# WEB开发基础

# 一 HTML

### 1.文档结构和标题标签

<!DOCTYPE html> 告诉浏览器使用什么样的html或者xhtml来解析html文档

<html></html>是文档的开始标记和结束标记。此元素告诉浏览器其自身是一个 HTML 文档，在它们之间是文档的头部<head>和主体<body>。

<head></head>元素出现在文档的开头部分。<head>与</head>之间的内容不会在浏览器的文档窗口显示，但是其间的元素有特殊重要的意义。

<meta charset="UTF-8"> 定义编码类型<单闭合标签>

<title></title>定义网页标题，在浏览器标题栏显示。

<body></body>之间的文本是可见的网页主体内容

### 2.块级元素和行内元素

块级元素：

1.总是从新的一行开始

2.高度、宽度都是可控的

3.宽度没有设置时，默认为100%

4.块级元素中可以包含块级元素和行内元素

行内元素：

1.和其他元素都在一行

2.高度、宽度以及内边距都是不可控的

3.宽高就是内容的高度，不可以改变

4.行内元素只能行内元素，不能包含块级元素

行内块标签：

1.在一行上显示，可以设置大小

例如：img标签

### 3.基本元素标签

<h1></h1>…..<h6></h6>：标题

<p></p>：段落

<hr>: 分割线

<br>：换行

<i></i>：字体倾斜

<strong></strong>：字体加粗标签

&nbsp; ：空格

&copy; ：版权 © 版权符号

&gt; ：大于号

&lt; ：小于号

### 4.img标签的使用

常见图片格式：jpg，gif，bpm，png

img标签用法：

<img src="图片路径/加载图片资源" alt="图片加载资源失败之后替换的文本" title="鼠标悬停的标题" width="x" height="x">

src：可以是一个链接地址或者是本地文件路径等。

alt：是文件加载失败以后在网页显示的文本内容。

title：是鼠标悬停在图片上时显示图片的标题

注意：img 是行内块标签，因此图片是在一行显示，并且可以设置图片大小和宽度

### 5.列表标签和超链接标签

<ol></ol>：有序列表

<ul></ul>：无序列表

<li></li> ：列表项目

<a></a> ：定义超链接，用于从一个页面链接到另一个页面。

<a href="https://foofish.net/" target="\_blank">Python之禅</a>

href：可以是一个连接地址，或者指定一个标签的位置；

例如：<a href="#top">跳转到顶部</a> ：表示跳转到id=top 的位置。

target：属性规定在何处打开被链接文档。只能在 href 属性存在时使用。

\_blank ：在新窗口中打开被链接文档。

\_self ：在被点击时的同一框架中打开被链接文档（默认）。

\_parent ：在父框架中打开被链接文档。

\_top ：在窗口主体中打开被链接文档。

### 6.div标签和span标签

<div></div> ：定义 HTML 文档中的一个分隔区块或者一个区域部分。常用于组合块级元素，以便通过 CSS 来对这些元素进行格式化。

<span></span> ：标签用于对文档中的行内元素进行组合。span可以修饰内容，修饰的方法可以在head里面也可以在span标签内部

例如：<p><span style="color:red;">This is a paragraph</span></p>

### 7.标签分类

块级标签：

特点：独占一行，对高度、宽度、行高以及顶和底边距都可设置的属性值生效；如果不给宽度，块级元素就默认为浏览器的宽度，即就是100%宽；

典型的块级标签有：<div> , h系列，<li>，<dt>，<dd>，<p> , <form>，<ul>，<ol>, <table>, <marquee>（滚动文本）

行内标签：

特点：可以多个标签存在一行，不能直接设置行内标签的高度、宽度、行高以及顶和底边距，完全靠内容撑开宽高！

典型的行内标签有：<span>，<a>，<b>（加粗），<i>，<u>，<em>，<strong>，<label>，<br>

行内块标签：

特点：结合行内和块级的有点，不仅可以对宽高属性值生效，还可以多个标签存在一行显示；

典型的行内标签有：<img>，<input>

上面三种标签的特点可以通过 display属性来将他们互换：

1、块级标签转换为行内标签：display:inline;

2、行内标签转换为块级标签：display:block;

3、转换为行内块标签：display：inline-block;

### 8.表格的基本用法

<table></table>：定义 HTML 表格

一个简单的 HTML 表格包括 table 元素，一个或多个 tr、th 以及 td 元素。

tr 元素定义表格行，th 元素定义表头，td 元素定义表格单元。

更复杂的 HTML 表格也可能包含 caption, col, colgroup, thead, tfoot, tbody 等元素。

table : 表示整个表

thead ：表头

tbody ：表格内容

tr: 表示每一行

td: 表示每一列

colspan ：跨列（单位数字）

rowspan ：跨行（单位数字）

<**table** border=**"1"** width=**"28%"**>  
 <**thead**>  
 <**tr**>  
 <**td** colspan=**"3"**>**三年级五班的语文成绩跟数学成绩**</**td**>  
 </**tr**>  
 <**tr**>  
 <**td**>**姓名**</**td**>  
 <**td**>**学科**</**td**>  
 <**td**>**成绩**</**td**>  
 </**tr**>  
 </**thead**>  
 <**tbody**>  
 <**tr**>  
 <**td** rowspan=**"2"**>**ryan**</**td**>  
 <**td**>**语文**</**td**>  
 <**td**>**98**</**td**>  
 </**tr**>  
 <**tr**>  
 <**td**>**数学**</**td**>  
 <**td**>**95**</**td**>  
 </**tr**>  
 <**tr**>  
 <**td** rowspan=**"2"**>**cherry**</**td**>  
 <**td**>**语文**</**td**>  
 <**td**>**80**</**td**>  
 </**tr**>  
 <**tr**>  
 <**td**>**数学**</**td**>  
 <**td**>**100**</**td**>  
 </**tr**>  
 </**tbody**>  
</**table**>



### 9.表单控件的使用

<form></form> : 标签标签用于创建供用户输入的 HTML 表单。

form 元素包含一个或多个表单元素，比如：

* button
* input
* keygen
* object
* output
* select
* textarea

语法： <form action="表单数据提交的地址" method="表单数据提交的方法 post/get">...</form>

<from>标签就像一个容器，它里面会有很多的获取用户输入的标签（表单控件）。

<**form** action=**"https://www.baidu.com"** method=**"get"**>  
 <!--label 中得for属性对应的值，指向了input标签中的id-->  
 <**label** for=**"name"**>**姓名:**</**label**>  
 <**input** type=**"text"** name=**"name"** id=**"name"**>  
 <**br**>  
 <**label** for=**"password"**>**密码:**</**label**>  
 <**input** type=**"password"** name=**"password"** id=**"password"**>  
 <**br**>  
 <!--input 表单控件类型设置type=submit -->  
 <**input** type=**"submit"** value=**"提交"** >  
</**form**>



点击提交按钮以后，会把name=ryan和password=1234 传给<https://www.baidu.com>

结果为：https://www.baidu.com/?name=ryan&password=1234

### 10.input表单的类型和按钮

<1> 表单类型

type:

text 文本输入框

password 密码输入框

radio 单选框

checkbox 多选框

submit 提交按钮

button 按钮(需要配合js使用.) button和submit的区别？

file 提交文件：form表单需要加上属性enctype="multipart/form-data"

上传文件注意两点：

请求方式必须是post

enctype="multipart/form-data"

<2> 表单属性

name: 表单提交项的键.

注意和id属性的区别：name属性是和服务器通信时使用的名称；

而id属性是浏览器端使用的名称，该属性主要是为了方便客户端编程，而在css和javascript中使用的

value: 表单提交项的值.对于不同的输入类型，value 属性的用法也不同：

type="button", "reset", "submit" - 定义按钮上的显示的文本

type="text", "password", "hidden" - 定义输入字段的初始值

type="checkbox", "radio", "image" - 定义与输入相关联的值

checked: radio 和 checkbox 默认被选中

readonly: 只读. text 和 password

disabled: 对所用input都好使.

method 方法规定如何发送表单数据（form-data）（表单数据会被发送到在 action 属性中规定的页面中）。

可以通过以下方式发送 form-data ：

* 以 URL 变量 (使用 method="get") 的形式来发送
* 以 HTTP post (使用 method="post") 的形式来发送

关于 "get" 方法的注释：

* 该方法将表单数据以名称/值对的形式附加到 URL 中
* 该方法对于用户希望加入书签的表单提交很有用
* 在 URL 中放置的数据量是有限制的（不同的浏览器有差别），所以无法确保所有表单数据得到正确地传输
* 绝不要使用 "get" 方法来传输敏感信息！（密码或其他敏感信息在浏览器地址栏中是可见的）

关于 "post" 方法的注释：

* 该方法以 HTTP post 事务的方式来传递表单数据
* 无法将通过 "post" 方法提交的表单加入书签
* 与 "get" 相比，"post" 方法更健壮更安全，而且 "post" 没有容量限制

# 二 CSS

### 1.CSS介绍

CSS：主要用于修饰页面

     - 层叠样式表（Cascading Style Sheet）

CSS 的优势

* 1.内容与表现分离
* 2.网页的表现统一，容易修改
* 3.丰富的样式，使页面布局更加灵活
* 4.减少网页的代码量，增加网页浏览器速度，节省网络带宽
* 5.运用独立页面的css,有利于网页被搜索引擎收录

CSS基础选择器

* 标签选择器
* 类选择器
* ID选择器

### 2.CSS引入方式

网页中引用CSS样式

* 内联样式
* 内部样式表
* 外部样式表

  - 链接式

  - 导入式

1.内联样式（内嵌式）

<**h1** style=**"***font-size*: 50**px**;*color*:**yellow**;**"**>**路飞学城**</**h1**>

直接在标签内对样式进行修饰

2.内部样式表（内链式）

<**style** type=**"text/css"**>  
 /\*内链式\*/  
 h1{  
 *font-size*: 30**px**;  
 *color*: **red**;  
 }  
</**style**>

3.外部样式表

3.1 链接式

<**link** rel=**"stylesheet"** type=**"text/css"** href=**"./css\_样式/index.css"/**>

href：文件路径；

rel ：使作外部样式表

type：文件类型

index.css内容：

h2{  
 *color*: **chartreuse**;  
 *font-size*: 50**px**;  
}

3.2 导入式

<**style** type=**"text/css"**>  
 /\*导入式\*/  
 @import url('./css\_样式/index.css');  
</**style**>

内嵌式的优先级高于内链式

@import url(other.css)

注意： @import url()必须写在文件最开始的位置。

链接式与导入式的区别

1、<link/>标签属于XHTML,@import是css的属性

2、使用<link/>链接的css文件先加载到网页当中，再进行编译显示

3、使用@import导入的css文件，客户端显示HTML结构，再把CSS文件加载到网页当中

4、@import是属于CSS2.1特有的，对于不兼容CSS2.1的浏览器来说就是无效的

### 3.CSS基础选择器-通配符选择器

通配符选择器，匹配所有样式，清除默认个事

<**style** type=**"text/css"**>  
 /\*通配符选择器 ，匹配所有的样式，清除默认格式\*/  
 \*{  
 *padding*: 0;  
 *margin*: 0;  
 *color*: **red**; /\*字体等的颜色全部为红色\*/  
 }  
</**style**>

### 4.CSS基础选择器-ID选择器

ID选择器：用 “#” 加上 id 的值

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**CSS基础选择器**</**title**>  
 <**style** type=**"text/css"**>  
 /\*ID选择器\*/  
 #container{  
 *width*: 1000**px**;  
 *background-color*: **aqua**;  
 *margin*: 0 **auto**; /\*让内容在中间显示\*/  
 }  
 </**style**>  
</**head**>  
<**body**>  
 <**div** id=**"container"**>  
 <**h1**>**我爱中国**</**h1**>  
 </**div**>  
</**body**>  
</**html**>

### 5.CSS基础选择器-标签选择器

标签选择器，直接使用标签名称；设置了该标签选择器，那么所有的该名称标签都会被修饰！

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**CSS基础选择器**</**title**>  
 <**style** type=**"text/css"**>  
 /\*标签选择器\*/  
 h1{  
 *color*: **yellowgreen**;  
 *font-size*: 20**px**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
 <**div** id=**"container"**>  
 <**h1**>**我爱中国**</**h1**>  
 </**div**>  
</**body**>  
</**html**>

### 6.CSS基础选择器-子代选择器

子代选择器，用“>”，如下：指 <ul>标签下的<li>标签都被该样式修饰。该方法常用语：.class>li ,指某个class下面的li标签都被该样式修饰

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**CSS基础选择器**</**title**>  
 <**style** type=**"text/css"**>  
 /\*子代选择器\*/  
 ul>li{  
 *color*: **red**;  
 }  
 /\*这个也属于子代选择器\*/  
 #container>ul>li{  
 *background-color*: **yellowgreen**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
 <**div** id=**"container"**>  
 <**ul**>  
 <**li**>**我**</**li**>  
 <**li**>**爱**</**li**>  
 <**li**>**中**</**li**>  
 <**li**>**国**</**li**>  
 </**ul**>  
 </**div**>  
</**body**>  
</**html**>

### 7.CSS基础选择器-后代选择器

后代选择器就是，某个标签下的所有标签都使用该样式。例如下列：就是标签<ul>下的所有<a>标签都用该样式

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**CSS基础选择器**</**title**>  
 <**style** type=**"text/css"**>  
 /\*后代选择器\*/  
 ul a{  
 *color*: **red**;  
 *background-color*: **yellowgreen**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
 <**div** class=**"last"**>  
 <**ul**>  
 <**li**><**a** href=**"#"**>**我爱你中国**</**a**></**li**>  
 <**li**><**a** href=**"#"**>**SH**</**a**></**li**>  
 </**ul**>  
 </**div**>  
</**body**>  
</**html**>

### 8.CSS基础选择器-类选择器

某个标签定义了一个class=“content”，那就使用.content{样式去修饰}。该class下的所有内容都使用该样式。

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**CSS基础选择器**</**title**>  
 <**style** type=**"text/css"**>  
 /\*类选择器\*/  
 .content{  
 *width*: 400**px**;  
 *background-color*: **yellowgreen**;  
 *margin*: 0 **auto**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
 <**div** class=**"content"**>  
 <**ul**>  
 <**li**><**a** href=**"#"**>**我爱你中国**</**a**></**li**>  
 <**li**><**a** href=**"#"**>**SH**</**a**></**li**>  
 </**ul**>  
 </**div**>  
</**body**>  
</**html**>

### 9.CSS高级选择器-群组选择器

群组选择器，是一群属性的集合。

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**CSS高级选择器**</**title**>  
 <**style** type=**"text/css"**>  
 /\*群组选择器\*/  
 .title,.content,.footer{  
 *width*: 968**px**;  
 *margin*: 0 **auto**;  
 *background-color*: #BFBFBF;  
 *border*: 1**px solid red**;  
 }  
 #title\_p1,#footer\_p1{  
 *font-size*: 20**px**;  
 *color*: **greenyellow**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
 <**div** class=**"title"**>  
 <**p** class=**"p1"** id=**"title\_p1"**>**我是一个段落**</**p**>  
 </**div**>  
 <**div** class=**"content"**>  
 <**h3**>**我是三级标题1**</**h3**>  
 <**a** href=**"https://www.baidu.com"**>**百度一下**</**a**>  
 <**p**>**我是另一个段落**</**p**>  
 </**div**>  
 <**div** class=**"footer"** id=**"footer\_p1"**>  
 <**p**>**我是底部**</**p**>  
 </**div**>  
</**body**>  
</**html**>

### 10.CSS高级选择器-交集选择器

交集选择器：由两个选择器构成，选中二者范围的交集，两个选择器之间不能有空格，第一个必须是标签选择器，第二个必须是类选择器或者ID选择器

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**CSS高级选择器**</**title**>  
 <**style** type=**"text/css"**>  
 /\*交集选择器\*/  
 p.title\_p1{  
 *color*: **red**;  
 }

p#title\_p1{  
 *font-size*: 30**px**;  
 *color*: **darkmagenta**; /\*覆盖上面的color\*/  
 }  
 p#footer\_p1{  
 *color*: **greenyellow**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
 <**div** class=**"title"**>  
 <**p** class=**"title\_p1"** id=**"title\_p1"**>**我是一个段落**</**p**>  
 </**div**>  
 <**div** class=**"footer"** >  
 <**p** id=**"footer\_p1"**>**我是底部**</**p**>  
 </**div**>  
</**body**>  
</**html**>

### 11.CSS高级选择器-毗邻选择器

自己得出的结论:毗邻选择器，也要说是兄弟选择器，只能是同级标签

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**CSS高级选择器**</**title**>  
 <**style** type=**"text/css"**>  
 /\*毗邻选择器\*/  
 /\*找到所有紧挨在p3后面的第一个p标签，设置字体颜色为红色。\*/  
 h3+p {  
 *color*: **red**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
 <**div** class=**"content"**>  
 <**h3**>**我是三级标题1**</**h3**>  
 <**p**>**中国**</**p**>  
 <**p**>**最美**</**p**>  
 </**div**>  
</**body**>  
</**html**>

### 12.CSS高级选择器-弟弟选择器

下面例子指的是，找到h3 下面的所有同级p标签进行修饰。

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**CSS高级选择器**</**title**>  
 <**style** type=**"text/css"**>  
 h3~p{  
 *color*: **red**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
 <**div**>  
 <**h3**>**我是标题3**</**h3**>  
 <**p**>**中国**</**p**>  
 <**p**>**最美**</**p**>  
 </**div**>  
</**body**>  
</**html**>

### 13.CSS选择器-属性选择器

除了HTMLyu元素的ID属性和CLASS属性外，还可以根据HTML元素的特定属性选择元素。

意思就是说：可以自己设定一个属性名称，然后自己去设置样式，如下title就是设置的属性元素。

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**CSS高级选择器**</**title**>  
  
 <**style** type=**"text/css"**>  
  
 /\*属性选择器\*/  
 /\*找到所有title的选择器\*/  
 [title]{  
 *color*: **red**;  
 }  
  
 /\*找到所有title开头是hello的选择器\*/  
 [title^=hello]{  
 *font-size*: 20**px**;  
 *background-color*: **greenyellow**;  
 }  
  
 /\*找到所有title结尾是china的选择器\*/  
 [title$=china]{  
 *font-size*: 40**px**;  
 *background-color*: #868686;  
 }  
 </**style**>  
</**head**>  
<**body**>  
 <**div** title=**"hello word"**>  
 <**p** class=**"p1"** id=**"title\_p1"**>**我是一个段落**</**p**>  
 </**div**>  
 <**div** title=**"hello china"**>  
 <**p**>**我爱中国**</**p**>  
 </**div**>  
</**body**>  
</**html**>

### 14.CSS选择器总结

总结：

基础选择器：

1.标签选择器 div

2.类选择器 .div1

3.id选择器 #box

4.通配符选择器 \*

高级选择器：

1.群组选择器：中间用 ,

2.交集选择器，选择器之间不能有空格，第一个标签必须是标签选择器，第二个标签可以是ID选择器或者是class选择器

3.后代选择器，选择器之间用空格

4.子代选择器 >

5.毗邻选择器 +

6.弟弟选择器 ~

7.属性选择器 [ ]

### 15.CSS选择器优先级

行内样式 > 内部样式 > 外部样式

ID选择器 > 类选择器 > 标签选择器

样式表中的特殊性描述了不同规则的相对权重，它的基本规则是：

1 内联样式表的权值最高(行内样式) style=""－－－－－－－－－－－－－ 1000；

2 统计选择符中的ID属性个数。 #id －－－－－－－－－－－－－ 100

3 统计选择符中的CLASS属性个数。 .class －－－－－－－－－－－－－ 10

4 统计选择符中的HTML标签名个数。 p －－－－－－－－－－－－－ 1

.div3{  
 *color*: **yellowgreen!important**;  
}

！important 声明以后高于一切。如果都有important，那么比较优先级，如果优先级一样，那么后来者居上。

继承是CSS的一个主要特征，它是依赖于祖先-后代的关系的。继承是一种机制，它允许样式不仅可以应用于某个特定的元素，还可以应用于它的后代

继承是CSS重要的一部分，我们甚至不用去考虑它为什么能够这样，但CSS继承也是有限制的。有一些属性不能被继承，如：border, margin, padding, background等。

### 16.CSS 伪类选择器一

1）a:link :没有访问的超链接a标签样式;

2）a:visited :访问过的超链接a标签样式;

3）a:hover : 鼠标悬浮在元素上应用样式：

4）a:active：鼠标点击瞬间的样式;

5）input:focus : input输入框获取焦点时样式;

举一列说明即可：

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**伪类选择器的使用**</**title**>  
 <**style** type=**"text/css"**>  
 a:hover{  
 *color*: **coral**;  
 *font-size*: 30**px**;  
 }  
 input:focus{  
 *background-color*: **red**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
 <**ul**>  
 <**li**><**a** href=**"#"**>**百度一下**</**a**></**li**>  
 </**ul**>  
<**form** action=**""**>  
 <**input** type=**"text"**>  
</**form**>  
</**body**>  
</**html**>

### 17.补充-伪类选择器（nth-child等）

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**2-伪类选择器一**</**title**>  
 <**style** type=**"text/css"**>  
 /\*选中第一个元素\*/  
 .box ul li:first-child{  
 *color*: **red**;  
 *font-size*: 30**px**;  
 }  
 /\*选中最后一个元素\*/  
 .box ul li:last-child{  
 *color*: **green**;  
 *font-size*: 30**px**;  
 }  
 /\*选中指定的元素，从1开始计数\*/  
 .box ul li:nth-child(2){  
 *color*: **fuchsia**;  
 *font-size*: 40**px**;  
 }  
 /\*n 表示选中所有，从0开始(0的时候表示没有选中)\*/  
 /\*2n 表示所有的偶数\*/  
 /\*2n-1 表示所有的奇数\*/  
 .box ul li:nth-child(2n-1){  
 *color*: **purple**;  
 *font-size*: 50**px**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
 <**div** class=**"box"**>  
 <**ul**>  
 <**li** class=**"item1"**>  
 **1** <**a** href=**"#"**>**张三**</**a**>  
 </**li**>  
 <**li** class=**"item2"**>  
 **2** <**a** href=**"#"**>**李四**</**a**>  
 </**li**>  
 <**li** class=**"item3"**>  
 **3** <**a** href=**"#"**>**ryan**</**a**>  
 </**li**>  
 <**li** class=**"item4"**>  
 **4** <**a** href=**"#"**>**cherry**</**a**>  
 </**li**>  
 <**li** class=**"item5"**>  
 **5** <**a** href=**"#"**>**Lucky**</**a**>  
 </**li**>  
 </**ul**>  
 </**div**>  
</**body**>  
</**html**>

知识点：

1.nth-child 可以选择指定的元素

### 18.CSS伪元素选择器

在指定标签的前后或者本身内容进行修饰或加内容。

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**伪元素选择器**</**title**>  
 <**style** type=**"text/css"**>  
 /\*被修饰的标签的字体变成30px\*/  
 p:first-letter{  
 *font-size*: 30**px**;  
 }  
 /\*给被修饰的标签的内容前面加内容和颜色\*/  
 p:before{  
 *content*: "cherry,";  
 *color*: **red**;  
 }  
 /\*给被修饰的标签的内容后面加内容和颜色\*/

/\*在指定选择器之后添加内容（一定要结合content属性）使用非常频繁，与布局有很大的关联(清除浮动\*/  
 p:after{  
 *content*: "love ";  
 *color*: **black**;  
 *font-size*: 35**px**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
<**p**>**爱你哟**</**p**>  
</**body**>  
</**html**>

### 19.CSS的继承性和层叠性(优先级)

**继承性：**

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**5-CSS中的继承性**</**title**>  
 <**style** type=**"text/css"**>  
 .box{*color*: **red**;*font-size*: 20**px**;*background-color*: **blue**;}  
 </**style**>  
</**head**>  
<**body**>  
<**div** class=**"box"**>  
 <**p**>**继承**</**p**>  
</**div**>  
</**body**>  
</**html**>

继承：诶父级设置一些属性，子级继承了父级的该属性，这就是我们的css中的继承  
有一些属性是可以继承下来的：color,font-\* , text-\* , line-\*  
像一些盒子元素，定位的元素（浮动，绝对定位，固定定位）不能继承

层叠性-**选择器的优先级**

内联样式权重为：1000

id选择器权重为：100

类选择器权重为: 10

元素选择器权重为: 1

当权重一样的时候后面的样式覆盖前面的样式；（注意：当继承来的样式，权重为0 ，选中的样式大于继承的样式）

总结：先看有没有被选中，如果选中了，就根据权重显示样式。如果没有被选中权重为0 ；如果属性都是杯继承下来的权重为0，但是样式根据“就近原则”（谁描述的近就显示谁）进行显示。

注意：还有一种不讲道理的!import方式来强制让样式生效，但是不推荐使用。因为大量使用!import的代码是无法维护的。

### 20.CSS important

内容………………

### 21.字体样式

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**字体样式**</**title**>  
 <**style** type=**"text/css"**>  
 p{  
 /\*设置字体大小\*/  
 *font-size*: 20**px**;  
 /\*设置字体为斜体：  
 normal: 默认值.  
 oblique: 浏览器会显示一个倾斜的字体样式。  
 italic : 浏览器会显示一个斜体的字体样式。  
 inherit: 规定应该从父元素继承字体样式。  
 \*/  
 *font-style*: **inherit**;  
 /\*!\*设置字体的粗细\*!\*/  
 *font-weight*: 700;  
 /\*!\*设置字体类型\*!\*/  
 /\*font-family: Algerian;\*/  
 /\*!\*设置字体颜色\*!\*/  
 *color*: **firebrick**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
<!--默认字体大小是：16px=1em 0.75em=12px-->  
<**p**>**这是一个p标签**</**p**>  
</**body**>  
</**html**>

### 22.文本样式

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**文本样式**</**title**>  
 <**style** type=**"text/css"**>  
 .box1{  
 *width*: 500**px**;  
 *height*: 200**px**;  
 *background-color*: **yellowgreen**;  
 *color*: **red**;  
 /\*设置字体下划线：underline；删除线:line-through,上横线：overline\*/  
 *text-decoration*: **underline**;  
 *text-decoration-color*: **black**;  
 /\*设置鼠标属性\*/  
 *cursor*: **pointer**;  
 /\*设置文字往右移动20px，首行缩进\*/  
 *text-indent*: 20**px**;  
 /\*设置对齐方式,  
 对齐方式：center，left，right、  
 两端对齐:justify(只对英文有效)\*/  
 *text-align*: **center**;  
 /\*设置字体阴影:  
 第一个值：设置对象的阴影水平偏移值。可以为负值  
 第二个值：设置对象的阴影垂直偏移值。可以为负值  
 第三个值：从来设置对象的阴影模糊值。不允许负值  
 第四个值：设置对象的阴影的颜色。  
 \*/  
 *text-shadow*: 5**px** 0 5**px black**;  
 /\*行高:它决定了元素中文本内容的高度，height则是定义元素自身的高度。\*/  
 /\*当行高等于盒子的高度时，那么会实现文本垂直方向居中\*/  
 *line-height*: 200**px**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
 <**div** class=**"box1"**>**我爱你中国**</**div**>  
 <**a** href=**"#"**>**我爱你中国**</**a**>  
</**body**>  
</**html**>

### 23.背景属性

background: 背景 照片位置 图片展开方式 滚动/固定 position/图片长,宽

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**背景属性**</**title**>  
 <**style** type=**"text/css"**>  
 #img{  
 *width*: 1000**px**;  
 *height*: 1000**px**;  
 /\*设置背景颜色\*/  
 /\*background-color: yellowgreen;\*/  
 /\*设置背景图片\*/  
 /\*background-image: url(./images/1.png);\*/  
 /\*background-repeat:属性定义了图像的平铺模式。  
 repeat 默认。背景图像将在垂直方向和水平方向重复。  
 repeat-x 背景图像将在水平方向重复。  
 repeat-y 背景图像将在垂直方向重复。  
 no-repeat 背景图像将仅显示一次。  
 inherit 规定应该从父元素继承 background-repeat 属性的设置。  
 \*/  
 /\*background-repeat: no-repeat;\*/  
 /\*background-attachment: fixed;\*/  
 /\*background-position: -50px -50px;\*/  
  
 /\*简写  
 注意：此时上面的background 的一些值会被覆盖掉，比如颜色  
 此时可以有两种写法，可以在background里面直接写颜色  
 或者重新定义一个颜色 \*/  
 /\*background: 背景 照片位置 图片展开方式 滚动/固定 position/图片长,宽  
 background: yellowgreen url(./images/1.png) no-repeat fixed right/300px 300px;  
 /\*background-color: yellowgreen;\*/  
 }  
 </**style**>  
</**head**>  
<**body**>  
 <**div** id=**"img"**></**div**>  
</**body**>  
</**html**>

### 24.盒模型定义

所有HTML元素可以看作盒子，在CSS中，“box model”这一术语是用来设计和布局时使用。

CSS模型本质上是一个盒子，分装周围的HTML元素，它包括：边距，边框，填充，和实际内容。



* **Margin(外边距)** - 清除边框外的区域，外边距是透明的。
* **Border(边框)** - 围绕在内边距和内容外的边框。
* **Padding(内边距)** - 清除内容周围的区域，内边距是透明的。
* **Content(内容)** - 盒子的内容，显示文本和图像。

最终元素的总宽度计算公式是这样的：

总元素的宽度=宽度+左填充+右填充+左边框+右边框+左边距+右边距

元素的总高度最终计算公式是这样的：

总元素的高度=高度+顶部填充+底部填充+上边框+下边框+上边距+下边距

### 25.盒模型-border

边框样式：border-style（上下左右一样）

border-top/right/bottom/left 上左下右边框属性。

none： ：无边框。

dotted ：点状虚线边框。

dashed ：矩形虚线边框。

solid : 实线边框。

double ：双线边框

hidden : 与none相同，应用于表解决边框冲突

边框颜色：border-color

边框粗细：border-width

- 关键字: thin、medium、thick

- 像素值：px

### 26.盒模型-margin(外边距)相遇

当两个垂直外边距相遇时会形成一个外边框，称为 “外边距合并”！  
所有毗邻的两个或更多盒元素的margin将会合并为一个margin共享之。毗邻的定义为：同级或者嵌套的盒元素，并且它们之间没有非空内容、Padding或Border分隔。  
  
注意：父子级盒子嵌套设置外边距会形成外边距合并，影响布局，  
  
解决办法：  
 1. 父级或子元素使用浮动或者绝对定位absolute 浮动或绝对定位不参与margin的折叠  
 2. 父级overflow:hidden; 溢出隐藏  
 3. 父级设置padding（破坏非空白的折叠条件）  
 4. 父级设置border

margin:10px 5px 15px 20px;-----------上 右 下 左

margin:10px 5px 15px;----------------上 右左 下

margin:10px 5px;---------------------上下 右左

margin:10px; ---------------------上右下左

### 27.文档流介绍

标准文档流定义：在web页面制作，是个“流”，文档流指的是元素排版布局过程中，元素会自动从左往右，自上而下的流式布局。

标准文档流的微观现象：

1.空白折叠现象：无论多少空格都会折叠称为一个空格（除 &nbsp; 除外）

2.高矮不起，底边对齐

3.自动换行，一行写不满，换行写

### 28.display用法

标签元素有：块标签，行内块标签，行内标签。

上面三种标签的特点可以通过 display属性来将他们互换：

1、块级标签转换为行内标签：display:inline;

2、行内标签转换为块级标签：display:block;

3、转换为行内块标签：display：inline-block;

display :none; 不显示，位置以及样式全部隐藏。

visibility: hidden ; 隐藏标签元素，占位置。

inline-block默认的空格符就是标签与标签之间的空隙造成的。

(1) 我们可以通过margin:-3px来解决，但是

1.我们布局肯定很多元素，不可能每个都添加margin负这样维护成本太大了

2.我们线上代码如果压缩，那么我们就不存在哪个4px的问题了，那么我们的margin负就回造成布局混乱！

(2)我们可以给几个标签加一个父级div，然后：

div{word-spacing: -7px;}

### 29.浮动产生的效果

所谓的文档流，指的是元素排版布局过程中，元素会自动从左往右，从上往下的流式排列。

脱离文档流，也就是将元素从普通的布局排版中拿走，其他盒子在定位的时候，会当做脱离文档流的元素不存在而进行定位。

只有绝对定位absolute和浮动float才会脱离文档流。

定位处理：

浮动：float

属性值:

left:元素向左浮动;从左向右依次排列

right：元素向右浮动；从右向左一次排列

none：默认值，元素不浮动。从上往下排列

inherit：规定应该从父元素继承 float 属性的值。

浮动产生的效果：

浮动可以是元素按指定位置排列，直到遇到父元素的边界或另外一个元素的边界停止。

定位：position

堆叠：z-index

### 30.浮动元素脱标

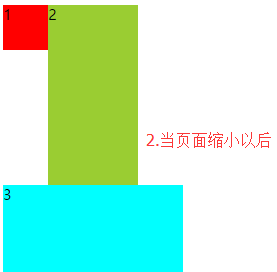
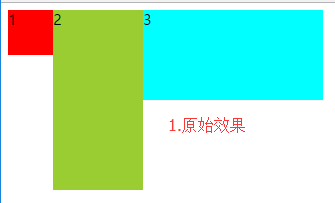
脱标含义：脱离了标准文档流。

脱标带来的隐藏属性：所有的标签一旦设置浮动，都不区分行内元素和块状元素，他们都能够设置宽高

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**11-浮动的元素脱标**</**title**>  
 <**style** type=**"text/css"**>  
 \*{*padding*: 0;*margin*: 0;}  
 .box1{  
 *width*: 100**px**;  
 *height*: 100**px**;  
 *background-color*: **red**;  
 *float*: **left**;  
 }  
 .box2{  
 *width*: 200**px**;  
 *height*: 200**px**;  
 *background-color*: **yellowgreen**;  
 }  
 span{  
 *background-color*: #2459A2;  
 *float*: **left**;  
 *width*: 300**px**;  
 *height*: 50**px**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
<!--脱标：脱离了标准文档流  
 box1盒子设置了浮动，脱离了标准文档流，此时box2盒子就是标准文档流中的第一个盒子，所以就渲染到了左上方  
-->  
<**div** class=**"box1"**>**box1**</**div**>  
<**div** class=**"box2"**>**box2**</**div**>  
<!--span标签设置了 float 不需要转成块级元素 也能够设置设置宽高-->  
<**span**>**span标签**</**span**>  
<**span**>**span标签**</**span**>  
</**body**>  
</**html**>

### 31.浮动元素相互贴靠

设置浮动以后相互贴靠的原理是（自己的理解）：如果document都不设置浮动，那么，html按照标准文档流去展示。但是如果全部元素或者相邻的元素都设置了浮动，那么，可以看成所有的浮动元素，在浮动元素这个层级去进行标准文档流的形式去展示。



<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**12-浮动元素互相贴靠**</**title**>  
 <**style** type=**"text/css"**>  
 .box1{*width*: 50**px**;*height*: 50**px**;*background-color*: **red**;*float*: **left**;}  
 .box2{*width*: 100**px**;*height*: 200**px**;*background-color*: **yellowgreen**;*float*: **left**;}  
 .box3{*width*: 200**px**;*height*: 100**px**;*background-color*: **aqua**;*float*: **left**;}  
 </**style**>  
</**head**>  
<**body**>  
 <!--紧挨着的几个元素都设置了浮动，那相当于浮动元素类似于标准文档流格式去展示-->  
 <**div** class=**"box1"**>**1**</**div**>  
 <**div** class=**"box2"**>**2**</**div**>  
 <**div** class=**"box3"**>**3**</**div**>  
</**body**>  
</**html**>

### 32.浮动元素字围效果

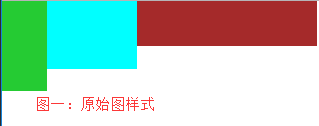
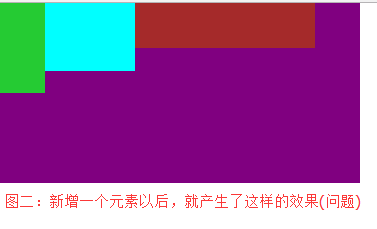


<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**13-浮动元素字围效果**</**title**>  
 <**style** type=**"text/css"**>  
 \*{*margin*: 0;*padding*: 0;}  
 body{*width*: 200**px**;*height*: 200**px**;*border*: 1**px solid** #2b542c}  
 div{*float*: **left**;}  
 img{*width*: 150**px**;*height*: 150**px**}  
 </**style**>  
</**head**>  
<**body**>  
 <!--所谓字围效果：  
 当div浮动，p不浮动  
 div挡住了p，div的层级提高了，但是p中的文字不会被遮盖。此时就形成了字围效果  
 关于浮动一定要注意:浮动这个元素，我们一定要遵循一个原则：永远不是一个盒子单独浮动，要浮动就要一起浮动  
 -->  
 <**div**><**img** src=**"./3.png"** alt=**""**></**div**>  
 <**p**>**123字围效果字围效果字围效果字围效果字围效果字围效果字围效果字围效果**</**p**>  
</**body**>  
</**html**>

### 33.浮动元素紧凑效果

如果一个浮动元素没有设置宽(width) 高(height)，那么久自动收缩成为文字的宽度；

### 34.浮动带来的问题

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**15-浮动带来的问题**</**title**>  
 <**style** type=**"text/css"**>  
 \*{*padding*: 0;*margin*: 0;}  
 /\*子元素设置了浮动，一般父盒子不设置高度\*/  
 .father{*width*: 400**px**;}  
 .box1{*width*: 50**px**;*height*: 100**px**;*background-color*: #25CB33;}  
 .box2{*width*: 100**px**;*height*: 75**px**;*float*: **left**;*background-color*: **aqua**;}  
 .box3{*width*: 200**px**;*height*: 50**px**;*float*: **left**;*background-color*: **brown**;}   
 .mother{*width*: 400**px**;*height*: 200**px**;*background-color*: **purple**;}  
 </**style**>  
</**head**>  
<**body**>  
<**div** class=**"father"**>  
 <**div** class=**"box1"**></**div**>  
 <**div** class=**"box2"**></**div**>  
 <**div** class=**"box3"**></**div**>  
</**div**>  
<**div** class=**"mother"**></**div**>  
</**body**>  
</**html**>

### 35.清除浮动1：给父盒子设置高度

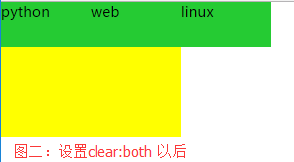


<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**16-清除浮动1-给父盒子设置高度**</**title**>  
 <**style** type=**"text/css"**>  
 \*{*margin*: 0;*padding*: 0;}  
 div{*width*: 400**px**;*height*: 40**px**;}  
 ul{*list-style*: **none**;}  
 div ul li{*float*: **left**;*width*: 100**px**;*height*: 40**px**;*background-color*: #25CB33;}  
 .box2{*width*: 200**px**;*height*: 100**px**;*background-color*: **yellow**;}  
 </**style**>  
</**head**>  
<**body**>  
<**div** class=**"box1"**>  
 <**ul**>  
 <**li**>**python**</**li**>  
 <**li**>**web**</**li**>  
 <**li**>**linux**</**li**>  
 </**ul**>  
</**div**>  
<**div** class=**"box2"**></**div**>  
</**body**>  
</**html**>

图一产生的原因：其实此时图一里面的box1 的高度为0；虽然box1 里面的li标签设置了高度，但是，li标签设置了浮动，设置了浮动元素以后，li标签就脱标（脱离标准文档流），脱离标准文档流以后不在当前页面占据位置，就不会填充父元素的高度。所以，box2 就会顶上去。

图二的结果产生原因：给box1 设置了高度以后，就在标准文档流中撑开了，因此，就按照标准文档流去展示。

### 36.清除浮动2：clear:both



<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**16-清除浮动2-clear:both**</**title**>  
 <**style** type=**"text/css"**>  
 \*{*margin*: 0;*padding*: 0;}  
 div{*width*: 400**px**;}  
 ul{*list-style*: **none**;}  
 div ul li{*float*: **left**;*width*: 100**px**;*height*: 50**px**;*background-color*: #25CB33;}  
 .box2{*width*: 200**px**;*height*: 100**px**;*background-color*: **yellow**;}  
 .clear{*clear*: **both**}  
 </**style**>  
</**head**>  
<**body**>  
<**div** class=**"box1"**>  
 <**ul**>  
 <**li**>**python**</**li**>  
 <**li**>**web**</**li**>  
 <**li**>**linux**</**li**>  
 </**ul**>  
 <!--给浮动元素最后面加一个空的div并且该元素不浮动，然后设置clear:both(清除别人对我的浮动影响) -->  
 <**div** class=**"clear"**></**div**>  
</**div**>  
<**div** class=**"box2"**></**div**>  
</**body**>  
</**html**>

给浮动元素的最后一个元素加一个空的块级元素，且该元素不浮动，设置clear：both；缺点：结构冗余；

### 37.清除浮动3：伪元素清除法（常用）

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**17-清除浮动3-伪元素清除法(常用)**</**title**>  
 <**style** type=**"text/css"**>  
 \*{*margin*: 0;*padding*: 0;}  
 div{*width*: 400**px**;}  
 ul{*list-style*: **none**;}  
 div ul li{*float*: **left**;*width*: 100**px**;*height*: 50**px**;*background-color*: #25CB33;}  
 .box2{*width*: 200**px**;*height*: 100**px**;*background-color*: **yellow**;}  
 .clearfix:after{  
 *content*: '';  
 *display*: **block**;  
 *clear*: **both**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
<**div** class=**"box1 clearfix"**>  
 <**ul**>  
 <**li**>**python**</**li**>  
 <**li**>**web**</**li**>  
 <**li**>**linux**</**li**>  
 </**ul**>  
</**div**>  
<**div** class=**"box2"**></**div**>  
</**body**>  
</**html**>

### 38.清除浮动4：overflow:hidden

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**18-清除浮动4-overflow-hidden**</**title**>  
 <**style** type=**"text/css"**>  
 \*{*margin*: 0;*padding*: 0;}  
 .box1{*width*: 300**px**;*overflow*: **hidden**;}  
 ul{*list-style*: **none**;}  
 .box1 ul li{*float*: **left**;*width*: 100**px**;*height*: 50**px**;*background-color*: #25CB33;}  
 .box2{*width*: 200**px**;*height*: 100**px**;*background-color*: **yellow**;}  
  
 </**style**>  
</**head**>  
<**body**>  
<**div** class=**"box1"**>  
 <**ul**>  
 <**li**>**python**</**li**>  
 <**li**>**web**</**li**>  
 <**li**>**linux**</**li**>  
 </**ul**>  
</**div**>  
<**div** class=**"box2"**>  
</**div**>  
</**body**>  
</**html**>

overflow还有几种用法，可查看下面浮动产生的问题补充

### 39.浮动产生的问题的补充

浮动产生的问题：

父元素不设置高度时，子元素设置了浮动，不会撑开父元素的高度，子元素不占位置

解决办法：

1.给父盒子设置固定高度，缺点：不灵活。例：height：480px;

2.给浮动元素的最后一个元素加一个空的块级元素，且该元素不浮动，设置clear：both；缺点：结构冗余；

3.在父级元素标签里面使用伪类

.wrap:after{

visibility: hidden;

clear: both;

content: '.';

display: block;

height: 0;}

4.给父元素添加 overflow: hidden;

使用overflow属性扩展盒子高度减少代码量，也减少了空的HTML标签，使代码更加简洁，清晰，从而提高了代码的可读性和网页性能

如果页面中有定位元素，并且元素超出了父级的范围，就必须使用clear属性来清除浮动来扩展盒子高度

注释：clear:both; 表示在左右两侧均不允许浮动元素。

clear属性只会对自身起作用，而不会影响其他元素。如果一个元素的右侧有一浮动对象，而这个元素设置了不允许右边有浮动对象，即clear：right，则这个元素会自动下移一格，达到本元素右边没有浮动对象的目的。

当浮动问题不解决的情况下显示的效果如下：



解决浮动效果以后显示的图片如下：



代码如下：

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**浮动问题和解决方案**</**title**>  
 <**style** type=**"text/css"**>  
 \*{*padding*: 0;*margin*: 0;}  
 .wrap{  
 *width*: 190**px**;  
 *background-color*: **yellowgreen**;  
 *margin*: 50**px auto**;  
 }  
 .wrap div{  
 *float*: **left**;  
 }  
 .wrap\_1{  
 *width*: 40**px**;  
 *height*: 30**px**;  
 *background-color*: #BFBFBF;  
 *margin-right*: 15**px**;  
 }  
 .wrap\_2{  
 *width*: 80**px**;  
 *height*: 30**px**;  
 *background-color*: **darkorange**;  
 }  
 .wrap\_3{  
 *width*: 40**px**;  
 *height*: 30**px**;  
 *background-color*: **darkblue**;  
 *margin-left*: 15**px**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
 <**div** class=**"wrap"**>  
 <**div** class=**"wrap\_1"**></**div**>  
 <**div** class=**"wrap\_2"**></**div**>  
 <**div** class=**"wrap\_3"**></**div**>  
 <**div** id=**"clearfix"**></**div**>  
 </**div**>  
</**body**>  
</**html**>

第一种方法：

在父盒子div下面添加一个盒子的绝对高度：（缺点不灵活）

*height*: 40**px**;

第二种方法：

在标签里面添加一个空盒子，然后再css里面添加空盒子的样式：（缺点：结构冗余）

<**div** class=**"wrap"**>  
 <**div** class=**"wrap\_1"**></**div**>  
 <**div** class=**"wrap\_2"**></**div**>  
 <**div** class=**"wrap\_3"**></**div**>  
 <**div** id=**"clearfix"**></**div**>  
</**div**>

添加CSS：

#clearfix{  
 *float*: **none**;  
 *clear*: **both**;  
}

第三种方法：在父级元素标签里面使用伪类（官方推荐的方法）

.wrap:after{  
 *visibility*: **hidden**;  
 *clear*: **both**;  
 *content*: '.';  
 *display*: **block**;  
 *height*: 0;}

提示：

visibility 属性规定元素是否可见。

visible：默认值。元素是可见的。

hidden ：元素是不可见的。

collapse：当在表格元素中使用时，此值可删除一行或一列，但是它不会影响表格的布局。被行或列占据的空间会留给其他内容使用。如果此值被用在其他的元素上，会呈现为 "hidden"。

inherit：规定应该从父元素继承 visibility 属性的值。

第四种方法：给父元素添加 overflow: hidden;

*overflow*: **hidden**;

使用overflow属性扩展盒子高度减少代码量，也减少了空的HTML标签，是代码更加简洁，清晰，从而提高了代码的可读性和网页性能

如果页面中有定位元素，并且元素超出了父级的范围，就必须使用clear属性来清除浮动来扩展盒子高度

overflow的值：

visible 默认值。内容不会被修剪，会呈现在元素框之外。

hidden 内容会被修剪，并且其余内容是不可见的。

scroll 内容会被修剪，但是浏览器会显示滚动条以便查看其余的内容。

auto 如果内容被修剪，则浏览器会显示滚动条以便查看其余的内容。

inherit 规定应该从父元素继承 overflow 属性的值。

### 30．margin垂直方向塌陷问题

垂直方向塌陷的原因是：基于标准文档流的情况下，当两个兄弟盒子设置垂直方向上的margin，那么以较大的值为准，此时我们称这种现象为塌陷；

注意：浮动的盒子不塌陷。

.box1{  
 *width*: 100**px**;  
 *height*: 75**px**;  
 *background-color*: #25CB33;  
 *margin-bottom*: 20**px**;  
}  
.box2{  
 *width*: 200**px**;  
 *height*: 150**px**;  
 *background-color*: **aqua**;  
 *margin-top*: 50**px**;  
}

此时的两个盒子距离为 50px；

### 31.水平居中（盒子和文本）

*margin*: 0 **auto**;  
*text-align*: **center**;

水平居中盒子用： margin: 0 auto;

水平居中文字用：text-align：center;

水平居中盒子的情况：

盒子水平居中，必须要给该盒子设置一个宽度

只有标准流下的盒子，才能使用margin：0 atuo;

当一个盒子浮动了，固定定位/绝对定位 ，margin: 0 auto; 也不能使用

### 27.position定位

overflow属性：

visible: 默认值。内容不会被修剪，会呈现在盒子之外。

hidden：内容会被修剪，并且其余内容不可见的

scroll：内容会被修剪，但是浏览器会显示滚动条以便查看其余内容

auto：如果内容被修剪，则浏览器会显示滚动条以便查看其余的内容。

在页面布局中，大的模块用overflow，每个盒子里面小的部分用定位去处理

定位：position

属性值：

static：默认值，没有定位

relative：相对定位

absolute：绝对定位

fixed: 固定定位

一、relative 相对定位：相对于自身原来位置进行偏移

relative 相对定位。相对定位是相对于该元素在文档流中的原始位置，即以自己原始位置为参照物。有趣的是，即使设定了元素的相对定位以及偏移值，元素还占有着原来的位置，即占据文档流空间。对象遵循正常文档流，但将依据top，right，bottom，left等属性在正常文档流中偏移位置。而其层叠通过z-index属性定义。

注意：position：relative的一个主要用法：方便绝对定位元素找到参照物。

偏移设置：top,left,right,bottom

单位：px

影响：

1.没有偏移量的时候对元素本身基本没有影响

2.提升层级

设置了相对定位的网页元素，无论是在标准流中还是在浮动流中，都不会对它的父级元素和相邻元素有任何影响，它只针对自身原来的位置进行偏移。

二、absolute 绝对定位。

定义：设置为绝对定位的元素框从文档流完全删除，并相对于最近的已定位祖先元素定位，如果元素没有已定位的祖先元素，那么它的位置相对于最初的包含块（即body元素）。元素原先在正常文档流中所占的空间会关闭，就好像该元素原来不存在一样。元素定位后生成一个块级框，而不论原来它在正常流中生成何种类型的框。

重点：如果父级设置了position属性，例如position:relative;，那么子元素就会以父级的左上角为原始点进行定位。这样能很好的解决自适应网站的标签偏离问题，即父级为自适应的，那我子元素就设置position:absolute;父元素设置position:relative;，然后Top、Right、Bottom、Left用百分比宽度表示。

另外，对象脱离正常文档流，使用top，right，bottom，left等属性进行绝对定位。而其层叠通过z-index属性定义。

总结：参照物用相对定位，子元素用绝对定位，并且保证相对定位参照物不会偏移即可。

三、 position:fixed

fixed：对象脱离正常文档流，使用top，right，bottom，left等属性以窗口为参考点进行定位，当出现滚动条时，对象不会随着滚动。而其层叠通过z-index属性 定义。 注意点： 一个元素若设置了 position:absolute | fixed; 则该元素就不能设置float。这 是一个常识性的知识点，因为这是两个不同的流，一个是浮动流，另一个是“定位流”。但是 relative 却可以。因为它原本所占的空间仍然占据文档流。

在理论上，被设置为fixed的元素会被定位于浏览器窗口的一个指定坐标，不论窗口是否滚动，它都会固定在这个位置。

### 28.堆叠：z-index

调整元素定位时重叠层的上下位置

- z-index属性值：整数，默认值为0

- 设置了position属性时，z-index属性可以设置各元素之间的重叠高低关系

- z-index值大的层位于其值小的层上方

z-index 优先级：数字越大，优先级越高

# 三 JavaScript

## 一 JS基础

### 1.JS引入方式和输出方式

javascript 的引入方式：

a.行内式，将JavaScript代码块直接写在标签内

<input type="button" onclick="alert(1)"/>

b.内部式

c.外部式：引入一个外部JS文件

1.JS引入的时候一般放在HTML所有内容之后

2.外部式元素内写代码块不会生效

<script type="text/javascript" src="01.js"></script>

JavaScript的输出方式：

1.document.write("")向文档内输入内容，不会覆盖源文档的内容

2.console 控制台输出【开发者工具】

3.console.log() 向控制台抛出一条消息

4.console.error() 向控制台抛出一个异常

5.console.dir() 输入一个对象的全部属性

6.console.clear() 清除

7.alert() 弹出一个带有一条指定信息的消息（警告）

8.innerHTML/innerText获取或者向指定元素内添加内容

9.innerHTML 获取元素内所有的内容（包括元素），还可以解析内容的样式

10.innerText 获取元素的文本内容

### 2.JS基本数据类型



1）基本数据类型：

数字 number

字符串 string

布尔 boolean

空值 Null

空值 Undefined

2）引用数据类型：

包括对象数据类型和函数数据类型

(1) 对象数据类型：

数组 array [ ]，对象object 正则REGEXP

(2) 函数数据类型：

function

### 3.Null 和Undefined 的区别

Undefined 类型

Undefined 类型只有一个值，即 undefined。当声明的变量未初始化时，该变量的默认值是 undefined。

当函数无明确返回值时，返回的也是值 "undefined";

Null 类型

另一种只有一个值的类型是 Null，它只有一个专用值 null，即它的字面量。值 undefined 实际上是从值 null 派生来的，因此 ECMAScript 把它们定义为相等的。

尽管这两个值相等，但它们的含义不同。undefined 是声明了变量但未对其初始化时赋予该变量的值，null 则用于表示尚未存在的对象（在讨论 typeof 运算符时，简单地介绍过这一点）。如果函数或方法要返回的是对象，那么找不到该对象时，返回的通常是 null。

var person=new Person()

var person=null

### 4.数据类型转换

JavaScript属于松散类型的程序语言

变量在声明的时候并不需要指定数据类型

变量只有在赋值的时候才会确定数据类型

表达式中包含不同类型数据则在计算过程中会强制进行类别转换

数字 + 字符串：数字转换为字符串

数字 + 布尔值：true转换为1，false转换为0

字符串 + 布尔值：布尔值转换为字符串true或false

强制类型转换函数：

函数parseInt：强制转换成整数，例如parseInt("6.12")=6 ; parseInt(“12a")=12 ; parseInt(“a12")=NaN ;parseInt(“1a2")=1

函数parseFloat：强制转换成浮点数，parseFloat("6.12")=6.12

函数eval：将字符串强制转换为表达式并返回结果 eval("1+1")=2 ; eval("1<2")=true

//1.将数字类型转换成字符串类型  
 var *n1* = 123;  
 var *n2* = '';  
 var *n3* = *n1* + *n2* ; // string 类型。隐式转换  
 *console*.log(*n3*);  
 *console*.log(typeof *n3*);  
 //强制类型转换,两种方法，String 和toString；  
 var *str1* = *String*(*n1*);  
 *console*.log(*str1*);  
 *console*.log(typeof *str1*);  
 var *num* = 234;  
 *console*.log(typeof *num*.toString());  
//2.将字符类串类型转换成数字类型  
 var *stringNum* = '789';  
 var *num2* = *Number*(*stringNum*);  
 *console*.log(typeof *num2*);  
// 在js中所有的数据类型，都被归为boolean类型。  
 var *n* = null;  
 var *m* = -123;  
 *console*.log(*Boolean*(*n*));  
 *console*.log(*Boolean*(*m*));

NAN：not a number；属于Number的一种

*console*.log(parseInt("hello")); //NAN  
*console*.log(parseInt("123")); //123  
*console*.log(parseInt("123abc"));//123  
*console*.log(parseInt("abc1234"));//NAN

typeof：判断数据类型

*console*.log(typeof 123); //number  
*console*.log(typeof "hello"); //String  
*console*.log(typeof true); //boolean  
*console*.log(typeof null); //object  
*console*.log(typeof undefined); //Undefined  
*console*.log(typeof [12,3,4]); //object  
*console*.log(typeof {"name":"123"}); //object

### 5.JS运算符

1）ECMAScript 算数运算符

加(＋)、 减(－)、 乘(\*) 、除(/) 、余数(% ) 加、减、乘、除、余数和数学中的运算方法一样 例如：9/2=4.5，4\*5=20，9%2=1

-除了可以表示减号还可以表示负号 例如：x=-y

+除了可以表示加法运算还可以用于字符串的连接 例如："abc"+"def"="abcdef"

递增(+ +) 、递减(- -)

i++ 和 ++i的区别（--的理解一样）

var *i* = 10;  
var *num* = *i*++; //先赋值num=i=10，然后在 i+1  
var *num2* = ++*i*;  
*console*.log(*num2*) //先让 i+1 然后在赋值给num2

NaN只要参与比较就是false，出发是 !== （不等于）

2）ECMAScript 逻辑运算符

等于 ( == ) 、不等于( != ) 、 大于( > ) 、 小于( < )  大于等于(>=) 、小于等于(<=)

与 (&&) 、或(||) 、非(!)

&& 1 = 1 1 || 1 = 1

&& 0 = 0 1 || 0 = 1

&& 0 = 0 0 || 0 = 0

!0=1

!1=0

逻辑 AND 运算符(&&)

逻辑 AND 运算的运算数可以是任何类型的，不止是 Boolean 值。

如果某个运算数不是原始的 Boolean 型值，逻辑 AND 运算并不一定返回 Boolean 值：

* 如果某个运算数是 null，返回 null。
* 如果某个运算数是 NaN，返回 NaN。
* 如果某个运算数是 undefined，返回undefined。

逻辑 OR 运算符(||)

与逻辑 AND 运算符相似，如果某个运算数不是 Boolean 值，逻辑 OR 运算并不一定返回 Boolean 值

**比较数字和字符串**

比较运算符两侧如果一个是数字类型,一个是其他类型,会将其类型转换成数字类型.

比较运算符两侧如果都是字符串类型,比较的是最高位的asc码,如果最高位相等,继续取第二位比较.

*console*.log(2===2); //true  
*console*.log(2==="2"); //false  
*console*.log(2=="2"); //True

*console*.log(3>20); //false  
*console*.log(3>"20"); //false  
*console*.log("3">"20"); //true

### 6. 流程控制语句- IF

// 1.模拟，总分 400 并且数学成绩大于80  
// 逻辑与 && 两个条件都成立的时候。才成立。  
  
var *math* = 90;  
var *english* = 95;  
var *sum* = 486;  
  
if (*sum* > 400 && *math* >89){  
 *console*.log("first num one")  
}else {  
 alert("失败了")  
}

### 7. 流程控制语句- switch

var *gameScore* = prompt("请输入你的选择：");  
switch(*gameScore*){  
  
 //case 表示一个条件，满足这个条件就执行case后面的语句，遇到break 跳出  
 case 'better':  
 *console*.log('better job');  
 break;  
 case 'good':  
 *console*.log('good job');  
 break;  
 case 'best':  
 *console*.log('best jod');  
 break;  
 default:  
 *console*.log('你没有输入信息！')  
}  
alert(11111)

### 8. 流程控制语句- while

//将 1-100 之间的所有是3的倍数 输出出来。  
var *i* = 1;  
while(*i*<=100){  
 if(*i*%3===0){  
 *console*.log(*i*)  
 }  
 *i*++;  
}

### 9. 流程控制语句- do while

// 1.初始化循环变量。2.判断循环条件。3.更新循环变量  
//do while 不管有没有满足while中的条件，do 里面的代码都回执行  
var *i* = 3;  
do {  
 *console*.log(*i*);  
 *i*++;  
}while(*i*<10)

### 10. 流程控制语句 - for

// 在浏览器中输出直角三角形，和等边三角形。使用python语言和js语言输出。  
for(var *i* = 0;*i*<6;*i*++){  
 for( j=*i*;j>=1;j--){  
 *document*.write('\*')  
 }  
 *document*.write('\*');  
 *document*.write('<br>');  
}  
*document*.write('=================================================<br>');  
for(var *m* = 1;*m*<=6;*m*++){  
 for(var *n* = 6-1;*n*>=*m*;*n*--){  
 *document*.write('&nbsp;');  
 }  
 for(var *p* = 1;*p*<=2\**m*-1;*p*++){  
 *document*.write('\*');  
 }  
 *document*.write('<br>')  
}

### 11. 异常处理

try {

//这段代码从上往下运行，其中任何一个语句抛出异常该代码块就结束运行

}

catch (e) {

// 如果try代码块中抛出了异常，catch代码块中的代码就会被执行。

//e是一个局部变量，用来指向Error对象或者其他抛出的对象

}

finally {

//无论try中代码是否有异常抛出（甚至是try代码块中有return语句），finally代码块中始终会被执行。

}

try {  
 *console*.log(123);  
 throw *Error*('define value');  
}  
catch(*e*){  
 *console*.log(*e*)  
}  
finally {  
 *console*.log("执行完毕")  
}

throw Error('define value'); 主动抛出异常

### 12.JavaScript对象

object对象：ECMAScript 中的所有对象都由这个对象继承而来；Object 对象中的所有属性和方法都会出现在其他对象中

11种内置对象包括：

Array ,String , Date, Math, Boolean, Number Function, Global, Error, RegExp , Object

简介：在JavaScript中除了null和undefined以外其他的数据类型都被定义成了对象，也可以用创建对象的方法定义变量，String、Math、Array、Date、RegExp都是JavaScript中重要的内置对象，在JavaScript程序大多数功能都是通过对象实现的

var aa=Number.MAX\_VALUE;

//利用数字对象获取可表示最大数

var bb=new String("hello JavaScript");

//创建字符串对象

var cc=new Date();

//创建日期对象

var dd=new Array("星期一","星期二","星期三","星期四");

//数组对象

### 13.String对象

// 字符串创建  
var *str1* = 'hello';  
var *str2* = new *String*("world");  
*console*.log(typeof *str1*); //String  
*console*.log(typeof *str2*); //Object 堆的方式存储  
//字符串属性  
*console*.log(*str1*.length);  
//字符串方法  
//格式编排方法  
*console*.log(*str1*.italics()); //<i>hello</i>  
*console*.log(*str1*.bold()); //<b>hello</b>  
*console*.log(*str1*.anchor('abc')); //<a name="abc">hello</a>  
//字符串大小写转换：toLowerCase ：字符串全部小写；toUpperCase ：字符串全部大写；  
//获取指定字符串：x.charAt(index) index索引的字符；  
//查询字符串索引  
var *str3* = "welcome come to china";  
var *str4*=*str1*.indexOf("l"); //获取第一个l的index  
var *str5*=*str1*.lastIndexOf("l"); //获取最后一个l 的index  
//截取字符串  
*console*.log(*str3*.substr(1,5)); //elcom  
*console*.log(*str3*.substring(1,5)); // elco  
*console*.log(*str3*.slice(1,5)); //elco  
//替换字符串  
*console*.log(*str3*.replace('china','SH'));//welcome come to SH  
//切割字符串  
*console*.log(*str3*.split("o")); //["welc", "me c", "me t", " china"]  
//连接字符串  
*console*.log(*str1*.concat(","+*str2*));//hello,world

### 14.JS数组对象

//创建方式  
var *arr1* = [1,'a',2,[11,22],{"name":"ryan"}];  
var *arr2* = new *Array*(3);  
var *arr3* = new *Array*(1,"world",true,[1,2,3]);  
  
*console*.log(typeof *arr1*); //object  
*console*.log(typeof *arr2*); //object  
*console*.log(typeof *arr3*); //object  
//注意：如果创建数组的时候是 var arr2 = new Array(3); 这样创建的，那么会默认为数组的大小，3 不能内容，是大小；  
  
// 创建二维数组  
var *cnweek*=new *Array*(3);  
for (var *i*=0;*i*<=2;*i*++){  
 *cnweek*[*i*]=new *Array*(2);  
}  
*cnweek*[0][0]="星期日";  
*cnweek*[0][1]="Sunday";  
*cnweek*[1][0]="星期一";  
*cnweek*[1][1]="Monday";  
*cnweek*[2][0]="星期六";  
*cnweek*[2][1]="Saturday";  
*console*.log(*cnweek*);  
  
//Array对象的属性  
*console*.log(*cnweek*.length); //3  
//Array对象的方法  
var *arr4* = [1,'a',2,'bcd'];  
var *arr5* = [3,4];  
// 连接数组-join方法  
*console*.log(*arr4*.join("/")); //1/a/2/bcd  
//连接数组-concat方法  
*console*.log(*arr5*.concat(*arr4*)); //[3, 4, 1, "a", 2, "bcd"]  
// toString 所有的数据类型都有该方法  
*console*.log(typeof *arr4*.toString()); //1,a,2,bcd ： string类型  
// 数组排序-reverse sort  
// sort如果全是数字进行排序，是按照最高位。  
// sort 的意思是，只要一个值大于另一个值只要返回一个True（正值就行）  
*console*.log(*arr4*.reverse()); // ["bcd", 2, "a", 1]  
*console*.log(*arr4*.sort()); // [1, 2, "a", "bcd"]  
//sort 针对全数字  
var *arr6* = [1,3,100,200,400];  
//如果只使用sort那么排序结果出错  
*console*.log(*arr6*.sort()); //[1, 100, 200, 3, 400]  
//正确排序方法  
function f(*a*,*b*) {  
 if(*a*>*b*){  
 return 1  
 }else if(*a*<*b*){  
 return -1  
 }else {  
 return 0  
 }  
}  
*console*.log(*arr6*.sort(f)); //[1, 3, 100, 200, 400]  
//上面的 函数 f 相当于下面的函数  
function f2(*a*,*b*) {  
 return *a*-*b*;  
}  
*console*.log(*arr6*.sort(f)); //[1, 3, 100, 200, 400]  
//数组切片  
var *oArr1*=['a','b','c','d','e','f','g','h'];  
*console*.log(*oArr1*.slice(1,2)); //["b"]  
*console*.log(*oArr1*.slice(2,-1)); //["c", "d", "e", "f", "g"]  
//删除 splice  
//数组进出栈操作  
 //push pop这两个方法模拟的是一个栈操作

//push是将value值添加到数组x的结尾  
 //pop是将数组x的最后一个元素删除

var *numArr* = [1,2,3,4];  
 *numArr*.push(5,6);  
 *console*.log(*numArr*); //[1, 2, 3, 4, 5, 6]  
 *numArr*.pop();  
 *console*.log(*numArr*); //[1, 2, 3, 4, 5]  
 // unshift shift  
 //unshift是将value值插入到数组x的开始  
 //shift是将数组x的第一个元素删除  
 var *numArr2* = [1,2];  
 *numArr2*.unshift(3,4);  
 *console*.log(*numArr2*); //[3, 4, 1, 2]  
 *numArr2*.shift();  
 *console*.log(*numArr2*); //[4, 1, 2]

js中数组的特性

js中的数组特性1: js中的数组可以装任意类型,没有任何限制.

js中的数组特性2: js中的数组,长度是随着下标变化的.用到多长就有多长.

var *Arr10* = ['abc',123,1.14,true,null,undefined,new *String*('1213'),new *Function*('a','b','alert(a+b)')];

### 15.JS date对象

// 创建日期对象  
var *myDate* = new *Date*();  
  
//获取当天的号数  
var *day* = *myDate*.getDate();  
*console*.log(*day*);  
  
// 获取当天的日期时间  
*console*.log(*Date*()); // Mon Apr 09 2018 11:56:46 GMT+0800 (中国标准时间)  
  
// getMonth(): 是从0--11 开始计算，如果得到当月则 +1  
*console*.log(*myDate*.getMonth()+1);  
  
// getDay()： 0-6，0 代表周日；  
*console*.log(*myDate*.getDay());  
  
*console*.log(*myDate*.getFullYear()); // 2018  
  
*console*.log(*myDate*.getHours()); // 获取时间 的小时 0-23  
  
*console*.log(*myDate*.setDate(1))

### 16.JS math对象

/\*  
abs(x) 返回数的绝对值。  
exp(x) 返回 e 的指数。  
floor(x)对数进行下舍入。  
log(x) 返回数的自然对数（底为e）。  
max(x,y) 返回 x 和 y 中的最高值。  
min(x,y) 返回 x 和 y 中的最低值。  
pow(x,y) 返回 x 的 y 次幂。  
random() 返回 0 ~ 1 之间的随机数。  
round(x) 把数四舍五入为最接近的整数。  
sin(x) 返回数的正弦。  
sqrt(x) 返回数的平方根。  
tan(x) 返回角的正切。  
\*/  
 var *x* = 1.234;  
 // ceil 大于等于x，并且与它最接近的整数  
 var *a* = *Math*.ceil(*x*);  
 *console*.log(*a*);  
 // floor 小于等于x，并且与它最接近的整数  
 var *b* = *Math*.floor(*x*);  
 *console*.log(*b*);  
 *console*.log(*Math*.max(2,4)) ; //min 最小值  
 //经常使用的 random  
 var *ran* = *Math*.random(); // 0-1 之间的随机数  
 *console*.log(*ran*);  
 //100-200 之间的随机数  
 var *c* = 100+*Math*.random()\*100;  
 *console*.log(*c*);  
 // 5-15 之间的随机数 ; min + Math.random()\*(max-min)  
 var *d* = 5+*Math*.random()\*(15-5);  
 *console*.log(*d*)

### 17.JS RegExp对象

//RegExp对象  
 // 在表单验证时使用该对象验证用户填入的字符串是否符合规则.  
 //创建正则对象方式1 参数1 正则表达式 参数2 验证模式 g global / i 忽略大小写. //参数2一般填写g就可以，也有“gi”.  
 // 用户名 首字母必须是英文, 除了第一位其他只能是英文数字和\_ . 长度最短不能少于6位 最长不能超过12位  
 //----------------------------创建方式1  
 /\* var reg1 = new RegExp("^[a-zA-Z][a-zA-Z0-9\_]{5,11}$","g");  
 //验证字符串  
 var str = "bc123";  
 alert(reg1.test(str));// true  
 //----------------------------创建方式2 /填写正则表达式/匹配模式;  
 var reg2 = /^[a-zA-Z][a-zA-Z0-9\_]{5,11}$/g;  
 alert(reg2.test(str));// true  
 \*/  
 //-------------------------------正则对象的方法-------------------  
 //test方法 ==> 测试一个字符串是否复合 正则规则. 返回值是true 和false.  
 //-------------------------String 中与正则结合的4个方法------------------.  
 // macth search split replace  
 var *str* = "hello world";  
 //alert(str.match(/o/g)); //查找字符串中 复合正则的 内容.  
 //alert(str.search(/h/g));// 0 查找字符串中符合正则表达式的内容位置  
 //alert(str.split(/o/g)); // 按照正则表达式对字符串进行切割. 返回数组;  
 alert(*str*.replace(/o/g, "s")); // hells wsrld 对字符串按照正则进行替换.

### 18.JS Function对象

JS的函数对象没有先后顺序，调用在函数代码之前。

//函数的创建方式一  
function f(*x*,*y*) {  
 return *x*+*y*;  
}  
*console*.log(f(3,4));  
// 函数的创建方式二(了解)  
var *obj* = new *Function*("参数1","参数n","function\_body");  
// function 对象的属性 length 声明了函数期望的参数个数。  
*console*.log(f.length); //2  
  
function add(*x*,*y*,*z*) {  
 return *x*+*y*+*z*;  
}  
*console*.log(add(1,2,3,4,5,6)); // 6  
*console*.log(add(1,2)); // NaN (此时z没有值，是undefined)  
*console*.log(add(1,2,"a")); // 3a (1+2+a)  
  
function func01(*x*,*y*) {  
 return *x*+*y*;  
}  
  
var *func01* = 1;  
var *b* = 2;  
*console*.log("-----");  
// console.log(func01(func01,b)); //Uncaught TypeError: func01 is not a function 因为这里的func01 是做为函数调用的，可是此时的func01,被常量替换了，因此会报错  
function func02(*a*,*b*) {  
 // arguments 可以得到函数传参的列表  
 *console*.log(*arguments*[0]); // 1  
 *console*.log(*arguments*[1]); // 2  
 return *a*+*b*;  
}  
*console*.log(func02(1,2)); // 3  
  
// 计算多个不确定数字的和  
function func03() {  
 var sum = 0;  
 for (var i = 0;i<*arguments*.length;i++){  
 sum+=*arguments*[i]  
 }  
 return sum;  
}  
*console*.log(func03(1,3,4,5,6,7,8)); //34  
  
// 匿名函数(不是一个标准的函数对象,如果调用函数在匿名函数之前，那么会报错)  
var func04 = function(*arg*){  
 return "tony";  
};  
*console*.log(func04(1)); //tony  
*console*.log(typeof func04);  
  
//匿名函数2 （函数本身就是一个执行体、）  
(function(){  
 alert("tony");  
} )();  
  
(function(*arg*){  
 *console*.log(*arg*);  
})('123')

### 19．函数的作用域链和闭包

作用域：

js的作用域和py相似，if while等控制语句并没有自己作用域；而函数是有自己的作用域的；

var *city* = 'beijing';  
function Bar(){  
 *console*.log(*city*);  
}  
function func(){  
 var city = 'shanghai';  
 return Bar;  
}  
var *ret* = func();  
*ret*(); //beijing

### 20.定时器

setInterval 定时器：是window里面的方法。

setInterval(code,执行时间（单位：毫秒）)

setTimeout(code,1000) 清理定时器

<**script** type=**"text/javascript"**>  
 var *n* = 0;  
 var *time* = null;  
 *time* = setInterval(function () {  
 *n*++;  
 *console*.log(*n*)  
 },1000);  
 // setTimeout(code,1000)  
 setTimeout(function () {  
 *console*.log(*time*);  
 clearInterval(*time*)  
 },5000);  
</**script**>

## 二 JS DOM对象

### 1.什么是DOM对象

根据 W3C 的 HTML DOM 标准，HTML 文档中的所有内容都是节点：

* 整个文档是一个文档节点
* 每个 HTML 元素是元素节点
* HTML 元素内的文本是文本节点
* 每个 HTML 属性是属性节点
* 注释是注释节点

HTML DOM 将 HTML 文档视作树结构。这种结构被称为节点树。通过 HTML DOM，树中的所有节点均可通过 JavaScript 进行访问。所有 HTML 元素（节点）均可被修改，也可以创建或删除节点。

一些常用的 HTML DOM 方法：

* getElementById(id) - 获取带有指定 id 的节点（元素）
* appendChild(node) - 插入新的子节点（元素）
* removeChild(node) - 删除子节点（元素）

一些常用的 HTML DOM 属性：

* innerHTML - 节点（元素）的文本值
* parentNode - 节点（元素）的父节点
* childNodes - 节点（元素）的子节点
* attributes - 节点（元素）的属性节点

### 2.模态框案例

实现效果：



当点击“弹出”以后页面变成：



<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**模态框案例**</**title**>  
 <**style** type=**"text/css"**>  
 \*{*margin*: 0;*padding*: 0;}  
 html,body{*height*: 100%;}  
 #box{  
 *width*: 100%;  
 *height*: 100%;  
 *background*: #808080;  
 }  
 #content{  
 *position*: **relative**;  
 *top*: 150**px**;  
 *width*: 400**px**;  
 *height*: 200**px**;  
 *line-height*: 200**px**;  
 *text-align*: **center**;  
 *color*: **red**;  
 *background-color*: **white**;  
 *margin*: **auto**;  
  
 }  
 #span1{  
 *position*: **absolute**;  
 *background-color*: **greenyellow**;  
 *top*: 0;  
 *right*: 0;  
 *width*: 30**px**;  
 *height*: 30**px**;  
 *line-height*: 30**px**;  
 *text-align*: **center**;  
 *color*: **black**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
 <**button** id=**'btn'**>**弹出**</**button**>  
</**body**>  
<**script** type=**"text/javascript"**>  
 // 获取dom 元素。  
 var *btn* = *document*.getElementById('btn');  
 // 创建div,p,span标签， dom元素  
 var *oDiv* = *document*.createElement('div');  
 var *oP* = *document*.createElement('p');  
 var *oSpan* = *document*.createElement('span');  
 // 拿到节点对象 赋值属性  
 *oDiv*.id = 'box';  
 *oP*.id = 'content';  
 *oP*.innerHTML = '模态框成功弹出'; //一般设置文档内容  
 *oSpan*.id = 'span1';  
 *oSpan*.innerHTML = 'x';  
 // 在div下面添加 p标签  
 *oDiv*.appendChild(*oP*);  
 // 在p标签下面添加 span标签  
 *oP*.appendChild(*oSpan*);  
   
 *btn*.onclick = function () {  
 //动态的添加到body中一个div节点  
 *btn*.parentNode.insertBefore(*oDiv*,*btn*)  
 };  
   
 *oSpan*.onclick = function () {  
 *oDiv*.parentNode.removeChild(*oDiv*)  
 }  
</**script**>  
</**html**>

用到的知识点：

1.获取dom元素，如果标签里面有id，那么使用 document.getElementById('id名称获取')；

2.创建html元素标签，用 document.createElement('标签名称')；

3.给标签赋值属性：节点对象.id = 'box' ;相当于是给某个标签设置Id='box'

4.给标签元素里面添加内容： oSpan.innerHTML = 'x'; 给oSpan标签对象里面添加内容 'x'

5.添加子节点: oDiv.appendChild(oP); 在oDiv标签对象下面添加oP标签对象。

6.点击事件：onclick 点击事件。

7. btn.parentNode.insertBefore(oDiv,btn);在事件触发btn事件以后，在btn的前面添加一个oDiv 的父节点。

8. oDiv.parentNode.removeChild(oDiv) ；触发某个事件以后，把oDiv的父节点移除掉。

### 3.点击有惊喜案例

效果如下：不断点击知道没有

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**点击有惊喜案例**</**title**>  
 <**style** type=**"text/css"**>  
 \*{*margin*: 0;*padding*: 0;}  
 .box{  
 *width*: 200**px**;  
 *height*: 200**px**;  
 *background-color*: **red**;  
 *text-align*: **center**;  
 *color*: **greenyellow**;  
 *line-height*: 200**px**;  
 *font-size*: 25**px**;  
 *font-weight*: **bold**;  
 *margin*: 20**px auto**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
 <**div** class=**"box"**>  
 **点击有惊喜！！** </**div**>  
</**body**>  
<**script** type=**"text/javascript"**>  
 var *oBox* = *document*.getElementsByClassName('box')[0];  
 *console*.log(*oBox*);  
 var *a* = 0;  
 *oBox*.onclick = function () {  
 *a*++;  
 if(*a*%4===1){  
 this.style.background = 'green';  
 this.innerText = '继续点击！';  
 }else if(*a*%4===2){  
 this.style.background = 'blue';  
 this.innerText = '骗你的！';  
 }else if(*a*%4===3){  
 this.style.background = 'transparent';  
 this.innerText = '';  
 }else{  
 this.style.background = 'red';  
 this.innerText = '点击有惊喜';  
 }  
 }  
</**script**>  
</**html**>

该例子用到的知识点有：

1. getElementsByClassName('box')；这个的意思是获取到class名称为 box 的dom元素。此时得到的结果相当于一个列表；然后用 document.getElementsByClassName('box')[0];获得 第一个元素。

2. this ：这个关键词可以相当于python 类里面的 self 去理解，就是实例对象自己。此例中相当于 oBox

3. this.style.background :意思是，给某对象添加一个背景属性。

扩展： html dom style对象可以修饰：

背景，边框和边距，布局，列表，杂项，定位，打印，滚动条，表格，文本，规范

查看：http://www.w3school.com.cn/jsref/dom\_obj\_style.asp

### 4.简易留言板案例

效果如下：

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**简易留言板**</**title**>  
 <**style** type=**"text/css"**>  
 span{  
 *color*: **red**;  
 *font-size*: 10**px**;  
 *cursor*: **pointer**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
 <**h1**>**简易留言板**</**h1**>  
 <**div** id=**"box"**>  
 </**div**>  
 <**textarea** id=**"msg"**></**textarea**>  
 <**input** type=**"button"** id=**"btn"** value=**"留言"**>  
 <**button** onclick=**"**sum()**"**>**统计**</**button**>  
</**body**>  
<**script** type=**"text/javascript"**>  
 var *ol* = *document*.createElement('ol');  
 var *box* = *document*.getElementById('box');  
 *box*.appendChild(*ol*);  
 var *btn* = *document*.getElementById('btn');  
 var *msg* = *document*.getElementById('msg');  
 var *count* = 0;  
 *btn*.onclick = function () {  
 var li = *document*.createElement('li');  
 //设置内容：  
 if(*msg*.value.length !== 0){  
 *ol*.appendChild(li);  
 li.innerHTML = *msg*.value + "<span>&nbsp;&nbsp;删除</span>";  
 var lis = *document*.getElementsByTagName('li');  
 if(lis.length === 0){  
 *ol*.appendChild(li);  
 *count*++;  
 }else {  
 *ol*.insertBefore(li,lis[0]);  
 *count*++;  
 }  
 *msg*.value = '';  
 var spans = *document*.getElementsByTagName('span');  
 for(var i = 0;i<spans.length;i++){  
 spans[i].onclick = function () {  
 *ol*.removeChild(this.parentNode);  
 *count*--;  
 }  
 }  
  
 }else {  
 alert('内容不能为空');  
 }  
 };  
 function sum() {  
 alert("留言统计："+*count*+"条")  
 }  
</**script**>  
</**html**>

知识点：标签对象.value 表示获取的标签对象下面的text value

### 5.选项卡

效果如下：

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**选项卡**</**title**>  
 <**style** type=**"text/css"**>  
 \*{*padding*: 0;*margin*: 0;}  
 /\*去掉ul前面的黑色点\*/  
 ul{*list-style*: **none**;}  
 #tab{  
 *width*: 480**px**;  
 *margin*: 20**px auto**;  
 *border*: 1**px solid red**;  
 }  
 ul li{  
 *float*: **left**;  
 *width*: 160**px**;  
 *height*: 60**px**;  
 /\*垂直居中\*/  
 *line-height*: 60**px**;  
 *text-align*: **center**;  
 *background-color*: #cccccc;  
 }  
 ul li a{  
 /\*去掉a标签的下划线\*/  
 *text-decoration*: **none**;  
 *color*: **black**;  
 }  
 li.active{  
 *background-color*: **white**;  
 }  
 p{  
 /\*内容隐藏掉\*/  
 *display*: **none**;  
 *height*: 200**px**;  
 *text-align*: **center**;  
 *line-height*: 200**px**;  
 *background-color*: **white**;  
 }  
 p.active{  
 *display*: **block**;  
 *background-color*: **white**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
 <**div** id=**"tab"**>  
 <**ul**>  
 <**li** class=**"active"**><**a** href=**"#"**>**首页**</**a**></**li**>  
 <**li**><**a** href=**"#"**>**新闻**</**a**></**li**>  
 <**li**><**a** href=**"#"**>**图片**</**a**></**li**>  
 </**ul**>  
 <**p** class=**"active"**>**首页内容**</**p**>  
 <**p**>**新闻内容**</**p**>  
 <**p**>**图片内容**</**p**>  
 </**div**>  
</**body**>  
  
<**script** type=**"text/javascript"**>  
 var *tabLi* = *document*.getElementsByTagName('li');  
 var *tabContent* = *document*.getElementsByTagName('p');  
 for(var *i* = 0;*i* < *tabLi*.length;*i*++){  
 // 保存i的变量  
 *tabLi*[*i*].index = *i*;  
 *tabLi*[*i*].onclick = function () {  
 for (var j = 0;j < *tabLi*.length;j++){  
 *tabLi*[j].className = '';  
 *tabContent*[j].className = '';  
 }  
 this.className = 'active';  
 *tabContent*[this.index].className = 'active';  
 }  
 }  
</**script**>  
</**html**>

该例子需要注意的知识点：

1.隐藏标签属性： display：none

2.获取标签： getElementsByTagName(“标签名称”)

### 6.仿淘宝搜索框

效果如下：

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**仿淘宝搜索框**</**title**>  
 <**style** type=**"text/css"**>  
 \*{*padding*: 0;*margin*: 0;}  
 #search{*position*: **relative**;}  
 input{  
 /\*点击input框输入内容的时候，没有绿色外框\*/  
 *outline*: **none**;  
 *display*: **block**;  
 *width*: 200**px**;  
 *height*: 20**px**;  
 *font-size*: 20**px**;  
 *border*: 2**px solid orange**;  
 /\*设置框的圆角\*/  
 *border-radius*: 10**px**;  
 *margin-top*: 20**px**;  
 }  
 #msg{  
 *position*: **absolute**;  
 *top*: 3**px**;  
 *left*: 4**px**;  
 *font-size*: 12**px**;  
 *color*: **gray**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
 <**div** id=**"search"**>  
 <!--<input type="text" id="text" placeholder="hehe">-->  
 <**input** type=**"text"** id=**"text"** />  
 <**label** for=**"txt"** id=**"msg"**>**搜索内容**</**label**>  
 </**div**>  
</**body**>  
<**script** type=**"text/javascript"**>  
 var *txt* = *document*.getElementById('text');  
 var *msg* = *document*.getElementById('msg');  
 // 检测用户表单输入的时候  
 *txt*.oninput = function () {  
 if(this.value === ''){  
 *msg*.style.display = 'block'  
 }else {  
 *msg*.style.display = 'none'  
 }  
 }  
</**script**>  
</**html**>

用到的知识点有：

1.input事件：“oninput”当用户向 <input> 中尝试输入时执行

### 7.获取当前最新时间

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**获取当前的最新时间**</**title**>  
</**head**>  
<**body**>  
</**body**>  
<**script** type=**"text/javascript"**>  
 setInterval(function () {  
 var date = new *Date*();  
 var y = date.getFullYear();  
 var m = date.getMonth();  
 var d = date.getDate();  
 var h = date.getHours();  
 var min = date.getMinutes();  
 var s = date.getSeconds();  
 *document*.body.innerHTML = "今天是"+y+'年'+num((m+1))+"月" + num(d) + "日" + num(h)+":"+num(min)+":"+num(s)  
 },1000);  
 function num(*s*) {  
 if(*s*<10){  
 return "0"+*s*;}  
 return *s* }  
</**script**>  
</**html**>

### 8.匀速运动案例

效果如下：



<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**匀速运动案例**</**title**>  
 <**style** type=**"text/css"**>  
 \*{*padding*: 0;*margin*: 0;}  
 .box{  
 *width*: 100**px**;  
 *height*: 100**px**;  
 *background-color*: **greenyellow**;  
 *position*: **absolute**;  
 *top*: 50**px**;  
 *left*: 1**px**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
  
 <**div** id=**"wrap"**>  
 <**button** id=**"btn"**>**运动**</**button**>  
 <**button** id=**"btn2"**>**停止**</**button**>  
 <**div** class=**"box"** id=**"box1"**></**div**>  
 </**div**>  
</**body**>  
<**script** type=**"text/javascript"**>  
 var *btn* = *document*.getElementById('btn');  
 var *btn2* = *document*.getElementById('btn2');  
 var *box1* = *document*.getElementById('box1');  
 var *count* = 0;  
 var *num* = 0;  
 var *time* = null;  
 *btn2*.onclick = function () {  
 clearInterval(*time*);  
 *box1*.style.display = 'block';  
 };  
 *btn*.onclick = function () {  
 *time* = setInterval(function () {  
 *count* += 1;  
 *num* += 0.1;  
 if(*count* > 500){  
 clearInterval(*time*);  
 *box1*.style.display = 'none';  
 }  
 *box1*.style.left = *count*+'px';  
 *box1*.style.borderRadius = *num* +'px';  
 },10);  
 }  
</**script**>  
</**html**>

### 9. 5秒之后关闭广告

效果如下：



<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**5秒之后关闭广告**</**title**>  
 <**style** type=**"text/css"**>  
 \*{*margin*: 0;*padding*: 0;}  
 img{*position*: **fixed**;}  
 #left{*left*: 0;}  
 #right{*right*: 0;}  
 ul{*list-style*: **none**;}  
 ul li{*font-size*: 25**px**;*color*: #84a42b}  
 </**style**>  
</**head**>  
<**body**>  
 <**img** src=**"./images/4.png"** id=**"left"**>  
 <**img** src=**"./images/3.png"** id=**"right"**>  
 <**ul**><**li**>**两情若是久长时，又岂在朝朝暮暮！**</**li**></**ul**>  
</**body**>  
<**script** type=**"text/javascript"**>  
 *window*.onload = function () {  
 var left = *document*.getElementById('left');  
 var right = *document*.getElementById('right');  
 setTimeout(function () {  
 left.style.display = 'none';  
 right.style.display = 'none';  
 },5000)  
 }  
</**script**>  
</**html**>

本例知识点：

1. window.onload是一个事件，当文档加载完成之后就会触发该事件，可以为此事件注册事件处理函数，并将要执行的脚本代码放在事件处理函数中，于是就可以避免获取不到对象的情况。

2．如果只执行一次，那么就用setTimeout。

### 10.小米滚动

效果如下：

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**小米滚动**</**title**>  
 <**style** type=**"text/css"**>  
 #box{  
 *width*: 353**px**;  
 *height*: 400**px**;  
 *margin*: 100**px auto**;  
 *border*: 3**px solid** #25CB33;  
 *position*: **relative**;  
 /\*超出部分隐藏\*/  
 *overflow*: **hidden**;  
 }  
 img{  
 *position*: **absolute**;  
 *top*: 0;  
 *left*: 0;  
 }  
 span{  
 *width*: 100%;  
 *height*: 50%;  
 *position*: **absolute**;  
 /\*display: block;\*/  
 }  
 #picdown{  
 *bottom*: 0;  
 }  
 </**style**>  
</**head**>  
<**body**>  
 <**div** id=**"box"**>  
 <**img** src=**"./images/06.jpg"** id=**"meinv"**>  
 <**span** class=**"up"** id=**"picup"**>**up**</**span**>  
 <**span** class=**"down"** id=**"picdown"**>**down**</**span**>  
 </**div**>  
</**body**>  
 <**script** type=**"text/javascript"**>  
 var *up* = *document*.getElementById('picup');  
 var *down* = *document*.getElementById('picdown');  
 var *img* = *document*.getElementById('meinv');  
 var *count* = 0;  
 var *time* = null;  
 // 鼠标移入的时候  
 *up*.onmouseover = function () {  
 clearInterval(*time*);  
 *time* = setInterval(function () {  
 *count* -= 1;  
 *count* >= -6000 ? *img*.style.top = *count* + 'px':clearInterval(*time*);  
  
 })  
 };  
 *down*.onmouseover = function () {  
 clearInterval(*time*);  
 *time* = setInterval(function () {  
 *count* += 1;  
 *count* < 0 ? *img*.style.top = *count* + 'px':clearInterval(*time*);  
 })  
 };  
 </**script**>  
</**html**>

本例知识点：

1.鼠标事件：onmouseover：对鼠标移动进行事件操作。

2. count >= -6000 ? img.style.top = count + 'px':clearInterval(time); 相当于if else 语句，表示，如果count 大于-6000 ，那么 count+’px’

3.当在不同的事件操作中，应该清理下别的事件的定时器。

### 11.无缝轮播图

效果如下：



<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**无缝轮播图**</**title**>  
 <**style** type=**"text/css"**>  
 \*{*padding*: 0;*margin*: 0;}  
 ul{*list-style*: **none**;}  
 .box{  
 *width*: 440**px**;  
 *height*: 300**px**;  
 *margin*: 20% **auto** ;  
 *overflow*: **hidden**;  
 *position*: **relative**;  
 }  
 ul li{  
 *float*: **left**;  
 }  
 .box ul{  
 *width*: 400%;  
 *position*: **absolute**;  
 *top*: 0;  
 *left*: 0;  
 }  
 </**style**>  
</**head**>  
<**body**>  
 <**div** class=**"box"**>  
 <**ul**>  
 <**li**><**img** src=**"./images/1.png"** alt=**""**></**li**>  
 <**li**><**img** src=**"./images/3.png"** alt=**""**></**li**>  
 <**li**><**img** src=**"./images/4.png"** alt=**""**></**li**>  
 <**li**><**img** src=**"./images/5.png"** alt=**""**></**li**>  
 </**ul**>  
 </**div**>  
</**body**>  
<**script** type=**"text/javascript"**>  
 var *box* = *document*.getElementsByClassName('box')[0];  
 var *ul* = *box*.children[0];  
 //让图片往左走，先初始化一个num  
 var *num* = 0;  
 var *time* = null;  
 function autoPlay() {  
 *num*--;  
 *num* <= -410 ? *num* =0 :*num* ;  
 *ul*.style.left = *num*+'px';  
 }  
  
 *time* = setInterval(autoPlay,30);  
 // 鼠标移上去的时候  
 *box*.onmouseover = function () {  
 clearInterval(*time*)  
 };  
 *box*.onmouseout = function () {  
 *time* = setInterval(autoPlay,30)  
 }  
</**script**>  
</**html**>

### 12. 正反选案例

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**正反选**</**title**>  
</**head**>  
<**body**>  
  
<**button** type=**"submit"** onclick=**"**selectAll()**"**>**全选**</**button**>  
<**button** type=**"submit"** onclick=**"**Reverse()**"**>**反选**</**button**>  
<**button** type=**"submit"** onclick=**"**Cancel()**"** >**取消**</**button**>  
<**hr**>  
<**table** border=**"1px"**>  
 <**tr**>  
 <**td**><**input** type=**"checkbox"**></**td**>  
 <**td**>**111**</**td**>  
 </**tr**>  
  
 <**tr**>  
 <**td**><**input** type=**"checkbox"**></**td**>  
 <**td**>**222**</**td**>  
 </**tr**>  
 <**tr**>  
 <**td**><**input** type=**"checkbox"**></**td**>  
 <**td**>**333**</**td**>  
 </**tr**>  
</**table**>  
<**script**>  
 function selectAll() {  
 var inputs = *document*.getElementsByTagName('input');  
 for(var i=0;i<inputs.length;i++){  
 input = inputs[i];  
 input.checked = true;  
 }  
 }  
 function Cancel() {  
 var inputs = *document*.getElementsByTagName('input');  
 for(var i=0;i<inputs.length;i++){  
 input = inputs[i];  
 input.checked = false;  
 }  
 }  
 function Reverse() {  
 var inputs = *document*.getElementsByTagName('input');  
 for(var i=0;i<inputs.length;i++){  
 input = inputs[i];  
 if(input.checked){  
 input.checked = false  
 }else {  
 input.checked = true  
 }  
 }  
 }  
</**script**>  
</**body**>  
</**html**>

### 13.省份城市二级联动

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**城市二级联动**</**title**>  
</**head**>  
<**body**>  
<**select** id=**"provinces"**>  
 <**option** value=**""**>**请选择省份**</**option**>  
</**select**>  
  
<**select** id=**"citys"**>  
 <**option** value=**""**>**请选择城市**</**option**>  
</**select**>  
</**body**>  
<**script** type=**"text/javascript"**>  
 data={"上海":["浦东","浦西","黄埔","普陀"],"贵州":["贵阳","遵义","桐梓"],"辽宁":["锦州","沈阳","秦皇岛"]}; //object  
 var *pro\_ele* = *document*.getElementById('provinces');  
 var *city\_ele* = *document*.getElementById('citys');  
 city = null;  
 //添加省份  
 for(var *i* in data){  
 var *ele* = *document*.createElement('option');  
 *ele*.innerHTML=*i*;  
 *pro\_ele*.appendChild(*ele*);  
 }  
 //和省份进行联动  
 *pro\_ele*.onchange = function () {  
 *console*.log(this.selectedIndex)  
 *console*.log(this.options[this.selectedIndex])  
 var city2 = data[this.options[this.selectedIndex].innerHTML];  
 //保留 “请选择城市” option  
 *city\_ele*.options.length = 1;  
 for(var i=0;i<city2.length;i++){  
 var ele2 = *document*.createElement('option');  
 ele2.innerHTML = city2[i];  
 *city\_ele*.appendChild(ele2)  
 }  
 }  
</**script**>  
</**html**>

## 三 JS BOM对象

BOM（浏览器对象模型），可以对浏览器窗口进行访问和操作。使用 BOM，开发者可以移动窗口、改变状态栏中的文本以及执行其他与页面内容不直接相关的动作。

使 JavaScript 有能力与浏览器“对话”。

### 1.window对象

window对象

所有浏览器都支持 window 对象。

概念上讲.一个html文档对应一个window对象.

功能上讲: 控制浏览器窗口的.

使用上讲: window对象不需要创建对象,直接使用即可.

window对象方法：

alert() 显示带有一段消息和一个确认按钮的警告框。

confirm() 显示带有一段消息以及确认按钮和取消按钮的对话框。

prompt() 显示可提示用户输入的对话框。

open() 打开一个新的浏览器窗口或查找一个已命名的窗口。

close() 关闭浏览器窗口。

setInterval() 按照指定的周期（以毫秒计）来调用函数或计算表达式。

clearInterval() 取消由 setInterval() 设置的 timeout。

setTimeout() 在指定的毫秒数后调用函数或计算表达式。

clearTimeout() 取消由 setTimeout() 方法设置的 timeout。

scrollTo() 把内容滚动到指定的坐标。

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**BOM\_输出**</**title**>  
</**head**>  
<**body**>  
 <**button** onclick=**"**printInfo()**"**>**打印信息**</**button**>  
 <**button** onclick=**"**infoSearch()**"**>**查找**</**button**>  
</**body**>  
 <**script** type=**"text/javascript"**>  
 //1.js ECMAScript DOM BOM  
 //BOM Browser Object Model 浏览器对象模型  
 //核心 浏览器  
  
 // 输出 屏幕高度 滚动高度 setInterval .. window.open() close() location  
  
 //浏览器弹出对话框：如果不输入值，默认返回love china.修改则返回对应的值  
 var *pro* = prompt('中国很美','love china'); //prompt('msg',defaultValue);  
 *console*.log(*pro*);  
   
 // confirm  
 var *m* = confirm("学习BOM111"); //点击 确定返回True,点击取消，返回False  
 *console*.log(*m*);  
 //常用输出： alert console prompt confirm  
 //window.print 是打印页面  
 function printInfo() {  
 *window*.print();  
 }  
 function infoSearch() {  
 var m2 = confirm("查找");  
 find(m2)  
 }  
 </**script**>  
</**html**>

### 2.open 和 close 方法

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**BOM的open和close方法**</**title**>  
</**head**>  
<**body**>  
 <!--行间的js中的open window不能省略-->  
 <**button** onclick=**"***window*.open('https://foofish.net/')**"**>**python之禅**</**button**>  
 <**button**>**百度**</**button**>  
 <**button** onclick=**"***window*.close()**"**>**关闭**</**button**>  
 <**button**>**关闭**</**button**>  
</**body**>  
<**script** type=**"text/javascript"**>  
 var *oBtn* = *document*.getElementsByTagName('button')[1];  
 var *oClose* = *document*.getElementsByTagName('button')[3];  
 *oBtn*.onclick = function () {  
 // open('https://www.baidu.com')  
 //打开一个空白标签  
 //在当前页面打开用 \_self ，默认是新标签页打开 "\_target  
 open('about:blank',"\_self")  
 };  
 *oClose*.onclick = function () {  
 if(confirm("是否关闭")){  
 close()}  
 }  
</**script**>  
</**html**>

### 3.BOM其他对象

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**BOM的其他对象**</**title**>  
</**head**>  
<**body**>  
</**body**>  
<**script** type=**"text/javascript"**>  
 // 返回浏览器的用户设备信息  
 *console*.log(*window*.navigator.userAgent);  
 //返回：Mozilla/5.0 (Windows NT 10.0; Win64; x64) .....  
 *console*.log(*window*.location);  
  
 // 经常使用的一个方法  
 *window*.location.href = 'https://foofish.net/';  
  
 // 全局刷新,尽量少用 ==>对应局部刷新  
 // console.log("刷新了");  
 //  
 // setTimeout(function () {  
 // window.location.reload()  
 // },3000);  
</**script**>  
</**html**>

### 4.BOM的client系列

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**client系列**</**title**>  
 <**style** type=**"text/css"**>  
 .box{  
 *width*: 200**px**;  
 *height*: 200**px**;  
 *position*: **absolute**;  
 *border*: 10**px solid darkblue**;  
 *margin*: 10**px** 0 0 0;  
 *padding*: 10**px** 20**px** 30**px** 40**px**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
 <**div** class=**"box"**></**div**>  
</**body**>  
 <**script** type=**"text/javascript"**>  
 /\*stlye: top,left,right,bottom  
 \*/  
 /\*client  
 clientTop ：内容区域到边框顶部的距离  
 clientLeft ： 内容区域到边框左部得距离  
 clientWidth ：内容区域 + 左右 padding 可视宽度  
 clientHeight ： 内容区域 + 上下 padding 可视宽度  
 \*/  
 var *oBox* = *document*.getElementsByClassName('box')[0];  
 *console*.log(*oBox*.clientTop);  
 *console*.log(*oBox*.clientLeft);  
 *console*.log(*oBox*.clientWidth);  
 *console*.log(*oBox*.clientHeight);  
 </**script**>  
</**html**>

### 5.屏幕的可视区域

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**屏幕的可视区域**</**title**>  
</**head**>  
<**body**>  
</**body**>  
<**script** type=**"text/javascript"**>  
 *window*.onload = function () {  
 // console.log(document.documentElement.clientWidth);  
 // console.log(document.documentElement.clientHeight);  
 *window*.onresize = function () {  
 //动态获取屏幕宽度和高度  
 *console*.log(*document*.documentElement.clientWidth);  
 *console*.log(*document*.documentElement.clientHeight);  
 }  
 }  
</**script**>  
</**html**>

知识点:

1. window.onresize : onresize 事件会在窗口或框架被调整大小时发生。

### 6. offset系列

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**offset系列**</**title**>  
</**head**>  
<**body**>  
 <**div** style=**"***position*: **relative**;**"**>  
 <**div** id=**"box"** style=**"***width*: 200**px**;*height*: 200**px**;*border*: 1**px solid red**; *padding*: 10**px**;*margin*: 10**px**;*top*: 20**px**;*left*: 30**px**;*position*: **absolute"**></**div**>  
 </**div**>  
</**body**>  
<**script** type=**"text/javascript"**>  
 *window*.onload = function () {  
 var box = *document*.getElementById('box');  
 //占位宽，高，Top,Left  
 //offsetTop：如果盒子没有设置定位，到浏览器顶部的距离,如果盒子设置定位，那么是以父盒子为基准的top值  
 //offsetLeft：如果没有设置定位，到浏览器左部的距离,如果盒子设置定位，那么是以父盒子为基准的Left值  
 //offsetWidth： 内容+padding+border  
 //offsetHeight：内容+padding+border  
 *console*.log(box.offsetTop); //内容区域  
 *console*.log(box.offsetLeft);  
 *console*.log(box.offsetWidth);  
 *console*.log(box.offsetHeight)  
 }  
</**script**>  
</**html**>

7.scroll系列

*window*.onload = function () {  
 //实时监听浏览器窗口滚动事件  
 // document.documentElement.scrollTop 谷歌浏览器不能获取到值 恒为 0  
 *window*.onscroll = function () {  
 var wTop = (*document*.documentElement.scrollTop+*document*.body.scrollTop);  
 var wLeft = (*document*.documentElement.scrollLeft+*document*.body.scrollLeft);  
 var wWidth = (*document*.documentElement.scrollWidth); //body设置的宽  
 var wHeight = (*document*.documentElement.scrollHeight); //body的高  
 *console*.log('Top:'+ wTop);  
 *console*.log('Left:'+ wLeft);  
 *console*.log('width:'+ wWidth);  
 *console*.log('height:'+ wHeight);  
 };  
 //实时监听div 窗口滚动事件

var scro = *document*.getElementById('scro');  
 scro.onscroll = function () {  
 //scrollHeight :表示内容的高度+padding的高度(不包含margin)  
 *console*.log('Top:'+ scro.scrollTop);  
 *console*.log('Left:'+ scro.scrollLeft);  
 *console*.log('width:'+ scro.scrollWidth);  
 *console*.log('height:'+ scro.scrollHeight);  
  
 }  
};

# 四 jQuery

## 1.JavaScript 和 jQuery对比

1. Javascript是一门编程语言，我们用它来编写客户端浏览器脚本。

2. jQuery是javascript的一个库，包含多个可重用的函数，用来辅助我们简化javascript开发

3. jQuery能做的javascipt都能做到，而javascript能做的事情，jQuery不一定能做到

JavaScript

1.书写繁琐，代码量大

2.代码复杂

3.动画 开启定时器，小心定时器的清除，各种操作和处理时间 不好实现

4.浏览器兼容性

jquery-3.3.1.js :用在开发环境下，程序员使用。

jquery-3.3.1.min.js ： 压缩版本的。一般用在生产环境下。

代码对比：

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**example**</**title**>  
 <**style** type=**"text/css"**>  
 .div1{*width*: 200**px**;*height*: 30**px**;*display*: **none**;*background-color*: **red**;}  
 .div2{*width*: 200**px**;*height*: 30**px**;*display*: **none**;*background-color*: **yellowgreen**;}  
 </**style**>  
</**head**>  
<**body**>  
 <**button** id=**"btn1"**>**JS展示**</**button**>  
 <**button** id=**"btn2"**>**jQuery展示**</**button**>  
 <**div** class=**"div1"**></**div**>  
 <**div** class=**"div2"**></**div**>  
</**body**>  
<**script** src=**"./jquery-3.3.1.js"**></**script**>  
<**script** type=**"text/javascript"**>  
 //JavaScript实现  
 *window*.onload = function () {  
 var oBtn = *document*.getElementById('btn1');  
 var oDiv = *document*.getElementsByClassName('div1')[0];  
 oBtn.onclick = function () {  
 oDiv.style.display = 'block';  
 oDiv.innerHTML = 'JavaScript div 展示了';  
 }};  
 /\*jquery 实现上面的js代码\*/  
 $(function () {  
 $('#btn2').click(function () {  
 $('.div2').css('display','block');  
 $('.div2').html('jquery div 展示了');  
 })});  
</**script**>  
</**html**>

## 2.浏览器加载对象等

### 1.DOM文档加载的步骤

1. 解析HTML结构。
2. 加载外部脚本和样式表文件。
3. 解析并执行脚本代码。
4. DOM树构建完成。
5. 加载图片等外部文件。
6. 页面加载完毕。

### 2.window.onload 以及 $(document).ready() 的区别

1.执行顺序的不同

window.onload必须等到页面内包括图片的所有元素加载完毕后才能执行。

$(document).ready()是DOM结构绘制完毕后就执行，不必等到加载完毕。

2.加载方法的不同

window.onload不能同时编写多个，如果有多个window.onload方法，只会执行一个

$(document).ready()可以同时编写多个，并且都可以得到执行

3.简化写法

window.onload没有简化写法

$(document).ready(function(){})可以简写成$(function(){});

代码解释：

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**jquery的引入**</**title**>  
 <**script** type=**"text/javascript"**>  
 *window*.onload = function () {  
 //如果不写window.onload() 代码执行顺序，从上到下 此时的oDiv的值为 null  
 //如果写了window.onload() 那么此时的 oDiv的值为：﻿﻿﻿<div id="box"></div>  
 var oDiv = *document*.getElementById('box');  
 *console*.log(oDiv);  
 }  
 </**script**>  
</**head**>  
<**body**>  
 <**div** id=**"box"**></**div**>  
</**body**>  
</**html**>

/\*jquery 实现上面的js代码\*/  
// DOM元素加载完成之后就会调用  
//程序执行需要有入口 1.入口函数的方式  
$(*document*).ready(function () {  
 $('#btn2').click(function () {  
 $('.div2').css('display','block');  
 $('.div2').html('jquery div 展示了');  
 })});  
//等价于  
$(function () {  
 $('#btn2').click(function () {  
 $('.div2').css('display','block');  
 $('.div2').html('jquery div 展示了');  
 })});

## 3.jQuery基础选择器

jQuery的选择器的写法跟css的写法差不多。

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**jQuery基础选择器**</**title**>  
</**head**>  
<**body**>  
 <**ul**>  
 <**li** id=**"brother"** class=**"item"**>**路飞学诚1**</**li**>  
 <**li**><**a** href=**"https://www.luffycity.com"**>**路飞学诚2**</**a**></**li**>  
 <**li** class=**"item"**>**路飞学诚3**</**li**>  
 <**li**>**路飞学诚4**</**li**>  
 <**li** class=**"item"**>**路飞学诚5**</**li**>  
 <**li**>**路飞学诚6**</**li**>  
 <**li** class=**"item"**>**路飞学诚7**</**li**>  
 <**a** href=**"#"**>**百度一下**</**a**>  
 </**ul**>  
</**body**>  
<**script** src=**"jquery-3.3.1.min.js"**></**script**>  
<**script** type=**"text/javascript"**>  
 //使用jquery 的时候要有入口函数（回调函数）  
 $(*document*).ready(function () {  
 //基础选择器  
 //1.ID 选择器  
 *console*.log($('#brother'));  
 $('#brother').css('color','red');  
 //2.标签选择器  
 //设置一个值  
 // $('a').css('color','green');  
 //设置多个值，设置多个值使用对象存储， key:value  
 $('a').css({'color':'green','font-size':'24px'});  
 //3.类选择器  
 $('.item').css('background-color','yellowgreen')  
 //4.通配符选择器  
 // $('\*').html(''); //清空整个界面的dom元素  
 })  
</**script**>  
</**html**>

知识点：

1）4个选择器: 基础选择器，标签选择器，类选择器，通配符选择器

2）在使用jQuery的时候，最好有一个入口函数，$(document).ready(function () {}) 等价于 $(function () {})

## 4.jQuery层级选择器

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**jQuery层级选择器**</**title**>  
</**head**>  
<**body**>  
 <**div** id=**"box1"**>  
 <**p** id=**"p1"**>**锄禾日当午**</**p**>  
 <**p**>**汗滴禾下土**</**p**>  
 <**p**>**谁知盘中餐**</**p**>  
 <**p**>**粒粒皆辛苦**</**p**>  
 <**div** id=**"box2"**>  
 <**p**>**李白**</**p**>  
 </**div**>  
 </**div**>  
</**body**>  
<**script** src=**"jquery-3.3.1.min.js"**></**script**>  
<**script** type=**"text/javascript"**>  
 $(*document*).ready(function () {  
 //1.后代选择器  
 $('#box1 p').css('color','red');  
 //2.子代选择器  
 $('#box2>p').css('color','yellowgreen');  
 //3.毗邻选择器：匹配所有紧接着选中元素的兄弟  
 $('#p1+p').css('font-size','25px');  
 //4.兄弟选择器  
 $('#p1~p').css('font-size','30px');  
 //5.获取第一个元素  
 $('p:first').css('color','black');  
 //6.获取最后一个元素  
 $('p:last').css('color','blue');  
 //7.指定第几个标签元素 eq()  
 $('p:eq(2)').css('font-size','40px')  
 })  
</**script**>  
</**html**>

知识点：

1）后代选择器和子代选择器，子代选择器要使用“>”

2）获取第一个元素和最后一个元素 用 选择器：first 和“选择器”:last

3）获取指定的第一个标签元素，用eq(num) ,num 是从0 开始计算。

## 5.jQuery过滤选择器

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**jquery基本过滤选择器**</**title**>  
</**head**>  
<**body**>  
 <**ul**>  
 <**li**>**哈哈哈**</**li**>  
 <**li**>**嘿嘿嘿**</**li**>  
 <**li**>**天王盖地虎**</**li**>  
 <**li**>**小鸡炖蘑菇**</**li**>  
 </**ul**>  
</**body**>  
<**script** src=**"jquery-3.3.1.min.js"**></**script**>  
<**script** type=**"text/javascript"**>  
 $(function () {  
 //获取第一个 :first ,获取最后一个 :last  
 //奇数 从 0 开始计数  
 $('li:odd').css('color','red');  
 //偶数 从 0 开始计数  
 $('li:even').css('color','blue');  
 // 选中索引值为 1 的元素  
 $('li:eq(1)').css('font-size','24px');  
 // 选中索引值大于 1 的元素  
 $('li:gt(1)').css('background-color','yellowgreen');  
 //选中索引值小于 1 的元素  
 $('li:lt(1)').css('background-color','greenyellow')  
 })  
</**script**>  
</**html**>

## 6.jQuery属性选择器

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**jQuery属性选择器**</**title**>  
</**head**>  
<**body**>  
 <**div** id=**"box"**>  
 <**h2** class=**"title"**>**属性元素器**</**h2**>  
 <**p** class=**"p1"**>**我是一个段落**</**p**>  
 <**ul**>  
 <**li** id=**"li1"**>**分手应该体面1**</**li**>  
 <**li** class=**"what"** id=**"li2"**>**分手应该体面2**</**li**>  
 <**li** class=**"what"**>**分手应该体面3**</**li**>  
 <**li** class=**"heihei"**>**分手应该体面4**</**li**>  
 </**ul**>  
 <**form** action=**""** method=**"post"**>  
 <**input** name=**"username"** type=**'text'** value=**"1"** checked=**"checked"**></**input**>  
 <**input** name=**"username1111"** type=**'text'** value=**"2"**></**input**>  
 <**input** name=**"username2222"** type=**'text'** value=**"3"**></**input**>  
 <**input** name=**"username3333"** type=**'text'** value=**"4"**></**input**>  
 <**button** class=**"btn-default"**>**按钮1**</**button**>  
 <**button** class=**"btn-info"**>**按钮2**</**button**>  
 <**button** class=**"btn-success"**>**按钮3**</**button**>  
 <**button** class=**"btn-danger"**>**按钮4**</**button**>  
 </**form**>  
 </**div**>  
</**body**>  
<**script** src=**"jquery-3.3.1.min.js"**></**script**>  
<**script** type=**"text/javascript"**>  
 $(function () {  
 //标签名[属性名] 查找所有含有id属性的该标签名的元素  
 $('li[id]').css('color','red');  
 //[attr=value] 匹配给定的属性是某个特定值的元素  
 $('li[class=what]').css('font-size','24px');  
 //[attr!=value] 匹配所有不含有指定的属性，或者属性不等于特定值的元素  
 $('li[class!=what]').css('font-size','30px');  
 //匹配给定的属性是以某些值开始的元素  
 $('input[name^=username]').css({'display':'block','background-color':'yellowgreen'});  
 //匹配给定的属性是以某些值结尾的元素  
 $('input[name$=222]').css('background','green');  
 //匹配给定的属性是以包含某些值的元素  
 $('button[class\*=btn]').css('background','grey');  
 $('button[class\*=dan]').css('background','orange');  
 })  
</**script**>  
</**html**>

## 7.jQuery筛选选择器

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**jQuery筛选选择器**</**title**>  
</**head**>  
<**body**>  
 <**div** id=**"box"**>  
 <**p** class=**"p1"**>  
 <**span**>**我是第一个span标签**</**span**>  
 <**span**>**我是第二个span标签**</**span**>  
 <**span**>**我是第三个span标签**</**span**>  
 </**p**>  
 <**button**>**按钮**</**button**>  
 </**div**>  
 <**ul**>  
 <**li** class=**"list"**>**2**</**li**>  
 <**li**>**3**</**li**>  
 <**li**>**4**</**li**>  
 <**li**>**5**</**li**>  
 </**ul**>  
</**body**>  
<**script** src=**"jquery-3.3.1.min.js"**></**script**>  
<**script** type=**"text/javascript"**>  
 $(function () {  
 //点语法： get方法和 set方法  
 //获取第n个元素 数值从0开始  
 $('span').eq(1).css({'display':'block','background':'red'});  
 //first()获取第一个元素  
 $('span').first().css('color','yellowgreen')  
 //last()获取最后一个元素  
 $('span').last().css('color','green')  
 //.parent() 选择父亲元素 {parent('.p1')：精确找到父级元素}  
 $('span').parent('.p1').css({'width':'300px','height':'400px','background':'gray'})  
 //.siblings()选择所有的兄弟元素  
 $('.list').siblings('li').css('color','red');  
 //.find() 查找所有的后代元素  
 $('div').find('button').css('background','green');  
 })  
</**script**>  
</**html**>

知识点：

1）获取指定的第n个元素 eq(n) ; n 从0 开始计算

2）.parent() 选择父亲元素 {parent('.p1')：精确找到父级元素}  
3）.siblings()选择所有的兄弟元素

4）.find() 查找所有的后代元素

## 8.jQuery对象和DOM对象的区别与转换

### 1.区别

DOM对象是我们用传统的方法（javascript）获得的对象，jQuery对象就是用jQuery的类库选择器获得的对象。JQuery对象就是通过jQuery包装DOM对象后产生的对象。JQuery对象是jQuery独有的，其可以使用jQuery里的方法，但是不能使用DOM的方法；

例如：

$("#color").html();//获取id为color的元素内的html代码，html()是jQuery特有的方法;

它等同于: document.getElementById("color").innerHTML;

DOM对象就是javascript固有的一些对象。DOM对象能使用javascript固有的方法，但是不能使用jQuery里面的方法。

例如$("#id").innerHTML 和$("#id").checked之类的写法都是错误的，可以用$("#id").html()和$("#id").attr("checked")之类的 jQuery方法来代替。

### 2.转换

jQuery对象转换成DOM对象

jquery提供了两种方法将一个jquery对象转换成一个dom对象，即[index]和get(index)。

dom对象转换成jquery对象

对于一个dom对象，只需要用$()把dom对象包装起来，就可以获得一个jquery对象了，方法为$(dom对象);

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**jQuery对象和DOM对象转换**</**title**>  
</**head**>  
<**body**>  
 <**div** id=**"box"**>  
 **welcome to SH** </**div**>  
 <**button**>**隐藏**</**button**>  
</**body**>  
<**script** src=**"jquery-3.3.1.min.js"**></**script**>  
<**script** type=**"text/javascript"**>  
 $(function () {  
 // dom ==> jQuery对象  
 var oDiv = *document*.getElementById('box');  
 $(oDiv).click(function () { //此时的$(oDiv) 是jQuery对象  
 $('#box').css({'font-size':'24px','color':'red'})  
 });  
 // jQuery对象 ==> dom对象  
 //第一种方式 $('button')[0]  
 //第二种方式 $('button').get(0)  
 var isShow = true;  
 $('button').get(0).onclick = function () {  
 // $('#box').get(0).style.display = 'none'; //js 隐藏  
 if(isShow){  
 $('#box').hide(); // jquery 隐藏  
 *console*.log($(this)); //$(this) 是获取到 button的 jquery对象。  
 $(this).text('显示');  
 isShow = false;  
 }else {  
 $('#box').show();  
 $(this).text('隐藏');  
 isShow = true;  
 }}})  
</**script**>  
</**html**>

## 9.jQuery效果-隐藏和显示

show

概念:显示隐藏的匹配元素 语法：show(speed,callback) 参数：

speed：三种预定速度之一的字符串('slow','normal','fast')或表示动画时长的毫秒值（如：1000毫秒==1秒）

callback:在动画完成时执行的函数，每个元素执行一次

hide

hide(speed,callback)跟show使用方法类似，表示隐藏显示的元素。

可以通过show（）和hide（）方法，来动态控制元素的显示隐藏

toggle

如果元素是可见的，切换为隐藏的；如果元素是隐藏的，切换为可见的。

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**jQuery效果-隐藏和显示**</**title**>  
 <**style** type=**"text/css"**>  
 #box{  
 *width*: 100**px**;  
 *height*: 100**px**;  
 *border*: 1**px solid red**;  
 *display*: **none**;}  
 </**style**>  
</**head**>  
<**body**>  
 <**div** id=**"box"**></**div**>  
 <**button** id=**"btn"**>**显示**</**button**>  
</**body**>  
<**script** src=**"jquery-3.3.1.min.js"**></**script**>  
<**script** type=**"text/javascript"**>  
 //显示隐藏的集中办法  
 // 第一种方法：通过css方法进行隐藏和显示  
 $('#btn').click(function () {  
 $('#box').css('display','block');  
 });  
 // 第二种方法：jquery 提供了一些方法 show() hide() 控制元素显示 隐藏  
 var *isShow* = true;  
 $('#btn').click(function () {  
 if(*isShow*){  
 $('#box').show('slow',function () {  
 $(this).text('盒子出来了');  
 $('#btn').text('隐藏');  
 *isShow* = false;  
 })}else {  
 $('#box').hide(2000,function () {  
 $('#btn').text('显示');  
 *isShow* = true;  
 })}});  
 // toggle 开关 如果元素显示则隐藏，反之则显示  
 // 此方法建议只对元素开关，不做别的操作。  
 $('#btn').click(function () {  
 $('#box').toggle(3000,function () {  
 $('#box').text('toggle开关');  
 });})  
</**script**>  
</**html**>

知识点：

1）css隐藏一个block标签 通过 display=’none’ 来隐藏、

2）隐藏的jQuery对象 hide() 和show()

3) toggle 相当于一个开关，toggle也可以通过控制时间去隐藏显示

4）jQuery对象都是通过对象的透明度进行隐藏的。

## 10.jQuery效果-隐藏和显示2

**slideDown**

概念：通过高度变化（向下增大）来到动态地显示所有匹配的元素，在显示完成后触发一个回调函数

用法和参数跟（show，hide）类似

**slideUp**

通过高度变化（向上减小）来动态地隐藏所有匹配的元素，在隐藏完成后可选地触发一个回调函数。

用法和参数跟（show，hide）类似

**slideToggle**

概念:通过高度变化来切换所有匹配元素的可见性，并在切换完成后可选地触发一个回调函数

跟toggle用法类似

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**jQuery效果-slide**</**title**>  
 <**style** type=**"text/css"**>  
 #box{  
 *width*: 100**px**;  
 *height*: 100**px**;  
 *border*: 1**px solid red**;  
 *display*: **none**;}  
 </**style**>  
</**head**>  
<**body**>  
 <**div** id=**"box"**></**div**>  
 <**button** id=**"btn"**>**显示**</**button**>  
</**body**>  
<**script** src=**"jquery-3.3.1.min.js"**></**script**>  
<**script** type=**"text/javascript"**>  
 $(function () {  
 // 第一种方法  
 var isShow = true;  
 $('#btn').click(function () {  
 if(isShow){  
 $('#box').slideDown(2000);  
 $('#box').html('slideDown');  
 $('button').html('隐藏');  
 isShow = false;  
 }else {  
 $('#box').slideUp(1000);  
 $('button').html('显示');  
 isShow = true;  
 }});  
 //第二种方法  
 $('#btn').click(function () {  
 $('#box').slideToggle()  
 })  
 })  
</**script**>  
</**html**>

## 11.jQuery效果-淡入淡出fade

fadeOut

概念：通过不透明度的变化来实现所有匹配元素的淡出效果，并在动画完成后可选地触发一个回调函数。

这个动画只调整元素的不透明度，也就是说所有匹配的元素的高度和宽度不会发生变化。

fadeTo

概念：把所有匹配元素的不透明度以渐进方式调整到指定的不透明度，并在动画完成后可选地触发一个回调函数。

fadeToggle

概念：通过不透明度的变化来开关所有匹配元素的淡入和淡出效果，并在动画完成后可选地触发一个回调函数。

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**jQuery效果-fade**</**title**>  
 <**style** type=**"text/css"**>  
 #box{  
 *width*: 100**px**;  
 *height*: 100**px**;  
 *border*: 1**px solid red**;  
 *background-color*: #84a42b;}  
 </**style**>  
</**head**>  
<**body**>  
 <**div** id=**"box"**></**div**>  
 <**button** id=**"btn"**>**隐藏**</**button**>  
</**body**>  
<**script** src=**"jquery-3.3.1.min.js"**></**script**>  
<**script** type=**"text/javascript"**>  
 $(function () {  
 //淡入淡出  
 $('#btn').mouseover(function () {  
 // $('#box').fadeOut(2000);  
 $('#box').fadeTo(2000,0.5);  
 });  
 $('#btn').mouseout(function () {  
 // $('#box').fadeIn(2000);  
 $('#box').fadeTo(2000,1); //fadeTo(时间，透明度)  
 });  
 // $('#btn').click(function () {  
 // $('#box').fadeToggle(3000)  
 // })  
 })  
</**script**>  
</**html**>

## 12.jQuery效果-animate动画

animate

概念：用于创建自定义动画的函数

语法：animate(params,[speed],[fn])

参数：

params:一组包含作为动画属性和终值的样式属性和及其值的集合

speed:三种预定速度之一的字符串("slow","normal", or "fast")或表示动画时长的毫秒数值(如：1000)

fn:在动画完成时执行的函数，每个元素执行一次。

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**JQuery效果-animate(动画)**</**title**>  
 <**style** type=**"text/css"**>  
 #box{  
 *width*: 100**px**;  
 *height*: 100**px**;  
 *border*: 1**px solid red**;  
 *background-color*: #84a42b;  
 *position*: **absolute**;}  
 </**style**>  
</**head**>  
<**body**>  
 <**button** id=**"btn"**>**动画吧**</**button**>  
 <**button** id=**"btn2"**>**停止吧**</**button**>  
 <**div** id=**"box"**>  
 **hello everybody！** </**div**>  
</**body**>  
<**script** src=**"jquery-3.3.1.min.js"**></**script**>  
<**script** type=**"text/javascript"**>  
 $(function () {  
 $('#btn').click(function () {  
 //第一种：直接修改样式  
 $('#box').animate({  
 width:'200px',  
 height:'200px'});  
 //第二种： 动画：先右移动，在往下移动  
 $('#box').animate({  
 left:'50px'  
 }).animate({top:'100px'});  
 //第三种：往右和下同同时移动，耗时5秒  
 $('#box').animate({  
 left:'100px',  
 top:'100px'  
 },5000);  
 //第四种：先右移动，然后延时3秒，然后在往下移动  
 //delay 延迟  
 $('#box').animate({  
 left:'50px'  
 }).delay(3000).animate({top:'100px'});  
 //停止动画 stop  
 $('#btn2').click(function () {  
 $('#box').stop();  
 })})})  
</**script**>  
</**html**>

## 13.jQuery之右下角弹出小广告

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**右下角弹出小广告**</**title**>  
 <**style** type=**"text/css"**>  
 #box{  
 *width*: 330**px**;  
 *height*: 480**px**;  
 *position*: **absolute**;  
 *right*: 10**px**;  
 *bottom*: 0;  
 *display*: **none**;  
 }  
 .img1{  
 *width*: 100%;  
 *height*: 100%;  
 }  
 </**style**>  
</**head**>  
<**body**>  
<**div** id=**"box"**>  
 <**img** class=**"img1"** src=**"./广告.png"** alt=**""**>  
</**div**>  
</**body**>  
<**script** src=**"jquery-3.3.1.min.js"**></**script**>  
<**script** type=**"text/javascript"**>  
 $(function () {  
 $('#box').slideDown('normal').fadeIn(1000).click(function () {  
 $(this).fadeOut(1000);  
 })  
 })  
</**script**>  
</**html**>

## 14.jQuery的属性操作

jquery的属性操作模块分为四个部分：html属性操作，dom属性操作，类样式操作和值操作

html属性操作：是对html文档中的属性进行读取，设置和移除操作。比如attr()、removeAttr()

DOM属性操作：对DOM元素的属性进行读取，设置和移除操作。比如prop()、removeProp()

类样式操作：是指对DOM属性className进行添加，移除操作。比如addClass()、removeClass()、toggleClass()

值操作：是对DOM属性value进行读取和设置操作。比如html()、text()、val()

### 1.html操作-attr & removeAttr

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**jQuery的属性操作-attr** </**title**>  
</**head**>  
<**body**>  
 <**div** id=**"box"**>  
 <**p**>**路飞学城**</**p**>  
 </**div**>  
 <**button**>**获取**</**button**>  
</**body**>  
<**script** src=**"jquery-3.3.1.min.js"**></**script**>  
<**script** type=**"text/javascript"**>  
 $(function () {  
 //jQuery 属性操作  
 //attr() 如果有一个参数，表示获取值  
 $('button').click(function () {  
 $('#box p').text($('#box').attr('id'))  
 });  
 //attr() 如果设置两个值 表示设置属性  
 //设置多个值用对象存储，如果设置多个类名 不能使用attr()<会覆盖>  
 $('#box').attr('class','abc');  
 $('#box').attr({'type':'xls','name':'div'});  
 //删除一个属性 removeattr()  
 $('#box').removeAttr('name');  
 //删除多个属性值  
 $('#box').removeAttr('type class');  
 })  
</**script**>  
</**html**>

### 2.DOM操作-prop

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**jQuery的属性操作-prop**</**title**>  
</**head**>  
<**body**>  
 <**ul**>  
 <**li** class=**"luffy"**>**luffy1**</**li**>  
 <**li** class=**"luffy"**>**luffy2**</**li**>  
 </**ul**>  
</**body**>  
<**script** src=**"jquery-3.3.1.min.js"**></**script**>  
<**script** type=**"text/javascript"**>  
 $(function () {  
 //DOM属性操作  
 *console*.log($('li').prop('class')) //luffy  
 //设置值  
 $('li').first().prop({'name':'app','name2':'app2'});  
 *console*.log($('li').first());  
 //删除dom对象的name属性  
 $('li').first().removeProp('name');  
 *console*.log($('li').prop('name')); //undefined  
 })  
</**script**>  
</**html**>

### 3.类样式操作

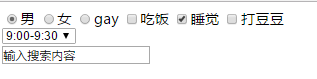
<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**jQuery属性操作-calss和值操作**</**title**>  
 <**style** type=**"text/css"**>  
 span.active{  
 *font-size*: 30**px**;  
 *color*: **red**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
 <**span** class=**"span-1"** style=**"***display*: **block**;**"**>**路飞吧！**</**span**>  
 <**span** class=**"span-2"** style=**"***display*: **block**;**"**>**学习吧！**</**span**>  
</**body**>  
<**script** src=**"jquery-3.3.1.min.js"**></**script**>  
<**script** type=**"text/javascript"**>  
 $(function () {  
 //addClass() removeClass()  
 //添加class  
 $('span').addClass('span2 span3');  
 //删除class  
 $('span').removeClass('span2 span3');  
 //添加类事件操作  
 //点击标签添加类，类赋予css属性  
 var isBig = true;  
 $('.span-1').click(function () {  
 if(isBig){  
 $(this).addClass('active');  
 isBig = false  
 }else {  
 $(this).removeClass('active');  
 isBig = true  
 }})  
 //移动鼠标添加类，类赋予css属性  
 $('.span-2').mouseover(function () {  
 $(this).addClass('active');  
 });  
 $('.span-2').mouseout(function () {  
 $(this).removeClass('active');  
 });  
 })  
</**script**>  
</**html**>

### 4.值操作

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**jQuery属性操作-值操作**</**title**>  
</**head**>  
<**body**>  
<**div** id=**"box2"**>  
 **hello world!** <**p**>**我是一个段落**</**p**>  
 <**a** href=**"#"**>**百度一下**</**a**>  
 <**input** type=**"text"** value=**"input"** name=**""**>  
 </**div**>  
</**body**>  
<**script** src=**"jquery-3.3.1.min.js"**></**script**>  
<**script** type=**"text/javascript"**>  
 $(function () {  
 //4.值属性的操作 text() html() val()  
 // text 仅仅是获取文本，设置值的时候也是文本，  
 // html 获取所有的元素和文本，设置的时候也是html样式  
 // $('#box2').text('<a href="#">百度一下</a>'); //<a href="#">百度一下</a>  
 // $('#box2').html('<a href="#">百度一下</a>'); //链接  
 //val 获取值  
 *console*.log($('input').val()); //input 获取到的是 input标签属性的value值  
 //设置input 的value值  
 $('input').val('搜索查询');  
 })  
</**script**>  
</**html**>

## 15.input中的value值

界面显示如下：



<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**input中的value中的值**</**title**>  
</**head**>  
<**body**>  
 <**form** action=**""**>  
 <**input** type=**"radio"** name=**"sex"** value=**"12"** />**男** <**input** type=**"radio"** name=**"sex"** value=**"13"** checked=**""** />**女** <**input** type=**"radio"** name=**"sex"** value=**"14"** />**gay** <**input** type=**"checkbox"** value=**"a"** checked=**""**/>**吃饭** <**input** type=**"checkbox"** value=**"b"** checked=**""**/>**睡觉** <**input** type=**"checkbox"** value=**"c"** checked=**""**/>**打豆豆** <**select** name=**"timespan"** id=**"timespan"** class=**"Wdate"** style=**"***display*: **block**;**"** >  
 <**option** value=**"select1"**>**8:00-8:30**</**option**>  
 <**option** value=**"select2"**>**8:30-9:00**</**option**>  
 <**option** value=**"select3"**>**9:00-9:30**</**option**>  
 </**select**>  
 <**input** type=**"text"** name=**""** id=**""** value=**"111"** />  
 </**form**>  
</**body**>  
<**script** src=**"jquery-3.3.1.min.js"**></**script**>  
<**script** type=**"text/javascript"**>  
 $(function () {  
 //取出 type 类型的标签，还可以取出：checkbox button submit file text disabled  
 // console.log($(':radio')) //取出 type=radio 的标签  
 //1.获取单选框里面的value的值。  
 *console*.log($('input[type=radio]:checked').val()) //13  
 //2.获取复选框里面的value的值  
 // 可以看到复选框里面有多个checked，但是仅仅获取第一个值  
 *console*.log($('input[type=checkbox]:checked').val()) // a  
 //3.下拉列表被选中的值;如果再option中不设置默认值，那么取第一个，如果设置了默认值，取出默认值。  
 var obj = $('#timespan option:selected')  
 *console*.log(obj.val()) // select1  
  
 //设置值  
 //1.设置单选的值  
 $('input[type=radio]').val(['12']) //12对应男，页面默认勾选 男  
 $('input[type=checkbox]').val(['b']) // 页面勾选 睡觉 ，虽然html里面都设置了checked，这段代码相当于重置了checked  
 //2.设置下拉列表选中的值，这里必须要用select  
 $('select').val(['select3']) //9:00-9:30  
 //3.text 获取值和设置值  
 *console*.log($(':text').val()) //默认显示value的值 111  
 $(':text').val('输入搜索内容') //替换value='111' 为 value='输入搜索内容'  
 })  
</**script**>  
</**html**>

知识点：

1.input的标签：radio为单选框，text：文本框，checkbox：复选框，默认选中为 checked=“”

2.select与option：下拉框，设置默认值的用selected="selected"

3.根据input标签类型获取val值。 $('input[type=radio]:checked').val()，$('input[type=text]').val()

4.获取select>option下的默认值 $('#timespan option:selected')

5.设置值：

5.1．设置单选框的值 ：$('input[type=radio]').val(['12'])

5.2．设置下拉列表的值：$('select').val(['select3'])

5.3．设置text类型的值：$(':text').val('输入搜索内容') 等价于$('input[type=text]').val('搜索内容')

## 16.jQuery文档操作

### 1.插入节点

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**jQuery操作-插入节点**</**title**>  
</**head**>  
<**body**>  
 <**span** class=**"span1"** style=**"***display*: **block"**>**hello,world!**</**span**>  
 <**ul**></**ul**>  
</**body**>  
<**script** src=**"jquery-3.3.1.min.js"**></**script**>  
<**script** type=**"text/javascript"**>  
 $(function () {  
 var oLi = *document*.createElement('li');  
 oLi.innerHTML='路飞';  
 //追加元素  
 $('ul').append('<li><a href="#">luffy</a></li>');  
 $('ul').append(oLi);  
 //如果直接的内容是当前页面中的某些元素，那么这些元素将从原位置上消失。简言之，就是一个移动操作  
 $('ul').append($('.span1'));  
 //appendTo()  
 $('<a href="#">呵呵一下</a>').appendTo($('ul'));  
 // prepend() 插入到被插元素得第一个位置  
 $('ul').prepend('<li>插入到第一位置</li>');  
 $('<li>插入到第0位置</li>').prependTo($('ul'));  
 // before after  
 $('ul').before('<h2>我是一个二级标题</h2>');  
 $('ul').after('<h3>我是一个三级标题</h3>');  
 //insertAfter  
 $('<a href="#">insertAfter插入</a>').insertAfter($('ul'))  
 $('<a href="#">insertBefore插入</a>').insertBefore($('ul'))  
 })  
</**script**>  
</**html**>

**知识点：**

1.创建元素标签并赋值：oLi = *document*.createElement('li'); （js的操作）；oLi.innerHTML='路飞';

2.追加元素到指定的选择器位置：

追加元素的4个方法的区别：

append() - 在被选元素的结尾插入内容 :在上列中是直接移动页面标签元素到指定位置，元素将从原位置消失；

appendTo() - 在被选元素的结尾（仍然在内部）插入指定内容。

prepend() - 在被选元素的开头插入内容

prependTo() -在被选元素的开头（仍然在内部）插入指定内容。

after() - 在被选元素之后插入内容 （选择器在前面，查看例子）

before() - 在被选元素之前插入内容 （选择器在前面，查看例子）

insertBefore() - 在被选元素之后插入内容 （选择器在后面，查看例子）

insertAfter() -在被选元素之前插入内容 （选择器在后面，查看例子）

### 2.复制文档-clone

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**jQuery文档操作-复制-替换-删除**</**title**>  
</**head**>  
<**body**>  
 <**span** style=**"***display*: **block"**>**hello,world!**</**span**>  
 <**ul**></**ul**>  
 <**button**>**按钮**</**button**>  
</**body**>  
<**script** src=**"jquery-3.3.1.min.js"**></**script**>  
<**script** type=**"text/javascript"**>  
 $(function () {  
 //复制  
 $('button').click(function () {  
 // 1.clone()：克隆匹配的DOM元素并且选中这些克隆的副本。  
 // 2.clone(true)：元素以及其所有的事件处理并且选中这些克隆的副本(简言之，副本具有与真身一样的事件处理能力)  
 $(this).clone(true).insertAfter($(this))  
 });})  
</**script**>  
</**html**>

注意点：

clone() : 的默认值是false ：clone(false)，当默认值改为true：clone(true)的时候就表示，复制的时候把事件也一起进行复制

### 3.替换-replaceWith

<**script** type=**"text/javascript"**>  
 $(function () {  
 $('ul').after('<h3>我是一个三级标题</h3>');  
 //替换  
 $('h3').replaceWith('<p style="color: red">我是一个p标签</p>');  
 $('<a href="#">替换的超链接</a>').replaceAll('button');  
 })  
</**script**>

知识点：

1.替换replaceWith 和replaceAll差不多，都可以替换一个标签或者替换多个标签，根据选择器进行替换

### 4.删除

<**script** type=**"text/javascript"**>  
 $(function () {  
 $('ul').append('<li><a href="#">luffy</a></li>');  
 $('button').click(function () {  
 $(this).clone(true).insertAfter($(this))  
 });  
 //删除  
 //empty():清空元素中的所有后代节点  
 //清空掉ul中的子元素，保留ul  
 // $('ul').empty()  
 //remove():删除节点后，事件也会删除（简言之，删除了整个标签）  
 // $('ul').remove();  
 //detach()：删除节点后，事件会保留  
 var $oldBtn = $('button');  
 $oldBtn.detach();  
 $oldBtn.text('新增btn');  
 $('ul').after($oldBtn[0])  
 })  
</**script**>

知识点：

1.删除的方法有：empty() ,remove() ,detach() ，具体的区别在例子中已说明。

## 17.jQuery的位置属性

html为：

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**jQuery的位置属性**</**title**>  
 <**style** type=**"text/css"**>  
 \*{*padding*: 0;*margin*: 0;}  
 #box1{*position*: **relative**;*width*: 200**px**;*height*: 200**px**;*border*: 1**px solid red**;*padding*: 10**px** 5**px**;}  
 #box2{*width*:200**px**;*height*:200**px**;*margin-top*:20**px**;*border*: 1**px solid deepskyblue**;}  
 p{*position*: **absolute**;*left*:30**px**;*top*:30**px**}  
 </**style**>  
</**head**>  
 <**body** style=**"***height*: 2000**px**; *width*: 2000**px**;**"**>  
 <**div** id=**"box1"**>  
 <**p**>**我是一个段落标签**</**p**>  
 </**div**>  
 <**button** id=**"btn"**>**动画吧**</**button**>  
 <**div** id=**"box2"** style=**""**></**div**>  
</**body**>  
</**html**>

### 1.位置属性-position

$(function () {  
 //1.获取匹配元素的先对父元素的偏移  
 //position 该方法返回的对象包含两个整型属性：top 和 left，以像素计  
 *console*.log($('p').position().left); // 30  
 *console*.log($('p').position().top); // 30  
 var offsetTop = $('p').position().top + 50 +'px';  
 $('#btn').click(function () {  
 $('p').animate({top:offsetTop},1000)  
 });  
})

### 2.获取匹配元素滚动条卷起的位置

$(function () {  
 // 2.获取匹配元素 相对滚动条卷起的位置信息 scrollTop scrollLeft  
 $(*document*).scroll(function () {  
 *console*.log($(this).scrollLeft());  
 *console*.log($(this).scrollTop());  
 });  
})

### 3.获取匹配元素在当前视口的相对偏移

$(function () {  
 //3.offset 获取匹配元素在当前视口的相对偏移,返回的对象包含两个整型属性：top 和 left （相对于浏览器）  
 *console*.log($('#box2').offset());  
 *console*.log($('#box2').offset().top);  
 *console*.log($('#box2').offset().left);  
})

注意点：主要注意“视口” 表示的意思是 屏幕可见区域内；假如 #box2 的位置是 fixed，那么他的视口就是一个固定的值。

### 4.获取元素宽高

$(function () {  
 //4.获取元素宽高  
 *console*.log("宽"+$('#box2').width());  
 *console*.log("高"+$('#box2').height());  
})

### 5.设置元素宽高

$(function () {  
 //5.设置元素宽高  
 $('#box2').width(300);  
 $('#box2').height(300);  
 //innerWidth // outerWidth  
 //获取元素的内部宽度 innerWidth = 宽度 + 2 \* padding  
 //获取元素的外部宽度 outerWidth = 宽度 + 2 \* padding + 2 \* border  
 $('#box2').innerWidth(500);  
 *console*.log("宽"+$('#box').width()); //宽490  
 *console*.log($('#box').innerWidth());  
})

知识点：

width()仅仅包括content(内容)

innerWidth()包括content(内容)和padding(内边距)

outerWidth()包括content(内容),padding(内边距)和border(边框)

outerWidth(true)包括content(内容),padding(补白),border(边框)和margin(外边距)

获取元素的内部宽度 innerWidth = 宽度 + 2 \* padding

获取元素的外部宽度 outerWidth = 宽度 + 2 \* padding + 2 \* border

## 18.仿淘宝搜索框-scroll应用

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**仿淘宝搜索框案例**</**title**>  
 <**style** type=**"text/css"**>  
 \*{*padding*: 0;*margin*: 0;}  
 .header{*width*: 100%;*height*: 50**px**;}  
 .header p{  
 *height*: 50**px**;  
 *line-height*: 50**px**;  
 *text-align*: **center**;  
 *font-size*: 30**px**;  
 }  
 .content{*width*: 100%;*height*: 50**px**;}  
 .content .search{  
 *width*: 40%;  
 *height*: 30**px**;  
 *margin-left*: 30%;  
 *border*: 1**px solid black**;  
 *border-radius*: 50**px**;  
 *outline*: **none**;  
 }  
 </**style**>  
 </**head**>  
<**body** style=**"***width*: 100%;*height*: 2000**px"**>  
 <**div** class=**"header"**>  
 <**p**>**仿淘宝搜索框案例**</**p**>  
 </**div**>  
 <**div** class=**"content"**>  
 <**input** class=**"search"** type=**"text"** value=**"请输入搜索内容"**>  
 </**div**>  
</**body**>  
<**script** src=**"jquery-3.3.1.min.js"**></**script**>  
<**script** type=**"text/javascript"**>  
 $(function () {  
 var h = $('.header').height();  
 $(*document*).scroll(function () {  
 var scrollTOP =$(*document*).scrollTop();  
 if(h<scrollTOP){  
 $('.content').css({'position':'fixed','top':'0'})  
 }else {  
 $('.content').css({'position':'static','top':'0'})  
 }})  
 })  
</**script**>  
</**html**>

知识点：

1.获取元素标签高度 height()

2.获取滚动的document到顶部的距离 scrollTop()

## 19.jQuery的筛选方法

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**jQuery的筛选方法**</**title**>  
 <**style** type=**"text/css"**>  
 .active{*font-size*: 30**px**;*color*: **red**;}  
 </**style**>  
</**head**>  
<**body**>  
</**html**>

### 1.children()

$(function () {  
 $('ul').children().each(function (*index*,*ele*) {  
 var name = $(this).hasClass('one');  
 if(name){  
 $(this).css({'color':'red'})  
 }else {  
 $(this).css({'font-size':'30px'})  
 }});  
})

知识点：

1. $('ul').children() 相当于 $('li')

2．$('ul').children().each() ：是遍历所有的子节点

3.hasClass ：检查被选元素是否包含指定的 class

*console*.log($('ul').children('.num'));

这一句代码表示的意思是，获取ul子节点下面，包含了 .num 类选择器的 子节点标签。

### 2.parent()

*console*.log($('a').parent());

含义是：a节点的所有父亲节点，结果为：获得

<**li** class=**"four num"**><**a** href=**"#"**>**4**</**a**></**li**>

<**ul**></ul>

这个可以通过index获取指定的标签。

### 3.prev() 和 prevAll()

prev() 获得匹配元素集合中每个元素紧邻的前一个同胞元素，通过选择器进行筛选是可选的。

prevAll() 获得当前匹配元素集合中每个元素的前面的同胞元素，使用选择器进行筛选是可选的。

$(function () {  
 *console*.log($('li').last().prev()); // li.three  
 *console*.log($('li').last().prevAll()); //li.three, li.two.num, li.one  
})

上面两个相似的代码表示的意思是，获取 li标签选择器的最后一个li标签，然后去执行方法。

### 4.siblings()

siblings() 获得匹配集合中每个元素的同胞，通过选择器进行筛选是可选的。

$(function () {  
 //siblings(pramas) 参数不同获取的标签不同  
 *console*.log($('.five').siblings('li')); // li.one, li.two.num, li.three, li.four.num  
 *console*.log($('.five').siblings('a')); // 没有兄弟元素  
 //  
 $('li').hover(function () {  
 $(this).addClass('active').siblings('li').removeClass('active')  
 })  
})

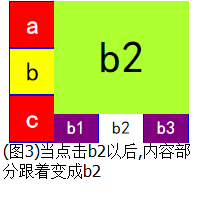
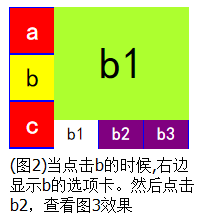
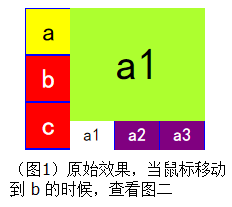
知识点：

1.选择器.siblings(标签名称)：注意后面获取标签的元素是什么。

2.addClass ，removeClass 给标签添加类 和删除类

## 20.案例-选项卡嵌套

效果如下：



<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**22-案例-选项卡嵌套**</**title**>  
 <**style** type=**"text/css"**>  
 \*{*padding*: 0;*margin*: 0;}  
 ul{*list-style*: **none**;}  
 /\*清除浮动产生的问题\*/  
 #box:after{*content*: "";*display*: **block**;*clear*: **both**;}  
 #box{*width*: 200**px**;*margin*: 20**px auto**;*background*: **blue**;}  
 #leftBox{*width*: 50**px**;*float*: **left**;}  
 #leftBox li{*width*: 50**px**;*height*: 50**px**;*background*: **red**;*border*:0.1**px solid blue**;*color*: **white**;*font*: 30**px**/50**px** "黑体";*text-align*: **center**;}  
 #rightBox div{*display*: **none**;*float*: **left**; *width*: 150**px**;}  
 #rightBox p{*width*: 100%;*height*: 125**px**;*font*: 50**px**/125**px** "黑体";*text-align*: **center**;*background-color*: **greenyellow**}  
 /\*!\*父元素设置display：table使它成为一个块级表格元素\*/  
 /\*子元素设置display：table-cell使子元素成为表格单元格，就好比是在表格中一样\*!\*/  
 #rightBox ul{*width*: 150**px**;*display*: **table**;}  
 #rightBox li{*display*: **table-cell**;*background*: **purple**;*height*: 32**px**;*border-right*: 2**px solid blue**;*font*: 20**px**/32**px** "黑体";*text-align*: **center**;*color*: **white**;}  
 #leftBox .active{*background*: **yellow**;*color*: **black**;}  
 #rightBox .active{*background*: **white**;*color*: **black**;}  
 </**style**>  
 </**head**>  
<**body**>  
 <**div** id=**"box"**>  
 <**ul** id=**"leftBox"**>  
 <**li** class=**"active"**>**a**</**li**>  
 <**li**>**b**</**li**>  
 <**li**>**c**</**li**>  
 </**ul**>  
 <**div** id=**"rightBox"**>  
 <**div** style=**"***display*: **block"**>  
 <**p**>**a1**</**p**>  
 <**ul**>  
 <**li** class=**"active"**>**a1**</**li**>  
 <**li**>**a2**</**li**>  
 <**li**>**a3**</**li**>  
 </**ul**>  
 </**div**>  
 <**div**>  
 <**p**>**b1**</**p**>  
 <**ul**>  
 <**li** class=**"active"**>**b1**</**li**>  
 <**li**>**b2**</**li**>  
 <**li**>**b3**</**li**>  
 </**ul**>  
 </**div**>  
 <**div**>  
 <**p**>**c1**</**p**>  
 <**ul**>  
 <**li** class=**"active"**>**c1**</**li**>  
 <**li**>**c2**</**li**>  
 <**li**>**c3**</**li**>  
 </**ul**>  
 </**div**>  
 </**div**>  
 </**div**>  
</**body**>  
<**script** src=**"jquery-3.3.1.min.js"**></**script**>  
<**script** type=**"text/javascript"**>  
 $(function () {  
 $('#leftBox li').mouseover(function () {  
 $(this).addClass('active').siblings('li').removeClass('active');  
 *console*.log($(this).index())  
 $('#rightBox div').eq($(this).index()).show().siblings('div').hide();  
 });  
 $('#rightBox li').click(function () {  
 $(this).addClass('active').siblings('li').removeClass('active');  
 $(this).parent().prev().html($(this).html())  
 })  
 })  
</**script**>  
</**html**>

知识点：

1. font: 20px/32px ：20px 表示字体大小；32px 表示行高；

2.addClass，removeClass 添加类，删除类

3.$(this).index() ：表示某个选择器里面的索引的值，从0开始计数。

4.show(),hide() ：显示/隐藏

5.eq(index) ：取值。

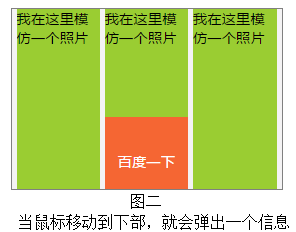
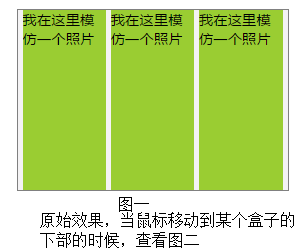
6.本案例赋值效果：

$(this).parent().prev().html($(this).html())

含义：当点击这个元素的时候，先获取该元素的父标签的前一个同胞元素的 value值。然后在赋值给点击的这个元素value。

## 21.案例-小米官网部分效果

效果如下：



<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**23-案例-小米官网部分效果**</**title**>  
 <**style** type=**"text/css"**>  
 \*{*padding*: 0;*margin*: 0;}  
 ul{*list-style*: **none**;}  
 .wrap{*width*: 300**px**;*height*: 200**px**;*margin*: 20**px auto** 0;*background*: #f4f3f4;*border*: 1**px solid gray**;}  
 ul li{*float*: **left**;*margin-left*: 5**px**;*position*: **relative**;*overflow*: **hidden**;*width*: 93**px**;*height*: 200**px**;*background-color*: **yellowgreen**}  
 ul li p{  
 *width*: 93**px**;  
 *height*: 80**px**;  
 *background*: #F56633;  
 *position*: **absolute**;  
 *bottom*: -100**px**;  
 *text-align*: **center**;  
 *color*: **white**;  
 *line-height*: 100**px**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
<**div** class=**"wrap"**>  
 <**ul**>  
 <**li**>**我在这里模仿一个照片**<**p**>**百度一下**</**p**></**li**>  
 <**li**>**我在这里模仿一个照片**<**p**>**百度一下**</**p**></**li**>  
 <**li**>**我在这里模仿一个照片**<**p**>**百度一下**</**p**></**li**>  
 </**ul**>  
</**div**>  
</**body**>  
<**script** src=**"jquery-3.3.1.min.js"**></**script**>  
<**script** type=**"text/javascript"**>  
 $(function () {  
 $('.wrap li').hover(function () {  
 $(this).children('p').stop(true).animate({bottom:0},400)  
 },function () {  
 $(this).children('p').stop(true).animate({bottom:-100},50)  
 })  
 })  
</**script**>  
</**html**>

知识点：

1.stop() : 被选元素停止当前正在运行的动画

## 22.案例-焦点式轮播图

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**24-案例-焦点式轮播图**</**title**>  
 <**style** type=**"text/css"**>  
 \*{*padding*: 0;*margin*: 0;}  
 ul,ol{*list-style*: **none**;}  
 #wrap{*width*: 650**px**;*height*: 250**px**;*margin*: 100**px auto** 0;*background*: **red**;*overflow*: **hidden**;*position*: **relative**;}  
 #wrap ul{*height*: 250**px**;*position*: **relative**;*z-index*: 1;}  
 #wrap ol{*height*: 30**px**;*position*: **absolute**;*z-index*: 10;*bottom*: 0;*right*: 0;}  
 #wrap>ul>li{*position*: **absolute**;*top*:0;*left*: 0;}  
 #wrap>ol>li{  
 *float*: **left**;  
 *width*: 20**px**;  
 *height*: 20**px**;  
 *text-align*: **center**;  
 *line-height*: 20**px**;  
 *border*: 1**px solid white**;  
 *background*: **gray**;  
 *margin-right*: 5**px**;  
 }  
 #wrap>ol>li:hover{*cursor*: **pointer**;}  
 #wrap li.active{  
 *padding*: 2**px**;  
 *color*: **orange**;  
 *margin-top*: -4**px**;  
 *border*: 1**px solid orange**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
<**div** id=**"wrap"**>  
 <**ul**>  
 <!--设置绝对定位之后 脱离标准流 最后一个盒子层级提升了-->  
 <**li** style=**"***z-index*: 1;**"**><**a** href=**"#"**><**img** src=**"./images/01.jpg"**/></**a**></**li**>  
 <**li**><**a** href=**"#"**><**img** src=**"./images/02.jpg"**/></**a**></**li**>  
 <**li**><**a** href=**"#"**><**img** src=**"./images/03.jpg"**/></**a**></**li**>  
 <**li**><**a** href=**"#"**><**img** src=**"./images/04.jpg"**/></**a**></**li**>  
 <**li**><**a** href=**"#"**><**img** src=**"./images/05.jpg"**/></**a**></**li**>  
 </**ul**>  
 <**ol**>  
 <**li** class=**"active"**>**1**</**li**>  
 <**li**>**2**</**li**>  
 <**li**>**3**</**li**>  
 <**li**>**4**</**li**>  
 <**li**>**5**</**li**>  
 </**ol**>  
</**div**>  
</**body**>  
<**script** src=**"jquery-3.3.1.min.js"**></**script**>  
<**script** type=**"text/javascript"**>  
 $(function () {  
 //控制层级关系的索引  
 var index = 0;  
 $('#wrap>ol>li').mouseenter(function () {  
 index++;  
 //修改下表的class  
 $(this).addClass('active').siblings('li').removeClass('active');  
 //修改图片  
 $('#wrap>ul>li').eq($(this).index()).css({left:650,'z-index':index}).animate({left:0},1000)  
 })  
 })  
</**script**>  
</**html**>

知识点：

1.z-index属性，只有当标签元素设置了 position的时候，这个才能生效。

2.因为所有的照片都在一个ul里面显示，并且有层叠样式。因此，在做动画的时候，就要对照片设置z-index的值。这样才能够不断的变换照片。

## 23-案例-动态实现轮播图

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**26-案例-动态实现轮播图**</**title**>  
 <**style** type=**"text/css"**>  
 \*{*padding*: 0;*margin*: 0;}  
 ul{*list-style*: **none**;}  
 #box{*width*: 240**px**;*height*: 180**px**;*position*: **relative**;*margin*: 50**px auto**;*overflow*: **hidden**;}  
 ul{*width*: 960**px**;*position*: **absolute**;}  
 ul li{*float*: **left**;}  
 p{*position*: **absolute**;*left*: 80**px**;*bottom*: 20**px**;}  
 p span{*color*: **white**;*display*: **inline-block**;*width*: 20**px**;*height*: 20**px**;*line-height*: 20**px**;*text-align*: **center**;*cursor*: **pointer**;}  
 p span.active{*color*: **red**;*background-color*: **white**;}  
 </**style**>  
</**head**>  
<**body**>  
 <**div** id=**"box"**>  
 <**ul**></**ul**>  
 <**p**></**p**>  
 </**div**>  
 <**button** id=**"play"**>**轮播吧！**</**button**>  
 <**button** id=**"stop"**>**暂停！**</**button**>  
</**body**>  
<**script** src=**"jquery-3.3.1.min.js"**></**script**>  
<**script** type=**"text/javascript"**>  
 $(function () {  
 //1.获取本地图片数据,后面可以从后端服务器获取  
 var imgArr = ['./images/06.jpg','./images/07.jpg','./images/08.jpg','./images/09.jpg'];  
 //2.动态的生成图片,并添加得到document的ul标签里面  
 for(var i=0;i<imgArr.length;i++){  
 $('ul').append("<li><img src="+imgArr[i]+" /></li>")  
 }  
 //3.生成索引（对应的图片序号）  
 var str = '';  
 $('li').each(function (*ind*,*ele*) {  
 str += "<span>"+(*ind*+1)+"</span>";  
 });  
 $('p').html(str);  
 //4.默认设置索引的第一个active  
 $('span').first().addClass('active');  
 //5.点击索引更改图片  
 var ind = 0;  
 $('span').click(function () {  
 $(this).addClass('active').siblings('span').removeClass('active');  
 //获取当前点击的索引  
 ind = $(this).index();  
 // $('ul').css({left:ind \* (-240)});  
 $('ul').animate({left:ind \* (-240)},100);  
 });  
 //6.自动播放  
 var timer = null;  
 $('#play').click(function () {  
 //开启定时器  
 clearInterval(timer);  
 timer = setInterval(next,1000);  
 function next() {  
 if(ind===$('li').length-1){  
 ind = 0;  
 $('p span').eq(ind).addClass('active').siblings('span').removeClass('active');  
 $('ul').css('left',0);  
 }else {  
 ind++;  
 $('p span').eq(ind).addClass('active').siblings('span').removeClass('active');  
 $('ul').css('left',(-240)\*ind);  
 }  
 }  
 });  
 //7.暂停轮播图  
 $('#stop').click(function () {  
 clearInterval(timer);  
 })  
 })  
</**script**>  
</**html**>

## 24.事件流

### 1.事件的概念

HTML中与javascript交互是通过事件驱动来实现的，例如鼠标点击事件、页面的滚动事件onscroll等等，可以向文档或者文档中的元素添加事件侦听器来预订事件。想要知道这些事件是在什么时候进行调用的，就需要了解一下“事件流”的概念

### 2.什么是事件流

事件流描述的是从页面中接收事件的顺序

1、DOM事件流

“DOM2级事件”规定的事件流包括三个阶段：

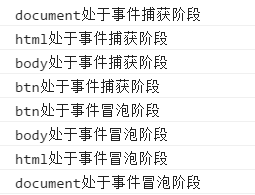
① 事件捕获阶段；

② 处于目标阶段；

③ 事件冒泡阶段

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**事件流**</**title**>  
 <**script**>  
 *window*.onload = function(){  
 var oBtn = *document*.getElementById('btn');  
 oBtn.addEventListener('click',function(){  
 *console*.log('btn处于事件捕获阶段');  
 }, true);  
 oBtn.addEventListener('click',function(){  
 *console*.log('btn处于事件冒泡阶段');  
 }, false);  
 *document*.addEventListener('click',function(){  
 *console*.log('document处于事件捕获阶段');  
 }, true);  
 *document*.addEventListener('click',function(){  
 *console*.log('document处于事件冒泡阶段');  
 }, false);  
 *document*.documentElement.addEventListener('click',function(){  
 *console*.log('html处于事件捕获阶段');  
 }, true);  
 *document*.documentElement.addEventListener('click',function(){  
 *console*.log('html处于事件冒泡阶段');  
 }, false);  
 *document*.body.addEventListener('click',function(){  
 *console*.log('body处于事件捕获阶段');  
 }, true);  
 *document*.body.addEventListener('click',function(){  
 *console*.log('body处于事件冒泡阶段');  
 }, false);  
 };  
 </**script**>  
</**head**>  
<**body**>  
 <**a** href=**"javascript:;"** id=**"btn"**>**按钮**</**a**>  
</**body**>  
</**html**>

运行结果：



知识点：

1. href="javascript:;"

这里的href="javascript:;"，其中javascript:是伪协议，它可以让我们通过一个链接来调用javascript函数.而采用这个方式 javascript:;可以实现A标签的点击事件运行时，如果页面内容很多，有滚动条时，页面不会乱跳，用户体验更好

1、addEventListener：addEventListener 是DOM2 级事件新增的指定事件处理程序的操作，这个方法接收3个参数：要处理的事件名、作为事件处理程序的函数和一个布尔值。最后这个布尔值参数如果是true，表示在捕获阶段调用事件处理程序；如果是false，表示在冒泡阶段调用事件处理程序。

2、document、documentElement和document.body三者之间的关系：

document代表的是整个html页面；

document.documentElement代表的是<html>标签；

document.body代表的是<body>标签；

## 25. jQuery的事件对象和事件冒泡

事件对象和事件冒泡，其实就是从24的事件流就可以进行一个总结；用自己的话去总结就是：在事件冒泡的时候，整个盒子的每一层都设置了点击事件，那么就会按照事件流去进行一个触发，因此，要阻止这个事件流的冒泡效果，就要设置 阻止事件。ev.preventDefault();ev.stopPropagation();

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**28-jQuery的事件对象和事件冒泡**</**title**>  
 <**style** type=**"text/css"**>  
 \*{*margin*: 0;*padding*: 0;}  
 #box{*left*: 20**px**;*right*: 20**px**;*width*: 200**px**;*height*: 200**px**;*background-color*: **greenyellow**;*position*: **relative**;}  
 p{*width*: 100**px**;*height*: 100**px**;*background-color*: **red**;*top*: 20**px**;}  
 </**style**>  
</**head**>  
<**body**>  
 <**div** id=**"box"**>  
 <**button** style=**"***display*:**inline-block**;**"**>**点击**</**button**>  
 <**p** class=**"p1"**>**我是p标签**</**p**>  
 <**a** href=**"#"**>**路飞**</**a**>  
 </**div**>  
  
</**body**>  
</**html**>

### 1.冒泡的效果

$(function () {  
 $('button').click(function () {  
 //冒泡 阻止冒泡  
 alert('btn事件触发了')  
 });  
 $('#box').click(function () {  
 alert('div事件触发了！')  
 });  
})

总结：

原始效果：当点击button按钮的时候，就只触发他自己的事件。

最后效果：结果当点击button按钮的时候，#box 的事件也被触发了。因为，button按钮在#box 的选择器内。所以，就会出现冒泡效果。

### 2.阻止事件冒泡

1.阻止事件冒泡

*ev*.stopPropagation();

2.阻止a标签的默认事件

return false;

注意：

1、 事件处理方法中 的 return false 并不是终止事件，而是阻止事件宿主的默认行为；

2、 不是在所有的事件处理方法中 return fasle 都能阻止事件宿主的默认行为；

事实上，仅仅是在HTML事件属性 和 DOM0级事件处理方法中 才能通过返回 return false 的形式组织事件宿主的默认行为。

3.阻止a标签默认事件

*ev*.preventDefault();  
*ev*.stopPropagation();

上面这两句就相当于 return false

preventDefault() 方法阻止元素发生默认的行为（例如，当点击提交按钮时阻止对表单的提交）

ev.stopPropagation：不再派发事件。终止事件在传播过程的捕获、目标处理或起泡阶段进一步传播。调用该方法后，该节点上处理该事件的处理程序将被调用，事件不再被分派到其他节点。

$(function () {  
 //事件对象  
 $('.p1').click(function (*ev*) {  
 //事件类型 事件属性  
 *console*.log(*ev*.type); //click  
 *console*.log(*ev*.target); // <p class="p1">我是p标签</p>  
 *console*.log(*ev*.pageX); // 57  
 *console*.log(*ev*.pageY); // 36  
 //常用的事件 方法  
 alert('p标签事件触发了！');  
 //1.阻止事件冒泡  
 *ev*.stopPropagation();  
 });  
 $('a').click(function (*ev*) {  
 //2.阻止默认事件: 阻止a标签的默认事件(不跳转到a标签里面的href连接)  
 // ev.preventDefault();  
 // ev.stopPropagation();  
 alert('a标签点击了');  
 return false;  
 });  
 $('#box').click(function () {  
 alert('box 点击了！')  
 })  
})

## 26.自定义事件和事件代理

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**30-jQuery自定义事件和事件代理**</**title**>  
 <**style** type=**"text/css"**>  
 \*{*margin*: 0;*padding*: 0;}  
 #box{*width*: 200**px**;*height*: 200**px**;*background*: #25CB33;*margin*: 40**px auto**;}  
 </**style**>  
</**head**>  
<**body**>  
<**ul**>  
 <**li**>**luffy1**</**li**>  
 <**li**>**luffy2**</**li**>  
 <**li**>**luffy3**</**li**>  
</**ul**>  
</**body**>  
</**html**>

### 1.绑定事件

$(*document*).ready(function () {  
 //先点击  
 $('ul>li').bind('click',fn1=function () {  
 *console*.log($(this));  
 *console*.log($(this).html())  
 });  
})

### 2.事件代理

事件代理就是自己完成不了当前的点击事件，交给父级元素来做这个事件操作

语法：父级.on('事件名字','选择器1，选择器2'，fn(回调函数))

$(*document*).ready(function () {  
 $('ul').on('click','li',function () {  
 *console*.log(this);  
 });  
 //后添加就不能使用前面的事件,因此要使用事件代理  
 $('ul').append('<li>ryan</li>')  
})

知识点：

只有在事件代理的情况下，添加元素到dom，才能使用执行前面的事件。如果在添加元素的前面使用的是事件绑定，那就不能使用事件绑定里面的事件。

## 27.jQuery鼠标事件

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**31-jQuery鼠标事件**</**title**>  
 <**style** type=**"text/css"**>  
 \*{*padding*: 0;*margin*: 0;}  
 #box{*width*: 200**px**;*height*: 200**px**;*background-color*: **crimson**;}  
 #child{*width*: 100**px**;*height*: 100**px**;*background-color*: **yellow**;}  
 </**style**>  
</**head**>  
<**body**>  
 <**div** id=**"box"**>  
 <**div** id=**"child"**></**div**>  
 <**input** type=**"text"** value=**"123"**>  
 <**br**>  
 <**input** type=**"password"**>  
 </**div**>  
</**body**>  
</**html**>

### 1.单机&双击事件

$(*document*).ready(function () {  
 //点击事件  
 //单击  
 $('#box').click(function () {  
 *console*.log('click 事件')  
 $('#box').css('background','yellowgreen')  
 });  
 // //双击  
 $('#box').dblclick(function () {  
 $('#box').css('background','aqua')  
 })  
})

### 2.按下&弹起事件

$(*document*).ready(function () {  
 //鼠标按下 弹起  
 $('#box').mousedown(function () {  
 $('#box').css('background','yellowgreen')  
 });  
 $('#box').mouseup(function () {  
 $('#box').css('background','aqua')  
 });  
})

### 3.mouseover & mouseout

mouseover 以及 mouseout

被选元素和子元素被选中时 会触发函数 移入移除；

意思就是，当鼠标从动到父盒子的时候先触发 mouseover 然后鼠标从父盒子移动到子盒子的时候会触发一次mouseout，然后再触发一次 mouseover

$(*document*).ready(function () {  
 $('#box').mouseover(function () {  
 $(this).css({background:'red'});  
 *console*.log('mouseover');  
 });  
 $('#box').mouseout(function () {  
 $(this).css({background:'#25CB33'});  
 *console*.log('mouseout');  
 })  
})

### 4.mouseenter & mouseleave

mouseenter 以及 mouseleave

只有被选元素移入的时候才会触发

和上面的mouseover/out 对比在于：mouseenter把整个盒子当成了一个元素，无论父盒子还是子盒子 移入移出就只有一次操作。

$(*document*).ready(function () {  
 $('#box').mouseenter(function () {  
 $(this).css({background:'red'});  
 *console*.log('mouseenter');  
 });  
 $('#box').mouseleave(function () {  
 $(this).css({background:'#25CB33'});  
 *console*.log('mouseleave');  
 })  
})

### 5.mousemove

当鼠标指针在指定的元素中移动时，就会发生 mousemove 事件。

mousemove() 方法触发 mousemove 事件，或规定当发生 mousemove 事件时运行的函数。

**注意：**用户把鼠标移动一个像素，就会发生一次 mousemove 事件。处理所有 mousemove 事件会耗费系统资源。请谨慎使用该事件。

$(*document*).ready(function () {  
 $('#box').mousemove(function(*event*){  
 $("#box").text(*event*.pageX + ", " + *event*.pageY);  
 });  
})

### 6.获取焦点(focus) & 失去焦点(blur)事件

$(*document*).ready(function () {  
 //获取焦点  
 $('input[type=text]').focus(function () {  
 *console*.log($(this).val())  
 });  
 //失去焦点  
 $('input[type=text]').blur(function () {  
 *console*.log($(this).val())  
 })  
})

### 7.键盘按下(keydown) & 弹起(keyup)事件

//keydown() /keyup()  
//keydown : 当键盘按下的时候的第一瞬间只是原始的值，当键盘弹起或者有值继续输入的时候就是 keyup获取的值  
$('input[type=password]').keydown(function () {  
 *console*.log($(this).val())  
});  
$('input[type=password]').keyup(function () {  
 *console*.log($(this).val())  
})

## 28.jQuery表单事件

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**32-jQuery表单事件**</**title**>  
 <**meta** name=**"viewport"** content=**"width=device-width, initial-scale=1"**>  
 <**style**>  
 .show{*color*: **red**;}  
 </**style**>  
</**head**>  
<**body**>  
 <**form** action=**"https://www.luffycity.com"**>  
 <**select** name=**"sweets"** id=**""** multiple=**''**>  
 <**option** value=**""**>**巧克力**</**option**>  
 <**option** value=**""** selected=**''**>**糖果**</**option**>  
 <**option** value=**""**>**焦糖**</**option**>  
 <**option** value=**""** selected=**''**>**曲奇饼**</**option**>  
 <**option** value=**""**>**烧饼**</**option**>  
 <**option** value=**""**>**麦香饼**</**option**>  
 <**option** value=**""**>**曲奇饼2**</**option**>  
 </**select**>  
 <**input** type=**"text"** value=**"hello"** id=**'target'**>  
 <**input** type=**"submit"** value=**"Luffy"**/>  
 <**input** type=**"button"** value=**"按钮"** />  
 </**form**>  
 <**textarea** id=**"other1"**>**Trigger the handler**</**textarea**>  
 <**input** id=**"other"** type=**"text"** value=**"Trigger the handler"**>  
 <**div** class=**"show"**></**div**>  
</**body**>  
</**html**>

### 1.change()事件

当元素的值发生改变时，会发生 change 事件。

该事件仅适用于文本域（text field），以及 textarea 和 select 元素。

change() 函数触发 change 事件，或规定当发生 change 事件时运行的函数。

**注释：**当用于 select 元素时，change 事件会在选择某个选项时发生。当用于 text field 或 text area 时，该事件会在元素失去焦点时发生。

$('select').change(function () {  
 $('.show').html($('select option:selected').html());});

### 2.select()方法

当 textarea 或文本类型的 input 元素中的文本被选择时，会发生 select 事件。

select() 方法触发 select 事件，或规定当发生 select 事件时运行的函数。

**注释**：此方法只适用于 input 的type=text类型或者 textarea标签

$('#other').select(function () {  
 *console*.log($(this).val())  
});

### 3.submit事件

当提交表单时，会发生 submit 事件。

该事件只适用于表单元素。

submit() 方法触发 submit 事件，或规定当发生 submit 事件时运行的函数。

例子：给上面的form表单设置阻止事件: 也就是阻止 form表单中设置的默认action的链接不执行。然后执行后面阻止事件后面的代码。

$('form').submit(function (*ev*) {  
 *ev*.preventDefault();  
 alert('111');  
})

## 29.jQuery的Ajax

### 1.什么是AJAX

AJAX = 异步的javascript和XML（Asynchronous Javascript and XML）

简言之，在不重载整个网页的情况下，AJAX通过后台加载数据，并在网页上进行显示。

通过 jQuery AJAX 方法，您能够使用 HTTP Get 和 HTTP Post 从远程服务器上请求文本、HTML、XML 或 JSON - 同时您能够把这些外部数据直接载入网页的被选元素中。

Ajax 在后面的django会有详细讲解。

### 2. **jQuery的load()方法**

jQuery load()方法是简单但强大的AJAX方法。

load()方法从服务器加载数据，并把返回的数据放入被选元素中。

**语法：**

$("selector").load(url,data,callback);

必须的url参数规定记载的url地址

可选的data参数规定与请求一同发送的查询字符串键/值对的集合

可选的callback参数是load()方法完成后所执行的函数名称

1、

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

//只传一个url，表示在id为#new-projects的元素里加载index.html

$('#new-projects').load('./index.html');

})

2、

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

//只传一个url，导入的index.html文件含有多个传递参数，类似于：index/html?name='张三'

$('#new-projects').load('./index.html',{"name":'张三',"age":12});

})

3、

//加载文件之后，会有个回调函数，表示加载成功的函数

$('#new-projects').load('./index.html',{"name":'张三',"age":12},function(){

});

注意：load函数最好在服务器网页中应用，也就是说要在服务器上运行，本地调试需要搭建后端本地环境。

### **2. jquery的getJSON方法**

jQuery的AJAX中使用getJSON()方法异步加载JSON格式数据。获取服务器中的数据，并对数据进行解析，显示到页面中

语法: $.getJSON(url,[data],[callback])

url参数为请求加载json格式文件的服务器地址，可选项data参数为请求时发送的数据，callback参数为数据请求成功后执行的函数

更多应用在django可以在详细笔记。

# 四 Bootstrap

## 1.常用网站

查询js，css，html的文档：开发者工具 https://developer.mozilla.org/zh-CN/

ECMAScript 6 入门 ：http://es6.ruanyifeng.com

Bootstrap官方文档：http://www.bootcss.com/

jqueryUI 官网：<https://jqueryui.com/>

jqueryUI 中文网：<http://www.jqueryui.org.cn/>

官网demo： <https://www.oschina.net/project/tag/273/jquery>

图表网站1：<https://chartjs.bootcss.com/>

css网站: http://caibaojian.com/30-seconds-of-css/

## 2.bootstrap介绍

Bootstrap，来自 Twitter，是目前最受欢迎的前端框架。Bootstrap 是基于 HTML、CSS、javascript 的，它简洁灵活，使得 Web 开发更加快捷。

**它用于开发响应式布局、移动设备优先的 WEB 项目**

## 3. 响应式页面介绍

描述响应式界面最著名的一句话就是“Content is like water”，翻译成中文便是“如果将屏幕看作容器，那么内容就像水一样”。

### 1.响应式界面的基本规则

1、可伸缩的内容区块：内容区块的在一定程度上能够自动调整，以确保填满整个页面

2、可自由排布的内容区块：当页面尺寸变动较大时，能够减少/增加排布的列数

3、适应页面尺寸的边距：到页面尺寸发生更大变化时，区块的边距也应该变化

4、能够适应比例变化的图片：对于常见的宽度调整，图片在隐去两侧部分时，依旧保持美观可用

5、能够自动隐藏/部分显示的内容：如在电脑上显示的的大段描述文本，在手机上就只能少量显示或全部隐藏

6、能自动折叠的导航和菜单：展开还是收起，应该根据页面尺寸来判断

7、放弃使用像素作为尺寸单位：用dp尺寸等方法来确保页面在分辨率相差很大的设备上，看起来也能保持一致。同时也要求提供的图片应该比预想的更大，才能适应高分辨率的屏幕

## 4. CSS3 的 @media 查询

### 1.定义和使用

使用 @media 查询，你可以针对不同的屏幕大小定义不同的样式。 @media 可以针对不同的屏幕尺寸设置不同的样式，特别是如果你需要设置设计响应式的页面，@media 是非常有用的。 当你重置浏览器大小的过程中，页面也会根据浏览器的宽度和高度重新渲染页面，这对调试来说是一个极大的便利。

CSS 语法：

@media mediaType and|not|only (media feature) {

/\*CSS-Code;\*/

}

媒体功能（media feature） 媒体功能也有很多

列出常用的几个值描述

max-width:定义输出设备中的页面最大可见区域宽度

min-width:定义输出设备中的页面最小可见区域宽度

### 2.兼容设备代码

首先我们在使用 @media 的时候需要先设置下面这段代码，来兼容移动设备的展示效果：

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

参数解释：

width = device-width：宽度等于当前设备的宽度

initial-scale：初始的缩放比例（默认设置为1.0，即代表不缩放）

user-scalable：用户是否可以手动缩放（默认设置为no，因为我们不希望用户放大缩小页面） 其他还有很多参数，想要了解直接去百度

### 3. 加载兼容文件JS

因为IE8既不支持HTML5也不支持CSS3 @media ，所以我们需要加载两个JS文件，来保证我们的代码实现兼容效果：

<**script** src=**"https://cdn.bootcss.com/html5shiv/3.7.3/html5shiv.min.js"**></**script**>  
<**script** src=**"https://cdn.bootcss.com/respond.js/1.4.2/respond.min.js"**></**script**>

### 4.设置IE渲染方式

设置IE渲染方式默认为最高(可选) 现在有很多人的IE浏览器都升级到IE9以上了，所以这个时候就有又很多诡异的事情发生了，例如现在是IE9的浏览器，但是浏览器的文档模式却是IE8 为了防止这种情况，我们需要下面这段代码来让IE的文档渲染模式永远都是最新的

<meta http-equiv="X-UA-Compatible" content="IE=Edge，chrome=1">

这段代码后面加了一个chrome=1，如果用户的电脑里安装了 chrome，就可以让电脑里面的IE不管是哪个版本的都可以使用Webkit引擎及V8引擎进行排版及运算，如果没有安装，就显示IE最新的渲染模式。

### 5.简单的响应式页面

<!DOCTYPE html>  
<**html** lang=**"en"**>  
<**head**>  
 <**meta** charset=**"UTF-8"**>  
 <**title**>**2-Bootstrap响应式原理**</**title**>  
 <!--使用@media 必须做一下操作-->  
 <**meta** name=**"viewport"** content=**"width=device-width, initial-scale=1.0, user-scalable=no"**>  
 <**style** type=**"text/css"**>  
 \*{*margin*: 0;*padding*: 0;}  
 /\*最大宽度为 500px 的时候\*/  
 @media **screen** and (*max-width*: 500**px**){  
 .box{*width*: 200**px**;*height*: 300**px**;*background-color*: **yellow**;}  
 }  
 /\*300px ~ 600px\*/  
 @media **screen** and (*min-width*: 500**px**) and (*max-width*: 900**px**){  
 .box{*width*: 200**px**;*height*: 300**px**;*background-color*: **red**;}  
 }  
 /\*最小宽度为900px的时候\*/  
 @media **screen** and (*min-width*: 900**px**){  
 .box{*width*: 200**px**;*height*: 300**px**;*background-color*: **black**;}  
 }  
 </**style**>  
</**head**>  
<**body**>  
 <**div** class=**"box"**>  
 </**div**>  
</**body**>  
<**script** src=**"jquery-3.3.1.min.js"**></**script**>  
<**script** type=**"text/javascript"**>  
</**script**>  
</**html**>

1、通过灵活应用以上技巧，开发出一个响应式页面，还不是近在咫尺的感觉。

2、不要被 min-width 和 max-width 所迷惑，初学者很容易误以为 min-width 的意思是小于xxx的时候才应用，然而这就陷入误区了，其实它的意思是：当设置了 min-width 的时候，文档的宽度如果小于设置的值，就不会应用这个区块里的CSS样式，所以 min-width 它才能实现大于等于设置的值得时候，才会应用区块里的CSS样式，max-width 也是如此。

3、或者这样想想，先看代码，这句代码的意思是宽度大于等于 300px ，小于等于 500px （ width >=300 && width <=500）的时候应用样式

@media screen and (min-width:300px) and (max-width:500px) {

/\* CSS 代码 \*/

}

min-width:300px 的作用是当文档宽度不小于 300px 的时候就应用 {} 里的CSS代码块，即大于等于 300px，max-width:500px 的作用是当文档宽度不大于 500px 的时候就应用{} 里的CSS代码块，即小于等于 500px

## 5.Bootstrap的引入方式

<!DOCTYPE html>  
<**html** lang=**"zh-CN"**>  
<**head**>  
 <**meta** charset=**"utf-8"**>  
 <**meta** http-equiv=**"X-UA-Compatible"** content=**"IE=edge"**>  
 <**meta** name=**"viewport"** content=**"width=device-width, initial-scale=1"**>  
 <**title**>**7-Bootstrap插件介绍**</**title**>  
 <**link** rel=**"stylesheet"** type=**"text/css"** href=**"./bootstrap-3.3.7-dist/css/bootstrap.min.css"**>  
<**body**>  
</**body**>  
<**script** src=**"./jquery-3.3.1.min.js"**></**script**>  
<**script** src=**"./bootstrap-3.3.7-dist/js/bootstrap.min.js"**></**script**>  
</**html**>

## 6.Bootstrap的具体使用

使用直接查看官方文档即可。

### 1.方法使用重点

1.布局容器

2.栅格系统

3.下拉菜单

### 2.插件使用重点

1.下拉菜单

2.模态框

3.滚动监听

4.元素的隐藏 使用 class="sr-only"