. ES6新增的方法
     1. Date.now（）静态方法

描述：返回从公元1970年至今的毫秒数

语法：int Date.now（）

说明：该方法为静态方法

1. Array对象
   1. 构建Array对象
      1. 直接量方法

new Array[value1,…]

* + 1. 构造函数

new Array（length）

new Array（value1,…）

* 1. 属性
     1. length

描述：获取数组成员的数量

语法：int object.length

* 1. 方法
     1. unshift（）

描述：在数组的开头添加一个或多个成员，并返回数组的新长度

语法：int object.unshift（）

* + 1. push（）

描述：在数组的末尾添加一个或多个成员，并返回数组的新长度

语法：int object.push（）

* + 1. shift（）

描述：在数组的开头添加一个或多个成员，并返回数组的新长度

语法：mixed object.shift（）

* + 1. pop（）

描述：在数组的开头添加一个或多个成员，并返回数组的新长度

语法：mixed object.shift（）

* + 1. join（）

描述：将数组成员使用指定分隔符连接成字符串

语法：string object.join（delimiter分隔符）

* + 1. isArray（）

描述：判断是否为数组，静态方法

语法：bool Array.isArray（obj）



* + 1. map（）

描述：数组成员一次调用回调函数，并且返回新数组

语法：array object.map（function（value[,index]）{

…

}）



1. Regular Expression对象（正则表达式对象）
   1. 构建正则表达式对象
      1. 直接量方式

/正则表达式/修正符

* + 1. 构造函数

new RegExp（”正则表达式”[，”修正符”]）

* 1. 正则表达式
     1. 元字符
        1. \d,匹配数字，等价于[0-9]
        2. \D,匹配非数字，等价于[^0-9]
        3. \s,匹配空白字符，等价于[\n\r\t\v\f]
        4. \S匹配非空白字符，等价于[^\n\r\t\v\f]
        5. \w,匹配任意的数字字母下划线，等价于[0-9a-zA-Z\_]
        6. \W,匹配任意的非数字字母下划线，等价于[^0-9a-zA-Z\_]
        7. . ，匹配除换行符外的任意字符
     2. 量词
        1. ?，出现零次或一次，等价于{0,1}
        2. \*，至少出现零次，等价于{0，}
        3. +，至少出现1此，等价于{1，}
        4. {m}，出现m次
        5. {m,}，至少出现m次
        6. {m,n}，至少出现m次，至多出现n次
     3. 其他
        1. |，或者
        2. ^，以指定的字符开头
        3. $，以指定的字符结尾
        4. []，表示范围，如[0-9]
        5. [^]，表示不在范围内
     4. 转义符
        1. \n,\r,\t,\b,\f,\\
        2. \.
        3. \\*
        4. \?
        5. \|
     5. 捕获组（…）与非捕获组（?:…）

* + 1. 修正符
       1. g（global），全局
       2. i（ignore），忽略大小写



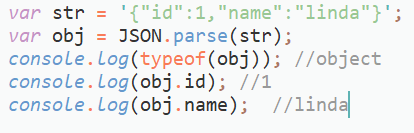
* 1. 方法
     1. test（）

描述：检测字符串是否符合正则表达式规范

语法：bool object.test（）



1. JSON对象
   1. string JSON.
   2. ad



## DOM编程

JavaScript由ECMAScript（简称ES）、DOM和BOM组成；

ECMAScript：关键字、保留关键字、流程语句、数据类型、内置对象；

DOM：提供HTML/XML文档的相关API；

DOM由DOM Core、HTML DOM和XML DOM组成；

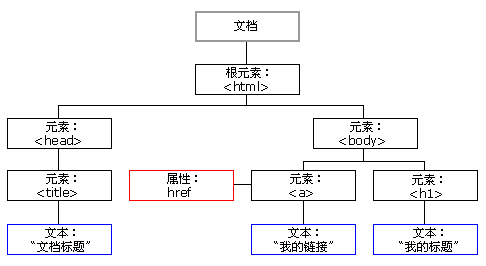
DOM Core：提供处理HTML和XML文档的API；

XML DOM ：提供处理XML文档的API

html DOM：提供处理HTML文档的API

### 节点（Node）

在DOM树中一切皆节点



1. 节点类型
   1. 1，代表元素节点（HTML标记）
   2. 2，代表属性节点（HTML标记的属性）
   3. 3，代表文本节点（纯文本）
   4. 8，代表注释节点（HTML注释）
   5. 9，代表文档节点（整个HTML文档）
2. 节点关系
   1. 子节点：某节点是另一个节点的直接下一级节点；如<a><b>…</b></a>中，a为b的父节点，b为a的子节点；
   2. 父节点
   3. 兄弟节点：同一个父节点的所有子节点互为兄弟节点

### DOM Core

1. 节点的属性
   1. nodeName

描述：获取节点的名称

语法：string node.nodeName

* 1. nodeValue

描述：获取节点值

语法：string node.nodeValue

* 1. nodeType

描述：获取节点类型

语法：int node.nodeType

* 1. parentNode

描述：返回节点的父节点

语法：node node.parentNode



* 1. childNodes

描述：返回节点内所有的子节点形成的集合(数组，包含文本节点)

语法：NodeList node.childNodes

* 1. children

描述：返回节点内所有的元素子节点形成的集合(数组，不包含文本节点)

语法：NodeList node.children

* 1. firstChild

描述：返回指定节点的第一个子节点

语法：node node.firstChild

* 1. lastChild

描述：返回指定节点的最后一个子节点

语法：node node.lastChild

* 1. nextSibling

描述：返回当前节点的下一个兄弟节点（可能为文本节点）

语法：node node.nextSibling



* 1. nextElementSibling

描述：返回当前节点的下一个元素兄弟节点（不包含文本节点）

语法：node node.nextElementSibling

* 1. previousSibling

描述：返回当前节点的上一个兄弟节点（可能含文本节点）

语法：node node. previousSibling

* 1. previousElementSibling

描述：返回当前节点的上一个元素兄弟节点（不包含文本节点）

语法：node node. previousElementSibling

1. nodeValue（）

描述：

语法：



1. 节点方法
   1. appendChild（）

描述：追加子节点

语法：Node（新插入的子节点） node.appendChild（Node）

* 1. removeChild（）

描述：删除子节点

语法：Node node.removeChild（Node）



* 1. replaceChild

描述：节点替换

语法：node node.replaceChild（newChild，oldChild）



#### document对象（继承自Node）

1. 属性

documentElement

描述：返回文档的根元素

语法：Element document.documentElement

1. 方法
   * 1. getELementById（）

描述：根据对象ID来获取元素

语法：Element document.getELementById（string id）

* + 1. createElement（）

描述：创建元素节点

语法：Element document.createElement（string tagName）

* + 1. getElementByTagName（）

描述：获取指定元素中包含指定标记名称所形成的集合（数组）

语法：NodeList Element.getElemenByTagName（string TagName）（单个对象内）

NodeList document.getElemenByTagName（string TagName）（整个文档）

* + 1. createTextNode（）

描述：创建文本节点

语法：TextNode document.createTextNode（string）



#### Element对象（继承自Node）

1. 方法
2. getAttribute（）

描述：获取属性值

语法：string Element.getAttribute（string name）

1. setAttribute（）

描述：设置属性

语法：void Element.setAttribute（string name，string value）

1. removeAttribute（）

描述：删除属性

语法：void Element.removeAttribute（string name）

1. hasAttribute（）

描述：返回是否存在属性

语法：Boolean Element.hasAttribute（string name）

1. getElementByTagName（）

描述：获取文档中指定标记名称所形成的集合（数组）

语法：NodeList document.getElemenByTagName（string TagName）（整个文档）

NodeList Element.getElemenByTagName（string TagName）（单个对象内）



### HTML DOM

HTML DOM：提供处理HTML文档的API

1. 选取对象

Element doucment.getElementById（string id）

NodeList doucment.getElementsTagName（string TagName）

NodeList Element.getElementsTagName（string TagName）

NodeList document.getElementsByName（string name）

NodeList document.querySelectorAll（string selector）

Element document.querySelector（string selector）

NodeList Element.querySelectorAll（string selector）

Element Element.querySelector（string selector）

1. 属性控制
   1. 对于单个单词的HTML标记属性即HTML DOM对象属性（class属性除外，因为为关键字）

pEle = document.getElementById（’p’）;

pEle.align = ‘center’;

* 1. 对于合成词的HTML标记属性（如table标记的cellspacing等）在HTML DOM编程时，采用“小驼峰标记法”命名（如cellSpacing）

tableEle = document.getElementById('table');

tableEle.cellPadding = '10px';

* 1. 对于HTML标记的class属性，在HTML DOM编程时使用className取代

var pEle = document.getElementById('p')

pEle.className = 'f1';

* 1. 对于HTML标记的布尔属性（如单/复选框的checked，列表项的selected属性）在HTML DOM编程时采用boolean类型表示

checkEle = document.getElementById('check');

checkEle.checked = true;

checkEle.disabled = true;

* 1. 在HTML DOM编程时，某些HTML DOM对象有自己特有的属性或方法（如form对象中的onsubmit和reset方法，input对象的focus和blur方法，下拉列表（select）的onchange方法和value属性）；

form：onsubmit reset

input：focus blur

select：onchange value

option：创建option对象：new option（string text[,int value]）

length属性：可以设置或获取

* 1. HTML标记的style属性在HTML DOM编程时形成CSSStyleDeclaration/CSS2Properaties对象

控制HTMLDOM对象的行内样式的语法是：object.style.CSS属性名称

* + 1. 如果CSS样式属性为单个单词，则直接写
    2. 如果CSS样式属性为合成词，则先将短横线去掉后采用“小驼峰标记法”



[点击查看CSS样式属性写法](JS相关资料图片/HTML的CSSStyle对象名称写法.PNG)

1. HTMLDOM方法
   1. getElementByName（）方法

描述：获取文档中name属性相同的对象的组成的集合（数组）

语法：NodeList document.getElementsByName（string name）

1. 控制内容
   1. innerText：不能解析HTML标记
   2. innerHTML：可以解析HTML标记
2. 元素的增删改：在HTMLDOM编程时除了极少数对象有自己的创建/删除方法外，其他的都需要使用DOMCore方法进行增加/删除操作
   1. select/option对象：有增删改

var opEle = new Option(proList[province][i])

cityEle.add(opEle);

* 1. image对象：可创建，但需用DOM Core append方法进行添加

var imageEle = new Image（）；（创建图像对象）

imageEle.src = ‘……jpg’；（设置属性）

document.body.appendChild（imageEle）（添加到DOM树中）

* 1. table对象

#### Document对象

1. 属性
   1. title

描述：获取/设置标题栏信息

语法：document.title = value;或

var 变量名称 = document.title；

* 1. body

描述：获取文档的body对象

语法：Element document.body

* 1. images

描述：返回文档中所有的图像组成的集合（数组）

语法：NodeList document.images;

* 1. forms

描述：返回文档中所有的表单组成的集合（数组）

语法：NodeList document.forms;

* 1. links

描述：返回文档中所有的<link>组成的集合（数组）

语法：NodeList document.links;

* 1. anchors

描述：返回文档中所有的<a>组成的集合（数组）

语法：NodeList document.anchors;

1. 方法
   1. querySelectorAll（）

描述：返回由使用指定的CSS选择器的对象组成的集合（数组）

语法：NodeList document.querySelectorAll（selector）

说明：该方法支持所有CSS3选择器

* 1. querySelector（）

描述：返回由使用指定的CSS选择器的对象的第一个元素

语法：Element document.querySelector（selector）

#### Element对象

1. 属性
   1. tagName

描述：获取元素的标记名称

语法：string Element.tagName

1. 方法
   1. qureySelector（）

描述：返回由使用指定的CSS选择器的对象的第一个元素

语法：Element Element.querySelector（selector）

* 1. querySelectorAll（）

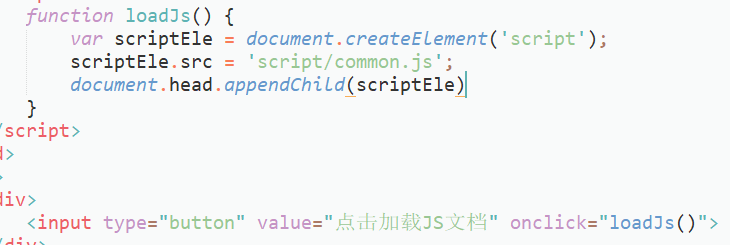
描述：返回由使用指定的CSS选择器的对象组成的集合（数组）

语法：NodeList Element.querySelectorAll（selector）

1. 方法

### extra

JS异步加载



## BOM编程

1. BOM（Browser Object Model），浏览器对象模型，提供与浏览器相关的API，window对象是BOM的顶级对象，代表浏览器窗口或iframe或frame.

### window对象

1. 属性
   1. document

描述：返回HTMLDocument对象

* 1. history

描述：返回History对象

* 1. screen

描述：返回Screen对象

* 1. navigator

描述：返回Navigator对象

* 1. location

描述：返回Location对象

* 1. Math

描述：返回Math对象

1. 方法
   1. alert（）

描述：弹出警示对话框（只有一个确定按钮）

语法：window.alert（string）

* 1. confirm（）

描述：弹出询问对话框（有确定和取消两个按钮）

语法：bool window.confirm（string）

if (window.confirm('确认删除吗')) {

window.alert('删除成功');

}

* 1. setInterval（）

描述：每间隔指定的时间执行相关的代码

语法：int window.setInterval（function，milliseconds）;

* 1. setTimeout（）

描述：间隔指定的时间后执行相关的代码（执行一次）

语法：int window.setTimeout（function，int timeId）;

* 1. clearInterval（）

描述：清除setInterval（）设置的timeId

语法：window. clearInterval（int timeId）;

* 1. clearTimeout（）

描述：清除setTimeout（）设置的timeId

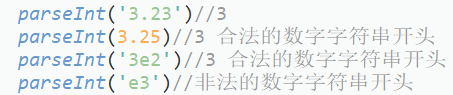
语法：window.clearTimeout（int timeId）;



* 1. parseInt（）

描述：转换成整数

语法：int parseInt（）



* 1. parseFloat（）

描述：转换成浮点数

语法：int parseFloat（）

* 1. isNaN（）

描述：检测值是否为NaN（）

语法：bool isNaN（value）

说明：如果在当前窗口内控制窗口内的对象时，window对象可以省略

### Location对象

1. 属性
   1. href属性

描述：获取/设置地址栏中的地址信息

语法：location.href = string

var 变量名称 = location.href

var selEle = document.getElementById('freindlink');

var url = selEle.value;

location.href = url;

* 1. search

描述：获取地址栏”?”以后所有的参数（包含?）

语法：string location.search

1. 方法
   1. reload（）

描述：刷新页面

语法：location.reload();

document.getElementById('random').innerHTML = Math.random();

function refresh(){

location.reload();

}

### Screen对象

属性

* 1. width

描述：获取显示屏幕分辨率

语法：number screen.width

* 1. height

描述：获取显示屏幕分辨率

语法：number screen.height

### History对象

1. back（）

描述：后退一步

语法：history.back(void)

1. forward（）

描述：前进一步

语法：history. forward (void)

1. go（）

描述：前进/后退

语法：history.go（number）

说明：如果为负数，则后退；否则前进

history.back()等价于history.go(-1)

history.forward()等价于history.go(1)

### Navigator对象

属性

* 1. userAgent（简称UA）

描述：返回代理器信息

语法：string navigator.userAgent

### JavaScript事件属性

1. 鼠标事件
   1. onclick
   2. onmouseover
   3. onmouseut
2. 键盘事件
   1. onkeydown 按下键盘按键
   2. onkeyup 释放键盘按键
   3. onkeypress 按下并释放按键
3. 表单对象
   1. onsubmit
   2. onreset
4. 列表框对象
   1. onchange
5. body对象
   1. onload
   2. onunload
6. 事件对象

获取事件对象

1、

<button onclick="btnClick(event)"></button>

function btnClick(event){}

2、

<button id="btnTest"></button>

var btn = document.getElementById("btnTest");

btn.onclick = function(event){

event : 表示当前事件的事件对象}

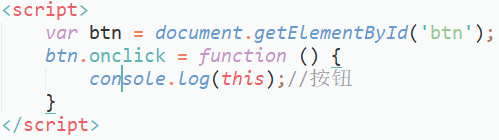
event：表示当前事件的事件对象

this

1.this在全局函数中，永远都是window对象



2.this在局部函数中，表示的都是函数的所有者



1. 事件对象的属性
   1. offsetX

获取鼠标所操作点的横坐标

* 1. offsetY

获取鼠标所操作点的纵坐标

* 1. target

获取事件源（事件所在的元素）

### extra

1.iframe标记

<iframe src="网页的url地址" frameborder="是否存在边框线（0/1）" scrolling="滚动条的显示方式（yes/no/auto）"></iframe>

2.分页的原理

只是每一页显示的效果

MySQL数据库的实现就是利用SELECT语句的LIMIT子句实现

SELECT ……LIMIT [offset,]pagesize

offset是指从第几条记录开始返回（编号从0开始）；offset的计算公式

## JSON

1. JSON基础

JSON（JavaScript Object Notation），是一种轻量级的数据交换格式；

JSON官网：<http://www.json.org>

1. 支持的数据格式
   1. 数组
   2. 对象
2. JSON的数据类型
   1. string
   2. number
   3. Boolean
   4. object
   5. array
   6. null

# 案例（DOM）

### 一、邮件全选

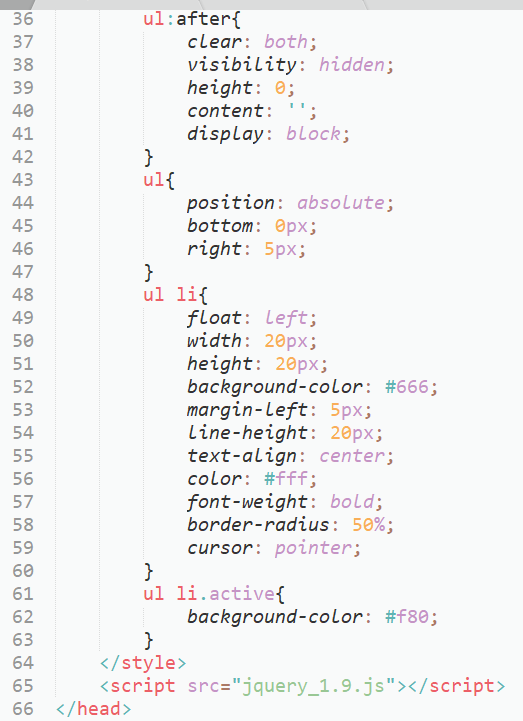






### 二、轮播图









### 三、列表二级联动





### 四、页面遮罩





### 五、换肤

注意css样式的重复加载问题



### 六、按钮倒计时



## extra

#### input元素有两个方法

blur（）：失去焦点

usernameEle.blur()

focus（）：获得焦点

usernameEle.focus()

#### SQL注入

密码：a’ ’or ‘1’=’1

SQL语句：SELECT \* FROM 表名 WHERE name = ‘xxx’ AND password = ‘a’ OR ‘1’=’1’，查询结果为真，则随意可以登录

#### 关于table

在HTML中。表格的行可以分组：thead（表头）、tbody（表体）、ffoot（表脚）

# 4.jQuery-js框架

赵旭 zhaoxu@tedu.cn

提高JavaScript开发效率

### jQuery介绍

是一个轻量级的JS库，是一个被封装好的JS文件，提供了更为简便的元素操作方式

核心理念：Write Less Do More

### 使用jQuery

1. 引入jQuery文件

<script src=”路径”></script>

注意：引入文件的操作必须要放在其他jQuery操作之前

1. 使用jQuery

### jQuery对象（重点）

1. 什么是jQuery对象

jQuery对象是由jQuery对页面元素封装后的一种体现

jQuery中所提供的所有操作只针对jQuery对象，其他对象（DOM）不能使用

1. 工厂函数-$()

作用：

* 1. 可以将页面元素加工成jQuery对象
  2. 可以将DOM对象转换成jQuery对象

1. DOM对象和jQuery对象之间的转换

DOM对象：不能用jQuery提供的操作

jQuery对象：不能用DOM提供的操作

* 1. 将DOM对象转换为jQuery对象

语法：var 变量 = $（DOM对象）

注意：所有的jQuery对象在起名的时候，最好在变量名称前加$符号，用于和DOM对象区分

var d1 = document.getElementById（”d1”）;

var $d1 = $（"d1”）;

* 1. 将jQuery对象转换为DOM对象
     1. var DOM对象 = jQuery对象[下标]；
     2. var DOM对象 = jQuery对象.get（下标）；

#### jQuery 选择器

1. 作用：获取页面上的元素，返回值都是由jQuery对象组成的数组
2. 语法：$（"选择器”）；
3. 选择器详解

##### 基本选择器

* + - 1. ID选择器

语法：$（”#id”）

返回：封装了包含id值的jQuery数组

* + - 1. 类选择器

语法：$（".className”）

返回：页面中指定className的所有元素

* + - 1. 元素选择器

语法：$（”element”）

返回：页面中指定标记的所有元素

* + - 1. 群组选择器/复合选择器

语法：$（"select1,select2…”）

返回：满足函数内所有选择器的元素

* + - 1. 通用选择器

语法：$（"\*”）

返回：返回页面中所有的元素

##### 层级选择器

1. 后代选择器

语法：$（"selector1 selector2”）

1. 子代选择器

语法：$（"selector>selector2”）

1. 相邻兄弟选择器

语法：$（"selector1+selector2”）

作用：匹配紧跟在selector1后面且满足selector2选择器的元素

$(“.title+p”)

1. 通用兄弟选择器

语法$（"selector1~selector2”）

作用：匹配selector1后面所有满足selector2选择器的元素们

##### 基本过滤选择器

1. ：first

只能匹配到第一个元素

$（"p:first”）

1. ：last

只能匹配到最后一个元素

$（" p:last”）

1. ：not（selector）

将满足selector选择器的元素排除在外

$（"p:not(first)”）

1. ：odd

匹配第偶数个元素（下标为奇数）

1. ：even

匹配第奇数个元素（下标为偶数）

1. ：eq（index）

匹配下标等于index的元素

1. ：gt（index）

匹配下标大于index的元素

1. ：lt（indes）

匹配下标小于index的元素

##### 属性过滤选择器

通过元素的属性及其值来匹配页面的元素

1. [attribute]

作用：匹配包含指定属性attribute的元素

div[id]：包含id属性的元素

1. [attribute = value]

作用：匹配attribute属性值为value的元素

input[type = ‘text’]

1. [attribute != value]

作用：匹配attribute属性值不为value的元素

1. [attribute ^=value]

匹配attribute属性值为value开头的元素

1. [attribute $ = value]

匹配attribute属性值为value结尾的元素

1. [attribute \* = value]

匹配attribute属性值中包含value字符的元素

##### 子元素过滤选择器

1. ：first-child

作用：匹配属于其父元素中的首个子元素，可能会匹配出多个

$(“li:first-child”);li的父元素中的第一个子元素



1. ：last-child

作用：匹配属于其父元素中的最后一个子元素，可能会匹配出多个

1. ：nth-child（n）

作用：属于其父元素中的第n个子元素，可能会匹配出多个

#### jQuery操作DOM

##### 基本操作

1. html（）

作用：获取或设置jQuery对象中的HTML内容，等同于DOM中的innerHTML属性

1. text（）

作用：获取或设置jQuery对象中的文本内容，等同于DOM中的innerText属性

1. val（）

作用：获取或设置jQuery对象的value属性值（主要针对表单控件）

<input type=’text’ id=’username’>

console.log($(‘#username’).val())

获取username的值并打印

$(“username”).val(‘Mary’)；设置username的值

##### 属性操作

1. attr（）

作用：读取或设置jQuery对象的属性值

$obj.attr（"id”）读取$obj对象的id属性值

$obj.attr（"id”,”container”）设置$obj对象的id属性值为container

1. removeAttr（"attrName”）

作用：删除jQuery对象的attrName属性

$obj.removeAttr（”class”）从$obj对象中将class属性移除出去

##### 样式操作

1. attr（"class”,”className”）

使用attr（）动态绑定class属性值

$obj.attr(“class”,”container”)

1. addClass（"className”）

作用：将className类选择器追加到jQuery对象class值之后

返回：返回当前的jQuery对象

$obj = $obj.addClass（"bigFont”）;

$obj.addClass（"red”）;

1. removeClass（"className”）

作用：删除指定的类选择器

1. removeClass（）

作用：删除所有的类选择器

1. toggleClass（"className”）

切换样式：

元素如果具有className类选择器，则删除

元素如果没有className类选择器，则添加

1. css（”属性”,”值”）

为元素设置某css属性及值

1. css(JSON对象)

为元素设置一组css属性及其值

$obj.css({

‘color’:’red’,

…

})

##### 创建对象

语法：$（"创建完整的标记”）

* 1. var $div = $（”<div></div>”）;创建一个div元素
  2. var $img = $（"<img>”）;创建一个img元素

##### 插入元素

1. 内部插入

作为子元素插入到网页中

* 1. $obj.append（$new）;

将$new元素追加到$obj元素的尾部（最后一个子元素）

* 1. $obj.prepend（$new）

将$new元素添加到$obj的头部（第一个子元素）

1. 外部插入

作为兄弟元素插入到网页中

* 1. $obj.before（$new）

将$new作为$obj的上一个兄弟元素插进来

* 1. $obj.after（$new）

将$new作为$obj的下一个兄弟元素插入进来

##### 删除节点

1. remove（）

语法：$obj.remove();

作用：删除$obj元素

1. remove（"selector”）

语法：$obj.remove（"selector”）;

作用：将满足selector选择器的元素删除

1. empty（）

语法：$obj.empty（）；

作用：清空$obj内的内容

##### 遍历节点

1. children（）/children（selector）

作用：获取对象中的子元素（只考虑元素节点）或满足selector选择器的子元素（只考虑元素节点）

注意：只能获取子元素，其他后代元素不可获取

1. next（）/next（selector）

作用：获取某对象的下一个兄弟元素，或者是满足selector的下一个兄弟元素

1. prev（）/prev（selector）

作用：获取某对象的上一个兄弟元素，或者是满足selector的下一个兄弟元素

1. siblings/siblings（selector）

作用：获取某对象的所有兄弟元素，或者是满足selector的所有兄弟元素

1. find（selector）

作用：查找满足selector选择器的所有后代元素

1. parent（）

作用：获取某对象的父元素

#### jQuery事件处理

##### 页面加载后的执行

指在DOM树加载完成后就要执行的操作

1. $(document).ready（function（）{页面的初始化操作，DOM树加载完毕后开始执行}）；
2. $（）.ready（function（）{}）；
3. $（function（）{}）；

##### jQuery的事件绑定

1. 方式一

$obj.bind（“事件名称”，事件处理函数）；（如：click、mousemove）

$obj.bind(“click”,function(){alert(“单击事件”)})；

1. 方式二

$obj.事件名称（事件处理函数）；

$obj.click(function(){…})

注意：在事件处理函数中，允许使用this来表示触发当前事件的DOM对象

##### 事件对象-event

在绑定事件的时候，允许传递event参数来表示事件对象

1. $obj.bind（"click”,function(event){…}）
2. $obj.click（function(event){…}）

##### 事件冒泡

即在执行子元素事件时，同时把父元素/祖先元素的对应的事件也执行了

阻止事件冒泡：event.stopPropagation（）



##### jQuery-动画

1. 基本显示/隐藏

语法：

显示：$obj.show()/$obj.show（执行时间）

隐藏：$obj.hide()/$obj.hide（执行时间）

1. 滑动式显示/隐藏

语法

显示：$obj.slideDown()/$obj. slideDown（执行时间）

隐藏：$obj. slideUp ()/$obj. slideUp（执行时间）

1. 淡入淡出式显示/隐藏

语法

显示：$obj.fadeIn()/$obj. fadeIn（执行时间）

隐藏：$obj. fadeOut ()/$obj. fadeOut（执行时间）

#### extra

1、

使用jquery的each函数快速遍历arr

$.each(array, function(key,obj ){

index:表示遍历元素的下标

obj：表示当前遍历出来的元素

})



##### jQuery插件

# Django框架

#### WEB与服务器

* 1. WEB：表示用户可以浏览的网页（HTML,CSS,JS）
  2. 服务器：能够给用户提供服务的机器
     1. 硬件与软件

硬件：一台计算机

软件：一个能够接收用户请求并给出响应的程序

1、APACHE

2、TOMCAT

3、IIS（Internet Information Service）

4、Nginx

* + 1. 作用
       1. 存储WEB所需要的信息（HTML、图片、音频）
       2. 能够处理用户的请求（request）并给出回应（response）
       3. 执行服务器端的程序

#### 框架

1. 什么是框架：是一个为了解决开放性问题而存在的一种结构。框架本身提供了一些最基本的功能，我们需要在基本的功能基础上开发属于自己的操作
2. Python中的框架（WEB）
3. Django：重量级的WEB框架
4. Tornado：异步框架
5. Flask：轻量级框架
6. Django框架
   1. 什么是Django：是一个开源框架，2005年发布，采用Python语言开发，早期是做新闻以及内容管理的网站，提供了强大的后台管理系统
   2. Django的框架模式—MTV（Modles）

M：Modles层

模型层，负责数据库的建模以及CRUD（增删改查）的操作

T：Templates层

模板层，用于处理用户显示的内容，如HTML

V：Views层

视图层，处理与用户交互的部分内容，从模型中获取数据，再将数据发送给模板，最终再将模板显示给用户

\*\*MVC

M：Models 模型层，负责数据库的建模以及CRUD（增删改查）的操作

V：Views 视图层，处理用户显示的内容，如HTML

C：Controller 控制器，处理与用户交互部分的内容，接收请求，处理请求并响应结果给客户端

MTV MVC

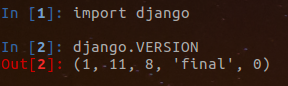
M------------M

T--------------V

V------------C

#### Django框架的使用

1. Django的安装
   1. 查看已安装的Django版本
      1. 进入终端python交互模式
      2. 输入import django，未报错则已安装了Django
      3. 查看已安装的版本，交互模式中：django.VERSION



* 1. 安装
     1. 在线安装-使用pip/pip3

pip：安装到python2.7下

pip3：安装到python3.5下

* + - 1. sudo pip3 install django（默认安装Django的最高版本）
      2. sudo pip3 install django==1.11.8（指定安装版本）
    1. 离线安装
       1. 下载所需的Django包

<http://www.djangoproject.com>

* + - 1. 在环境中解压Django包

tar –xvfDjango-1.11.8.tar.gz

* + - 1. 进入目录中

cd Django-1.11.8

* + - 1. 安装

sudo python3 setup.py install

* 1. Django文档
     1. 官网

<http://www.djangoproject.com>

* + 1. DjangoBook

<http://djangobook.py3k.cn/2.0/>

* 1. 使用Django
     1. 创建目录：用于保存Django项目

mkdir Django

* + 1. 创建Django项目

使用django-admin指令创建Django项目

django-admin startproject 项目名称

* + 1. 启动服务，访问网站

在创建好的项目中找到manage.py文件，通过其启动服务

python3 manage.py runserver

访问地址：

<http://127.0.0.1:8000/>

http://localhost:8000/

* + 1. Django的项目结构介绍
       1. manage.py

负责执行django中各项操作的文件

如：启动服务、创建应用、数据库的同步操作、创建后台的超级管理员

使用方式

1、python3 manage.py 子命令

2、./manage.py 子命令（如果需要在python3下运行，则打开manage.py文件，第一行改为python3）



* + - 1. 主文件夹（名称与项目名称相同）
         1. \_\_init\_\_.py

项目初始化文件，当服务器启动的时候，自动执行

* + - * 1. urls.py

项目基础路由配置文件（主路由配置文件）

包含所有的地址映射，将请求地址映射到对应的视图上

每一个请求到达后，都会由urls.py中的urlpatterns列表中的url（）进行匹配，当url（）匹配上之后，可能就会将请求转交给其他试图（Views）或其他的urls.py去处理

* + - * 1. wsgi.py

应用服务器的配置文件，暂时不用

* + - * 1. settings.py

项目的主设置文件：模板、数据库、应用、语言……

1、BASE\_DIR：当前项目所在的绝对路径

2、DEBUG：调试模式（布尔），开发过程中推荐使用True，上线后改为False

3、ALLOWED\_HOSTS：设置允许访问本项目的地址列表，如果不设置的话，只有本机可以访问，推荐写（’\*’），任何表示该机器的地址都可以访问当前项目

\*\*注意：如果允许被外部及其访问的话，则：

./manage.py runserver 0.0.0.0:8000

4、INSTALL\_APPS：指定已安装的应用，如果有自定义应用的话，需要在此注册

5、ROOT\_URLCONF：

6、TEMPLATES：指定模板的信息

7、DATABASES：指定数据库的信息

8、LANGUAGE\_CODE：指定当前项目的语言，允许修改为zh-Hans（表示简体中文）

9、TIME-ZONE：指定时区，建议修改为：Asia/Shanghai

#### url函数详解

1. url函数的语法

url（regex，views，kwargs=None，name=None）

* + 1. regex

正则表达式模式，匹配请求的url，字符串类型

* + 1. views

url对应的处理的视图函数，直接写函数名

* + 1. kwargs

字典，用于向views进行关键字传参，如果没有可以省略

* + 1. name

字符串，给url（）起别名，方便在模板中使用

1. 通过url向views传递参数
   * 1. 使用正则表达式子组传参

使用子组传参-（），一个子组是一个参数，如果需要传递多个参数，则使用多个子组，使用/间隔开

urlpatterns = [

#访问路径是run/，交给run\_views

url（r”^run/$”,run\_views），

#访问路径是run/两位数字，交给run\_arg1\_views

rul（r”^run/(\d{2})/$”，run\_arg1\_views）

]

\*\*注意：

1、在url（）中，一个子组表示一个参数

2、在views.py中对应的处理函数要根据url（）中子组的个数，相应的定义参数（形参），定义的参数要位于request之后

* + 1. 使用url（）第三个参数，字典传参

urlpatterns = [

url（r”^show/$”, show\_views, {“name”:”linda”，…}）  
]

def show\_views(request,name):

pass

#### Django中的应用

1. 什么是应用

应用就是网站中一个独立的程序

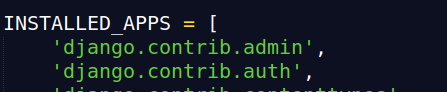
在Django中，主目录一般不处理用户的具体请求，主要做的是项目的初始化以及请求的分发

1. 创建应用
   * 1. 通过指令创建应用

./manage.py startapp 应用名称

* + 1. 在settings.py中对应用进行注册

在INSTALLED\_APPS中追加应用名称



* + 1. 应用的结构组成
       1. migrations目录

存放数据库中间文件的目录（日志文件）

* + - 1. \_\_init\_\_.py

应用的初始化文件

* + - 1. admin.py

应用的后台管理文件

* + - 1. apps.py

应用的属性配置文件

* + - 1. models.py

模型配置文件

* + - 1. tests.py

测试模块

* + - 1. views.py

视图处理文件

#### Django中的模板（Templates）

1. 什么是模板

要动态给用户呈现的网页

模板就是一个网页-前后端结合的网页，通过视图（Views）呈现给用户

1. 模板的设置

在settings.py中的TEMPLATES变量进行模板的设置

* + 1. BACKEND：指定的搜索引擎，不用改动
    2. DIRS：指定模板所存放的目录

如果DIRS内容为空的话，Django会自动地到每个应用下去搜索一个叫templates的目录作为模板的存放目录

推荐：1.DIRS内容为空

2.在每个应用中，创建一个templates目录

* + 1. APP\_DIRS

True：首先从DIRS中指定的目录中去查找模板，没找到的话再搜索templates目录

1. 模板的加载方式

模板的加载需要在“视图”中完成

* + 1. 使用loader获取模板，通过HTTPResponse进行响应

from django.template import loader

from django.http import HttpResponse

def index\_views(request):

#通过loader加载模板（得到一个模板对象）

temp = loader.get\_template(“模板名称”)

#通过模板对象temp，将模板渲染成字符串

html = t.render()（可传一个字典为参数）

#通过HTTPResponse将html响应给客户端

return HttpResponse(html)

* + 1. 使用render直接加载并返回模板

from django.shortcuts import render

def xxx\_views(request):

return render(request, “模板名称”)

1. 模板规范
   * 1. 变量
        1. 作用：允许将后端的数据传递给模板，在模板中，会根据变量的值进行显示
     2. Django中允许传递给模板做变量的数据类型

数字，字符串，列表，元组，字典，函数，对象

* + 1. 变量的语法

通过一个字典，将要传递给模板的变量封装起来

dic={‘变量1’:’值1’…}

1、使用loader

temp = loader.get\_template(“模板名称”)

html = t.render()（dic）

return HttpResponse（html）

2、使用render

return render（request，’模板名称’，dic）



* + 1. 在模板中使用变量

使用{{变量名}}

如果变量不存在或值为空的话，此位置不显示任何内容

#### extra

函数locals（），返回将局部变量和其值封装成字典

函数globle（），返回将全部变量和其值封装成字典

1. 标签
   * 1. 什么是标签

允许将服务器端的功能嵌入到模板中

* + 1. 语法

{% 标签内容 %}

* + 1. 常用标签
       1. for
          1. 语法

{% for 变量 in 列表/元组/字典 %}

<h3>{{变量}}</h3>

<h3>{{变量.属性}}</h3>

<h3>{{变量.方法}}</h3>

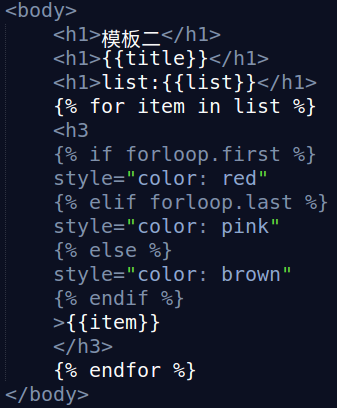
{% endfor %}

允许使用内置变量-forloop获取循环的信息，forloop在{%for%}中是免声明就可以使用的

forloop.counter：获取当前循环次数

forloop.counter0：获取当前循环下标

forloop.first：是否为第一次循环

forloop.last：是否为最后一次循环

* + - 1. if
         1. if标签

语法：

{% if 条件 %}

满足条件时要运行的内容

{% endif %}

* + - * 1. if … else …标签
        2. if … elif…else…标签
      1. 过滤器

作用：

在显示变量的数据之前，允许对数据进行筛选或改变

语法：

{{变量|过滤器}}

常用过滤器：

1.{{value|upper}}

将value变为大写字符输出

2.{{value|lower}}

将value变为小写字符输出

3.{{value|发咯atformat：n}}

将value四舍五入到n位小数

4.{{value|truncatechars：n}}

将字符串截取到n位字符（包含自动生成的…），故实际字符为n-3

1. url（）中的name参数

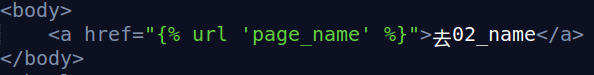
url（regex，views，kwargs=None，name=None）

name：允许为当前的url设置别名，可以在Template中使用别名找到相对应的url

语法：url（regex，views，name=“别名”）

在模板中使用别名找到对应的url：{% url “别名” 参数1 参数2 %}





* 1. 静态文件的处理

即不与用户发生动态交互的文件，称为静态文件

如：css，js，img，audio，video，html（部分）

* + 1. Django中静态文件的处理

需要在setting.py中设置静态文件的访问路径和存储路径

1.访问路径：STATIC\_URL=”/static/”

作用：当访问路径是localhost：8000/static/…时，一律去静态文件路径中去搜索静态文件

2.存储路径：STATICFILES\_DIRS = （BASE\_DIR, “静态目录名称”）

在项目以及各个应用中都可以创建一个对应名称的目录，来表示项目或应用中所存储静态文件的目录

* + 1. 访问静态文件
       1. 直接使用localhost：8000/static/…

<img src=”<http://localhost:8000/static/...>”>

<img src=[”/static/...](http://localhost:8000/static/...)”>

* + - 1. 使用{% static %}标签访问静态文件

{% static %}表示的就是静态文件访问路径

1.在模板的最顶层增加：{% load static %}

2.在使用静态文件时：<img src=”{% static ‘具体路径’ %}”>

如：<img src=”{% static ‘images/huiyuan.jpg’ %}”>

1. 模板的继承
   * 1. 什么是模板的继承

当多个模板（网页）具备大部分相同的内容时，就可以使用继承的方式，将相同的内容继承下来，再增加/修改属于自己的内容即可。

* + 1. 模板的继承的语法
       1. 在父模板中

必须要表示出在子模板中可以修改的内容

标签：{% block 名称 %}

父模板中要显示的内容

{% endblock %}

block的作用：

1.在父模板中，正常显示

2.在子模板中，如果不修改block中的内容，即按照父模板中的内容进行显示

* + - 1. 在子模板中
         1. 在最顶层添加{% extends ‘父模板名称’%}标签，表示继承关系
         2. 增加block标签，改写属于自己的内容

{% block 名称 %}

子模板中的内容

{% endblock %}

在子模板中的block会覆盖父模板中的同名block中的内容

#### 模型（Models）

1. 什么是模型

模型是根据数据库中表的结构来创建出来的class；

每一张表映射到编程语言中就是一个class，表中的每一列，解释class中的一个属性，在模型中还能完成对数据的CRUD操作

1. 创建和使用模型-ORM
   * 1. 什么是ORM

即Object Relational Mapping

简称：ORM，O/RM, O/M Mapping

中文：对象关系映射

ORM的三大特征：

1.数据表（Table）到类（class）的映射

将数据表自动生成一个class类，将class类自动生成一张表

2.数据类型的映射

允许将表中的字段的数据类型，自动映射到编程语言中的对应的数据类型，允许将编程语言中的数据类型，自动映射成表中字段的数据类型

3.关系映射

数据库中表的关联关系：一对一、一对多、多对多

将表与表之间的关系也映射到编程语言中，通过创建类与类（对象与对象）之间的关系来完成

* + 1. ORM的优点
       1. 提高了开发效率，能够自动完成表到对象的映射
       2. 不用SQL编码，也能够完成多数据的CRUD操作，可以省略庞大的数据访问层
    2. 创建和设置数据库
       1. 创建一个数据库（支持中文）

create database 数据库名 default charset utf8 collate utf8\_general\_ci；

mysql可视化开发工具-Navicat

创建一个数据库-webdb

* + - 1. 在setting.py

连接MySQL的配置：  
1、ENGINE：引擎django.db.backends.mysql

2、NAME：指定要连接到的数据库的名称

3、USER：指定的用户名称，通常为root

4、PASSWORD：指定密码

5、HOST：要连接的主机地址，本机：localhost或127.0.0.1

6、PORT：指定的端口号，通常为3306

注意：在django中来连接MySQL的话依赖于MySQLdb

* + - 1. 在主目录下的\_\_init\_\_.py中写入如下代码：

import pymysql

pymysql.install\_as\_MySQLdb()

* + - 1. 数据库的同步操作
         1. ./manage.py makemigrations

作用：将每个应用下的models.py文件生成一个数据库的中间文件，并保存在migrations文件夹中

* + - * 1. ./manage.py migrate

作用：将每个应用下的migrations文件夹下的中间文件同步到数据库中

* + - 1. 编写Models（重点）
         1. 注意：

Models中的每个class都称为模型类（Model类），或实体类（Entry）

实体：就是数据表中的一条记录

实体完整性：约束表中的记录不重复

Models中的每个实体类，必须继承自models.Model类

1. jango中的数据字段和字段类型

语法：属性 = models.字段类型（字段选项1，字段选项2，…）

1、字段类型（Field Type）

1、BooleanField（）

2、CharField（）

3、DateField（）

4、DateTimeField（）

5、DecimalField（）

6、EmailField（）

7、FloatField（）

8、ImageField（）#存图片的路径-varchar

用法：pic = models.ImageField(upload\_to=’ ’)

9、IntegerField（）

10、UrlFIeld（）#存网址-varchar

11、TextField（）#存大量数据-text

2、字段选项（Field Options）

1、max\_length：设置最大长度，在models.CharField（）中是必选项

2、default：设置默认值

3、null：指定当前属性是否可以为空，默认为False

4、unique：指定当前属性的值是否为唯一，默认为False

5、blank：指定当前属性是否允许为空字符串，默认为False

1. 数据的版本切换
   * 1. ./manage.py migrate

执行所有应用中最新版本的数据库中间文件

* + 1. ./managt.py migrate 应用名称 版本号

执行应用中对应版本号的中间文件，恢复到对应版本号的状态，其后的中间文件映射操作撤销

1. 通过数据库自动导出Models
   * 1. ./manage.py inspectdb > 文件名.py

##### 模型中的CRUD

1. 通过ORM向DB中增加数据
   * 1. Entry.objects.create(属性=值，属性=值…)（Entry为对象）

models.Author.objects.create(name='Rebeca', age=100, email='Rebeca@gmail.com')

* + 1. 创建一个Modules对象，并通过save（）完成增加

obj=Entry（属性=值，属性=值）

obj=save（）

author = models.Author(name='Austin', age=80, email='Austin@gmail.com')

author.save()

* + 1. 使用字典构建对象，并调用其save（）方法来完成增加

dic-{

‘属性1’：‘值1’，

‘属性2’：‘值2’

}

obj = Entry（\*\*dic）

obj.save

dic = {'name':'Astlofsky', 'age':120, 'email':'@gmail.com'}

author = models.Author(\*\*dic)

author.save()

1. 查询操作

通过Entry.objects调用查询接口

* + 1. 基本查询操作

语法：all（）

用法：Entry.object.all（）

返回：QuerySet（封装了若干对象的列表）

authors = models.Author.objects.all()

* + 1. 查询指定列的操作
       1. values(\*fields, \*\*expressions)

语法：values（‘列1’， ‘列2’）

用法：Entry.objects.values（‘列1’，‘列2’）

返回：QuerySet（封装了若干字典的列表，一个字典由一条记录对应字段值组成）

authors = models.Author.objects.values(‘name’,’age’)

author🡪[{‘name’:’Austin’,’age’:’80’},

{‘name’:’Rebeca’,’age’:’100’},

{‘name’:’Astlofsky’,’age’:’90’}]

可放在所有查询筛选语句的最后

authors = models.Author.objects.all().values(‘name’,’age’)

* + - 1. values\_list(\*fields, flat=False, named=False)

返回值：返回一个由元组组成的列表，元组由每条记录的相应字段记录值组成

当只有一列需要查询时，可将flat设置为True，返回的就不是元组，而是单独的值

>>> Entry.objects.values\_list('id')

<QuerySet[(1,), (2,), (3,), ...]>

>>> Entry.objects.values\_list('id', flat=True)

<QuerySet [1, 2, 3, ...]>

当named=True时，元组的显示方式为

>>> Entry.objects.values\_list('id', named=True)

<QuerySet [id=1, id=2, ...]>：

* + 1. 指定排序效果

语法：order\_by（‘列1’，‘列2’，…）

默认：升序，如需降序，则在列前加-

models.Author.objects.order\_by（'age’）

models.Author.objects.order\_by（-'age’）

* + 1. 对条件取反

语法：exclude（条件）

用法：Entry.objects.exclude（属性=值，属性=值）

models.Author.exclude(id=3)

* + 1. 根据条件查询部分行数据（重点）

语法：filter（条件）

用法：Entry.objects.filter（条件）

1、直接使用Entry中的属性作为查询条件，如果有多个查询条件使用逗号隔开，映射到SQL语句中是使用and关联的

models.Author.objects.filter(‘id’=1)

2、通过Field Lookups 完成复杂条件查询

Field Lookup：查询表达式（查询谓词）

每一个查询谓词都是一个独立的查询条件，可以用在所有的有查询条件的位置处

1、\_\_exact

作用：等值判断

用法：Entry.objects.filter(属性\_\_exact=1)

models.Author.objects.filter(id\_\_exact=1)

2、\_\_contains

作用：判断属性中是否包含指定关键字

用法：Entry.objects.filter(属性\_\_contains=值)

3、\_\_lt

作用：判断属性值是否小于指定的值

用法：Entry.objects.filter(属性\_\_lt=值)

4、\_\_lte

作用：判断属性值是否小于等于指定的值

用法：Entry.objects.filter(属性\_\_lte=值)

5、\_\_gt

作用：判断属性值是否大于指定的值

用法：Entry.objects.filter(属性\_\_gt=值)

6、\_\_gte

作用：判断属性值是否大于等于指定的值

用法：Entry.objects.filter(属性\_\_gte=值)

7、\_\_startswith

作用：筛选属性值是否以指定的值开头

8、\_\_endswith

作用：筛选属性值是否以指定的值结尾

* + 1. 查询只返回一条记录

语法：get（）

用法：Entry.objects.get（查询条件/谓词）

注意：该函数只能用在查询返回一条记录时使用，如果返回多条记录或不返回记录时，都会报错

1. 修改数据
   * 1. 修改单个数据

1查2改3保存

1、通过get（）得到要修改的数据

2、再通过实体对象的属性修改属性值

3、最后通过实体实体对象的save（）保存回数据库

au = models.Author.objects.get（id\_exact=1）

au.name=’xxx’

au.age=30

au.save()

* + 1. 批量修改数据

调用QuerySet的update（）方法即可

models.Author.objects.filter（id\_\_gt=1）.update（属性=值，属性=值，…）

1. 删除数据

调用实体对象/查询结果集的delete（）即可

obj = models.Author.objects.get（id=1）

obj.delete()

#### 转发&重定向

##### 转发

在一个视图中调用了另外一个视图，再将结果响应给浏览器

特点：浏览器地址栏的地址始终是第一个请求的地址

##### 重定向

重新向新的访问地址发送一个请求

语法：from django.http import HttpResponseRedirect

return HttpResponseRedirect（‘重定向地址’）

return HttpResponseRedirect('/check')

#### ORM操作

##### F操作和Q操作

1. F（）

作用：用于在执行中获取某列的值

语法：F（‘列名’）

from django.db.models import F

models.Author.objects.all().update(age = F(‘age’)+10)

1. Q（）

作用：在查询条件中，可以完成 或（or） 的操作

语法：Q（条件1）|Q（条件2）

from django.db.models import Q

models.Author.objects.filter（Q（id=1）|Q（age\_\_gte=50））

##### 原生的数据库操作方法

1. 查询

函数：raw（SQL语句）

语句：Entry.objects.raw（SQL语句）

返回：QuerySet

1. 增删改

def doSQL（request）：

with connection.cursor（） as cursor：

sql = ‘delete from Author…’

cursor.excute（sql）

return……

#### 使用后台管理Models

1. 后台的配置

登录地址：<http://localhost:8000/admin>

创建后台管理员：

./manage.py createsuperuser

Username:输入用户名

Email Address：电子邮件

Password：密码

Password（again）：重复密码

1. 注册Models（admin.py中）

from .models import \*

admin.site.register（Entry1）

admin.site.register（Entry2）

1. 修改models.py，处理显示的内容
   * 1. 在models.py中的各个class追加

def \_\_str\_\_（self）：

return self.name

* + 1. 通过Models类的内部类Meta定义更多的展现形式

class Author（models.Model）:

…

class Meta：

1、db\_table

指定该实体类对应到数据库中表的名字

取值：字符串

注意：需同步数据库才可生效

2、verbose\_name

定义该实体类在后台中的显示的名字（单数）

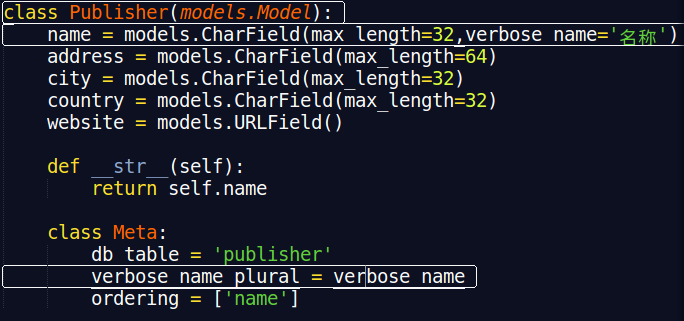
3、verbose\_name\_plural

效果同上（复数）

4、ordering

指定显示数据的排序规则

取值：一个列表，指定排序属性，默认是升序，需降序的话在属性前加-



* + 1. 字段名称在admin的显示可以在创建字段时通过verbose\_name处理

name = models.CharField(max\_length=32,verbose\_name='名称')

1. 高级管理
   * 1. 在admin.py中创建高级管理类-EntryAdmin

必须继承自admin.ModelAdmin

class AuthorAdmin（admin.ModelAdmin）:

pass

注册高级管理类

admin.site.register（Entry，EntryAdmin）

admin.site.register（Author，AuthorAdmin）

* + 1. 允许在EntryAdmin增加的属性（高级管理属性）
       1. list\_display

作用：指定在列表页中显示的字段们

取值：由属性名组成的列表或元组

list\_display = （'name’,’age’,’email’）

* + - 1. list\_display\_links

作用：定义在列表页中也能够连接到详情页中的字段们

取值：由属性名组成的列表或元组

注意：取值必须出现在list\_display中

list\_display\_links= （'name’，）

* + - 1. list\_editable

作用：定义在列表页中就可以修改的字段们

取值：由属性名组成的列表或元组

注意：取值不能出现在list\_display\_links中

list\_editable = （’age’,’email’）



* + - 1. search\_fields

作用：指定在列表页中允许被搜索的字段们

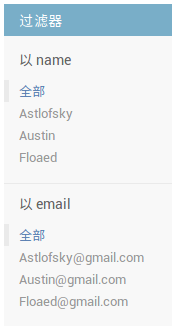
取值：由属性名组成的列表或元组



* + - 1. list\_filter

作用：在列表页右侧增加一个过滤器，实现快速筛选

取值：由属性名组成的列表或元组



* + - 1. date\_hierarchy

作用：在列表页的顶部增加一个时间选择器

取值：必须是DateField或DateTimeField的列



* + - 1. fields

作用：在详情页面中，指定显示哪些字段并按照什么样的顺序显示

取值：由属性名称组成的列表或元组

* + - 1. fieldsets

作用：在详情页中，对字段们进行分组显示

注意：fieldsets和field不能共存

fields=['name','age','email','isActive']

语法：

fieldsets=（

分组1

（‘分组名称’，{‘fields’：（‘属性1’：‘属性2’，）}），

……

）

fieldsets = (

('基本选项',{'fields':('name','age')}),

('高级选项',{'fields':('email','isActive'),

'classes':('collapse',)})

)

‘fields’为‘classes’，控制分组可折叠



#### 关系映射

1. 一对一映射
   1. 什么是一对一

A表中的一条记录只能与 B表中的一条记录关联

数据库的实现：

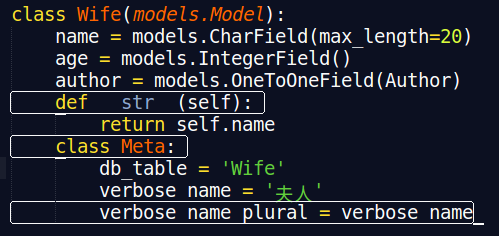
A表：设置主键

B表：增加一列，并引用自A表的主键，再增加一个唯一约束

* 1. 在Django中的语法规范

在关联的两个类的任何一个类中实现下列操作：

属性=models.OneToOneField（Entry）



* 1. 查询

正向查询：通过wife找author

获取id为1的wife的信息

wife = Wife.bojects.get(id=1)

再通过wife找到对应的author的信息

author = wife.author

反向查询：通过author找wife

获取id为3的author的信息

author = models.Author.objects.get（id=3）

再通过author找到对应的wife

wife = author.wife

1. 一对多映射

A表中的一条数据可以与B表中的任意多条数据关联

B表中的一条数据只能与A表中的一条数据关联

在数据库中的实现：

通过外键（Foreign Key）实现

在‘多’表中增加外键对‘一’表中的主键进行引用

语法：

使用外键

在‘多’的实体中增加：属性=model.ForeighKey（Entry）

publisher = models.ForeignKey(Publisher,null=True)

正向查询-通过Book查询Publisher

book = models.Book.ForeignKey（Publisher）

publisher = book.publisher

反向查询：

django默认会在Publisher中增加一个隐式属性-book\_set（多表类名全部小写\_set）

查询id为1的出版社的信息

pub=Publisher.objects.get(id=1)

1. 多对多映射(M:N)
   1. 在数据库中的实现

必须创建第三张表，关联涉及到的两张数据表

* 1. 语法：

在任何一个实体类中，增加：

entry = models.ManyToManyField（Entry）

如：class Author（models.Model）:

book = models.ManyToManyField（Author）

* 1. 查询
     1. 正向查询-通过Author查所有的Author

先获取author

author = Author.objects.get（id=1）

再获取对应的所有的book

booklist = author.book.all（）

通过关联属性.查询方法（）查询关联的数据

* + 1. 反向查询-通过book查询所有的author

django会在book实体类中增加一个隐式属性

author\_set（对应实体的全小写）

book = Book.objects.get（id=1）

auList = book.author\_set.all（）

#### 自定义查询对象-objects

1. 声明类EntryManager，继承自models.Manager

允许在EntryManager中添加自定义函数

class AuthorManager(models.Manager):

def auCount(self):

return self.all().count()

def ageCount(self,age):

return self.filter(age\_\_lt=age).count()

1. 使用EntryManager覆盖Models类中的objects

class Author(models.Model):

objects = AuthorManager()

1. 在CRUD中即可调用自定义的函数

def aucount\_views(request):

return HttpResponse(models.Author.objects.auCount())

#### 请求对象-HttpRequest

1. HTTP协议
   1. 请求
   2. 响应
2. HttpResquest介绍
   1. 什么是HTTPRequest

在django中就是请求对象的封装体现，封装了请求过程中的所有信息，在django中，被化身成了request封装到了视图处理函数中作为参数，在调用视图时自动传入

* 1. HTTPRequest中的主要内容
     1. request.scheme：请求协议
     2. request.body：请求主体
     3. request.path：请求路径
     4. request.get\_host（）：请求的主机地址/域名
     5. request.method：请求方法
     6. request.GET：get方法所提交的数据
     7. request.POST：post请求方法所提交的数据
     8. request.COOKIES：cookies中的数据

['COOKIES', 'FILES', 'GET', 'META', 'POST', '\_\_class\_\_', '\_\_delattr\_\_', '\_\_dict\_\_', '\_\_dir\_\_', '\_\_doc\_\_', '\_\_eq\_\_', '\_\_format\_\_', '\_\_ge\_\_', '\_\_getattribute\_\_', '\_\_gt\_\_', '\_\_hash\_\_', '\_\_init\_\_', '\_\_iter\_\_', '\_\_le\_\_', '\_\_lt\_\_', '\_\_module\_\_', '\_\_ne\_\_', '\_\_new\_\_', '\_\_reduce\_\_', '\_\_reduce\_ex\_\_', '\_\_repr\_\_', '\_\_setattr\_\_', '\_\_sizeof\_\_', '\_\_str\_\_', '\_\_subclasshook\_\_', '\_\_weakref\_\_', '\_encoding', '\_get\_post', '\_get\_raw\_host', '\_get\_scheme', '\_initialize\_handlers', '\_load\_post\_and\_files', '\_mark\_post\_parse\_error', '\_messages', '\_post\_parse\_error', '\_read\_started', '\_set\_post', '\_stream', '\_upload\_handlers', 'body', 'build\_absolute\_uri', 'close', 'content\_params', 'content\_type', 'csrf\_processing\_done', 'encoding', 'environ', 'get\_full\_path', 'get\_host', 'get\_port', 'get\_raw\_uri', 'get\_signed\_cookie', 'is\_ajax', 'is\_secure', 'method', 'parse\_file\_upload', 'path', 'path\_info', 'read', 'readline', 'readlines', 'resolver\_match', 'scheme', 'session', 'upload\_handlers', 'user', 'xreadlines']

1. HTTPResquest详解
   1. 获取get请求提交的数据
      1. form表单提交

<form action =’{% url ‘xxx’ %}’ method = ‘get’></form>

在服务器端：request.GET[‘控件的name属性值’]

* + 1. 通过地址发送请求

在地址栏上拼查询字符串（Query String）

http://localhost:8000/getDo/?uname=asdf&upwd=saf

* 1. 获取post请求提交的数据
     1. 发送post的方式

只有在表单的method为post时，才能是使用post

* + 1. 获取post请求提交的数据

value = request.POST[‘key’]

* + 1. 403-Forbidden-csrf
       1. csrf：Cross-Site Request Forgery

跨站点攻击

* + - 1. csrf解决方案
         1. setting.py（不推荐）

删除MiddleWare中的CsrfViewMiddleWare中间件

* + - * 1. 在处理视图上增加@csrf\_protest（不推荐）
        2. 在模板中的<from>下的第一句话增加{% csrf\_token %}

##### 使用forms处理表单

1. forms模块的作用

通过forms模块，允许将表单控件与py文件相结合，可以在网页中自动生成表单控件

1. 使用forms模块
   1. 在应用中创建forms.py文件
   2. 导入forms：from django import forms
   3. 创建class，一个class对应到页面上的一个表单，该类必须继承自froms.Form
   4. 在class中创建属性，
2. 在模板中解析forms对象

注意：

1、需要自定义<form>

2、需要自定义提交按钮

* 1. 手动解析

在视图中创建forms.Form对象，并发送到模板中

form = RemarkForm（）

return

render（request，‘xxx.html’，locals（）

手动解析：

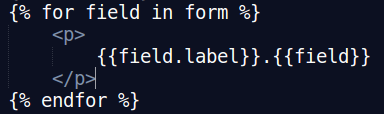
模板中：

{% for field in form%}

{{field.label}}表示label属性值

{{field}}：表示控件

{% endfor %}



* 1. 自动解析
     1. {{form.as\_p}}

将form中每个属性都解析成一个p标签，在p标签中再显示控件以及标签

{{form.as\_p}}

* + 1. {{form.as\_ul}}

将form中每个属性都解析成一个li标签，再显示控件以及标签

注意：需要自定义ul或者ol

<ul>

{{form.as\_ul}}

</ul>

* + 1. {{as\_table}}

将form中每个属性都解析成一个tr以及td标签，再显示控件以及标签

注意：需要自定义

<table’>

{{form.as\_table}}

</table>

1. 使用forms模块
   1. 在视图中，通过forms.Form快速获取表单数据
      1. 通过forms.Form的构造函数，接收post数据

form = xxxForm（request.POST）

* + 1. 需要让form通过验证后，再取值

目的：防止数据错误提交

form.is\_valid（）

返回true：通过验证

* + 1. 获取表单提交的数据

通过form.cleaned\_data获取提交的数据：返回字典

cd = form.cleaned\_data

value = cd[‘name’]…

##### 将 Models 和 Forms 结合到一起使用

1. 在 forms.py 中，创建 class，继承自 forms.ModelForm
2. 创建内部类：Meta

作用：关联Form和Models

* 1. model：指定要关联的Models类

model = Users

* 1. fields：指定从Models类中取出哪些字段生成控件
     1. 取值‘\_\_all\_\_’，全部属性都要生成表单控件
     2. 指定一个列表，允许生成表单控件的属性名

fields = [‘uname’,’upwd’]

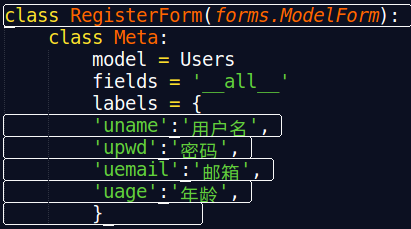
* 1. labels：指定每个属性要关联的label，取值为字典

labels = {

‘属性名’：‘label文本’，

……

}



1. 内置小部件
   1. 小部件（Widget）：表示的是生成到页面中的控件的类型以及其他属性，包括class等属性
   2. 常用小部件
      1. TextInput：type=‘text’
      2. NumberInput：type = ‘number’
      3. EmailInput：type = ‘email’
      4. URLInput：type = ‘url’
      5. PasswordInput：type = ‘password’
      6. HiddenInput：type = ‘hidden’
      7. CheckboxInput：type=‘checkbox’
      8. Textarea：<textarea></ textarea >
      9. Select：<select></ select >
   3. 小部件的使用
      1. 继承自forms.Form
         1. 基本版

属性 = forms.CharField（

label = ‘xxx’，

widget = forms.小部件类型

）

* + - 1. 高级版

属性 = forms.ChauField（

label = ‘xxx’，

widget = forms.小部件类型（

attrs = {

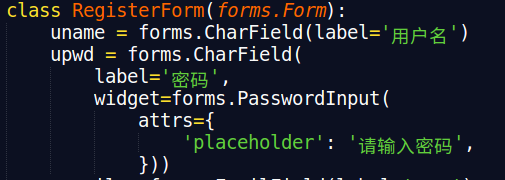
‘属性名’：‘属性值’，

……

}

）

）



* + 1. 继承自forms.ModelForm

class WidForm（forms.ModelForm）:

class Meta：

model = Users

fields = ‘\_\_all\_\_’

labels = {

‘属性1’：‘标签1’，

……

}

widgets = {

‘属性1’：forms.小部件类型（

attrs = {

‘属性’：‘值’，

……

}

）

}



##### Cookies

1. 什么是cookies

cookies是一种数据存储技术

将一段文本数据保存在客户端（浏览器）的一种技术，并可以长时间保存

1. 使用场合
   1. 保存登录信息
   2. 保存需要长期使用的数据
2. django使用cookies
   1. 设置cookies（将数据保存到客户端）

cookies的设置，在服务端是由相应对象发起的

语法：相应对象.set\_cookie（key，value，expires）

key：cookie的名字

value：cookie的值

expires:保存时间，以s为单位

1. 不使用模板-HTTPResponse

resp = HTTPResponse（‘给客户端的一句话’）

resp.set\_cookie（‘key’，‘value’，expires）

return resp

1. 使用模板-render（）

resp = render(request, 'xxx.html')

resp.set\_cookie('kdy', 'value2', expires)

return resp

1. 使用重定向=HttpResponse

resp = HttpResponseRedirect（‘/地址/’）

resp.set\_cookie('key', 'value2', expires)

return resp

* 1. 获取cookies（从客户端获取数据）

注意：每次请求时，请求对象（request）会将当前路径下所有的cookies都带到服务器

语法：request.COOKIES

##### session-会话

1. 什么是session

实际上是服务器上为每个浏览器开辟一段空间，用于保存相关的请求信息

1. session的使用场合

session也是为了存放数据而存在的，通常会把服务器端各个程序要用到的数据保存进去

1. Django中使用session
   1. 设置session的值

request.session[‘key’] = value

request.session.set\_expiry（time）

设置session过期时间，如果设置为0的话则表示关闭浏览器，session失效

* 1. 获取session的值

request.session[‘key’]

request.session.get（'key’，’default’）

* 1. 删除session的值

del request.session[‘key’]（删除之前先判断是否存在相应session）

* 1. 在settings.py中可添加的有关session的设置
     1. SESSION\_COOKIE\_AGE

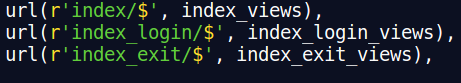
作用：设置sessionid在cookie中的存活时长

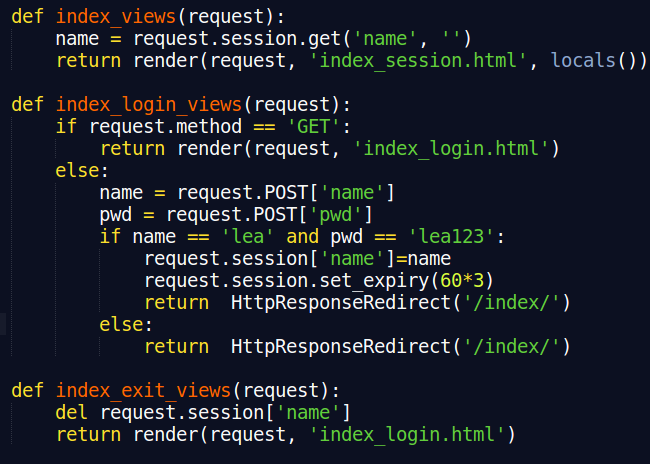
SESSION\_COOKIE\_AGE = 60\*30

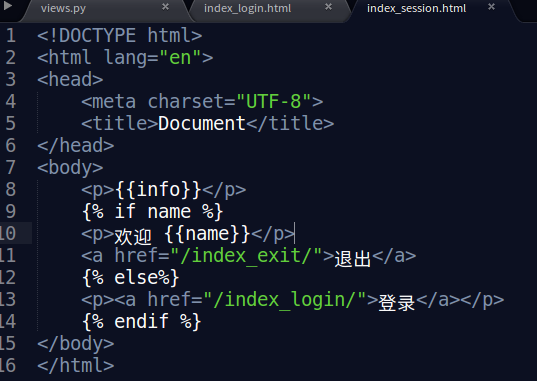
* + 1. SESSION\_EXPIER\_AT\_BROWSER\_CLOSE

作用：关闭浏览器时清除对应的session空间

SESSION\_EXPIRE\_AT\_BROWSER\_CLOSE=True







# AJAX

1. 什么是AJAX

AJAX-Asynchronous Javascript And Xml

异步 JS 和 Xml

异步地向服务器发送请求并接受相应数据

响应回来的数据格式：早期是xml，现在使用的是JSON

同步访问：当客户端向服务器发送请求时，服务器在处理的过程中，浏览器只能等待，效率偏低

异步访问：当客户端向服务器发送请求时，服务器在处理操作的同时，客户端可以做其他的操作，不需要一直等待，效率较高

使用场合：

1、搜索建议

2、表单验证

3、前后端完全分离时

1. AJAX核心对象 异步对象-XMLHttpRequest
   1. 什么是XMLHttpRequest

主要称为异步对象，代替浏览器向服务器异步地发送请求并接收响应

该对象有JavaScript创建

* 1. 创建异步对象

主流的异步对象时XMLHttpRequest类型的，并且主流浏览器全部支持XMLHttpRequest。但是在IE低版本中（IE6以及以下），不支持XMLHttpRequest，需要使用ActiveXObject来创建异步对象

判断浏览器是否支持XMLHTTPRequest

if（window.XMLHttpRequest）{

支持XMLHttpRequest

}else{

不支持XMLHTTPRequest，需要使用ActiveXObject创建异步对象

var xhr = new ActiveXObject（”Microsoft. XMLHTTP”）

}

* 1. xhr的成员
     1. 方法-open（）

作用：创建请求

语法：open（method，url，asyn）

method：异步请求的请求方式，取值‘get’或‘post’

url：请求地址，String类型

asyn：指定发送请求的方式，true：使用异步，false：使用同步

* + 1. 属性-readyState

作用：请求状态，通过不同的请求状态来表示xhr与服务器的交互情况

由0-4五个值来表示5个不同的状态

0：请求尚未初始化

1：已经与服务器建立连接

2：服务器端的请求已接收

3：请求处理中

4：响应已完成

* + 1. 属性-status

作用：服务器端的响应状态，用于表示服务器的状态

200：服务器正确处理所有的请求并给出响应

404：请求资源未找到

403：无权限

* + 1. 属性-responseText

作用：服务器端响应回来的数据

* + 1. 事件-onreadystatechange

作用：每当readyState发生改变时出发的操作-回调函数

只有当readyState的值为4并且status为200的时候，才可以正常地去接收responseText

* + 1. 方法-send（）

作用：发送请求

语法：send（body）

body：请求体

如果是get请求，此处为null

如果是post请求，此处为要提交的数据

1. 发送get请求
   1. 发送异步请求的步骤
      1. 创建xhr对象
      2. 创建请求-open（）
      3. 设置回调函数-onreadystatechange
      4. 发送请求-send（）
   2. 请求参数

推荐：地址栏传参

localhost:8000/xmlhttp/?name=value&age=name

1. 发送post请求
   1. 提交的参数要在send（）中编写

var xhr = getXhr（）；

…

xhr.send（‘name=value&age=value’）；

* 1. csrf验证

必须手动提交csrfmiddlewaretoken的值到服务器，否则无法通过csrf验证，服务器直接返回403

**解决方案1：**

* + 1. 在服务器从COOKIES中获取csrftoken的值并发送到模板中
    2. 在模板中获取csrftoken的值，拼成提交参数再发送给服务器端

xhr.send('name='+$('uname').val()+'&csrfmiddlewaretoken={{csrftoken}}'

弊端：由于要先走服务器，有可能在客户端中并没有csrftoken cookie，所以有可能会报错

**解决方案2：**

1. 在模板（前端页面）中，通过js获取cookies中csrftoken的值
2. 将获取出来的值拼成参数提交给服务器

xhr.send('name='+$('uname').val()+'&csrfmiddlewaretoken='+$.cookie('csrftoken'))

弊端：客户端中并必须包含csrftoken cookie，否则无法获取到对应的值，无法交给服务器，所以有可能会报错，可以在网页中任意位置处添加一个form，并且在form中只增加一个{% csrf\_token %}

**解决方案3**

1. 在模板（前端页面）中，通过js获取<input type=’hidden’ name=’csrfmiddlewaretoken’>的值
2. 将取出来的值拼成参数发送给服务器

xhr.send('name='+$('uname').val()+'&csrfmiddlewaretoken='+$("[name='csrfmiddlewaretoken]".val()));

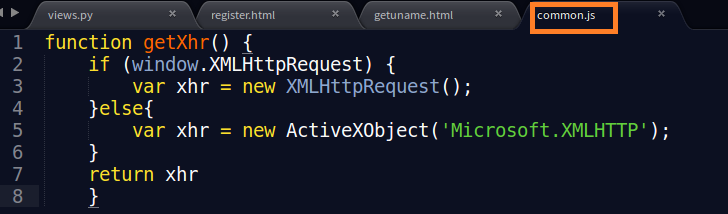
弊端：要求在网页中必须要有<input type=’hidden’ name=’csrfmiddlewaretoken’>标签，可以在网页中任意位置处添加一个form，并且在form中只增加一个{% csrf\_token %}

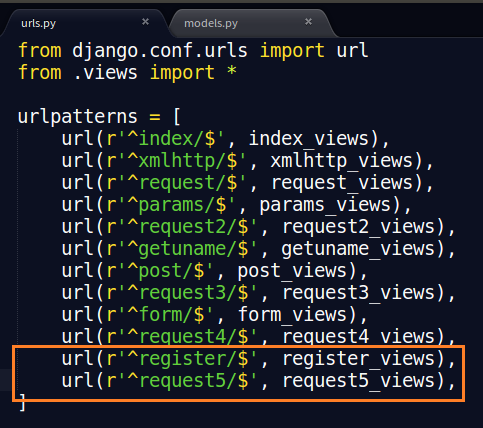
* 1. 必须设置一个请求消息头-Content-Type，AJAX默认将此消息头设置为text/plaiin，导致服务器无法正常接收请求数据。所以必须要将此消息头更新回原来的默认值，语法如下：

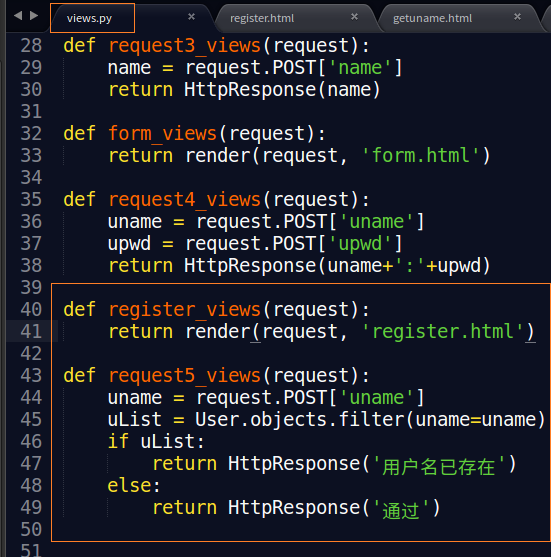
xhr.setResquestHeader（’’Content-Type‘’，’’application/x-www-form-urlencoded’’）；

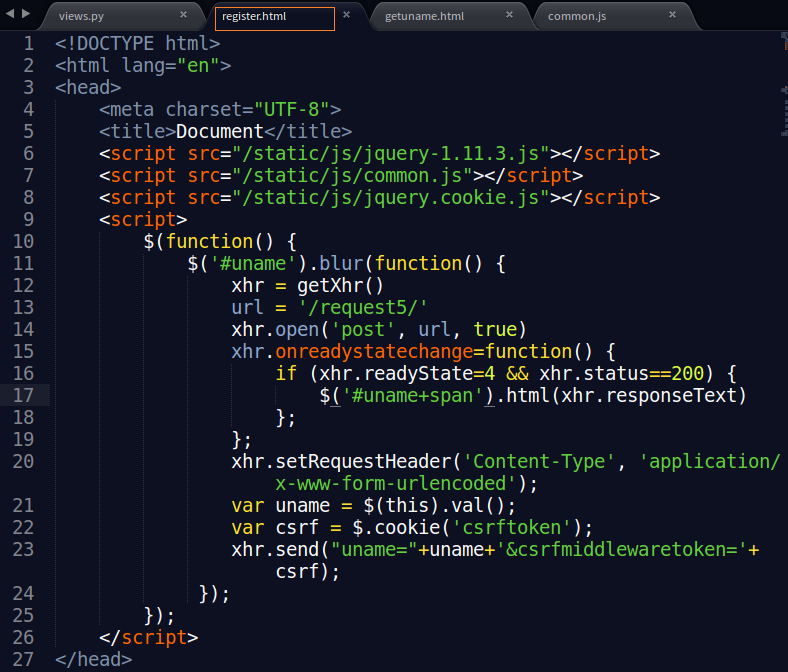
注意，该操作要在xhr.send()之前执行

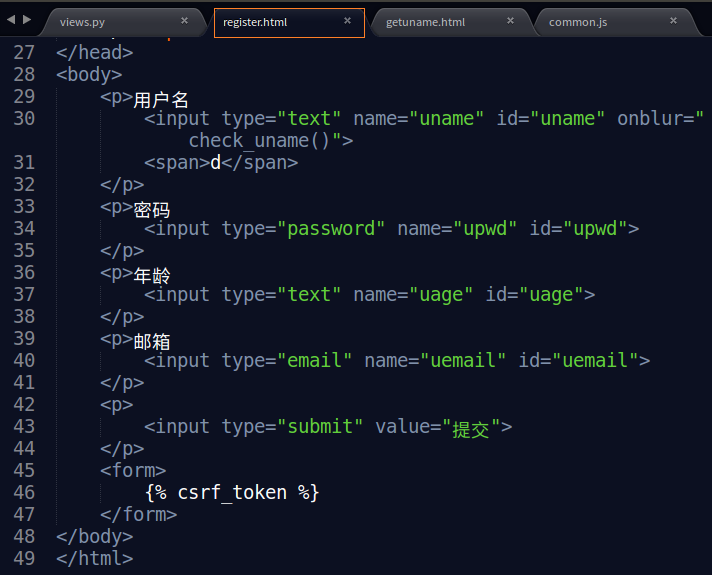
# 同步检测案例











# JSON

1. JSON

JSON：JavaScript Object Notation

JS 对象的表现形式

1. JSON表现
   1. JSON表示单个对象
      1. 使用{}表示单个对象
      2. 在{}中使用键值对key：value的形式表示属性
      3. key必须要用“”括起来
      4. value如果是字符串的话也必须使用“”括起来
   2. JSON表示数组
      1. 使用[]表示一个数组
      2. 数组中允许包含若干的JSON单个对象或字符串

var arr = [“范冰冰”,”李晨”,”范丞丞]；

var arr = [

{对象1}

……

]

## 后台处理JSON

1. 前后台JSON数据交互流程（后—>前）
   1. 后台先获取数据

类型允许为：

元组

列表

字典

QuerySet

* 1. 在后台将数据先转换为符合JSON格式的字符串
  2. 将JSON格式的字符串响应给前端

return HttpResponse（JSON格式字符串）

* 1. 在前端将响应回来的JSON串解析成JSON对象

1. Python（Django）的JSON处理
   1. 元组、列表、字典

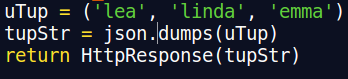
使用Python中的JSON类 就可以完成转换

语法：

import json

jsonStr = json.dumps（元组|列表|字典）

return HttpResponse（jsonStr）



* 1. Django中查询结果集

使用Django提供的序列化类来完成对QuerSet到JSON字符串的转换

from django.core import serializers

str = serializers.serialize（'json’, QuerySet）

return HttpResponse（str）

[{"model": "index.user",

"pk": 1,

"fields": {

"uname": "lea",

"upwd": "lea123",

"uage": 20,

"uemail": [lea@gmail.com](mailto:lea@gmail.com)

}}]

* 1. Django中的单个对象

使用Entry.objects.get()得到的单个对象时不可序列化的，也不能用json.dumps（）去转换

方案一、

使用Entry.objects.filter（）替换Entry.object.get（），替换后可以正常使用序列化

方案二、

将Entry.objects.get()得到的对象转换为字典然后再使用json.dumps（）

class Users（models.Model）:

……

def to\_dic(self):

dic={

‘name’=’self.name,

…

return dic

## 前端处理JSON

将后端响应回来的JSON字符串转换成JSON对象即可

语法：

在JS中

JSON.parse（JSON字符串）

# JQUERY AJAX

在jQuery中提供了对原生AJAX的封装

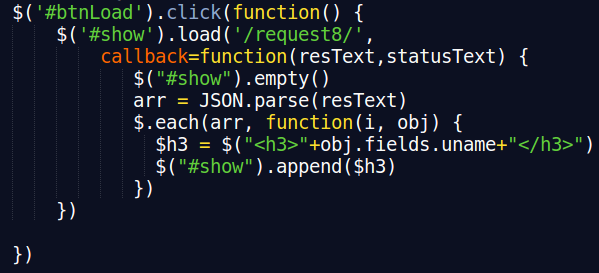
1. $obj.load()

作用：异步加载数据到$obj元素中

语法：$obj.load(url, data, callback)；

url：异步请求的地址

data：传递给请求地址的参数

可以传递普通字符串

‘name=xx&age=xx’

可以传递JSON

{

‘name’:xx,

‘age’:xx,

}

callback：异步请求完毕后要执行的操作（可选）

function（resText，statusText）{

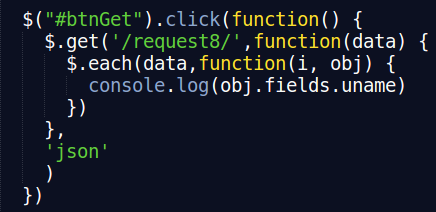
resText：响应数据

statusText：响应状态文本（原因短句）

}

1. $.get（）

作用：使用get方式异步地向服务器发送请求

语法：

$.get(url, data, callback, type)

url：异步请求的地址

data：请求提交的参数

callback：请求成功时的回调函数

type：返回内容的格式类型

1. $.post()

作用：使用post方式异步向服务器发送请求

1. $.ajax（）

作用：所有的操作均可自己定制

语法：$.ajax({传递参数的JSON对象})

JSON对象：

1、url：字符串，要异步请求的地址

2、type：字符串，提交方式，get或post

3、data：JSON对象或字符串

4、dataType：响应回来的数据类型

1、“html”：响应回来的数据是html文本

2、“text”：响应回来的数据是普通文本

3、“script”：响应回来的数据是js脚本

4、“json”：响应回来的数据是JSON对象

5、“xml”：响应回来的数据是xml文档

6、“jsonp”：JSONP格式，跨域时使用

5、success：回调函数，请求和响应成功时才执行的操作

function（data，statusText，jqXHR）{

data：响应回来的数据

statusText：状态文本（可选）

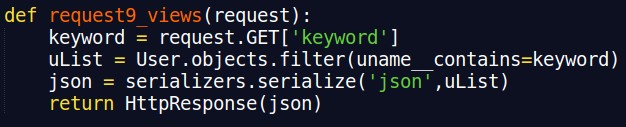
jqXHR：jq中的XMLHttpRequest对象

}

6、error：回调函数，请求或响应产生错误时的回调函数

# 输入同步模糊查询案例

使用$.ajax（）方法





1. s

# sublime插件安装

1. 安装packagecontrol

https://packagecontrol.io/installation

* 1. 复制相关版本代码
  2. 打开sublimeText3控制台（ctrl+`）
  3. 粘贴代码

1. 安装插件
   1. ctrl+shift+p，输入install回车
   2. 输入插件名称
2. 常用插件
   1. Emmet
   2. AutoFilename
   3. jsformat