# 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.CSS伪类选择器二

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\*/  
 p:after{  
 *content*: "love ";  
 *color*: **black**;  
 *font-size*: 35**px**;  
 }  
 </**style**>  
</**head**>  
<**body**>  
<**p**>**爱你哟**</**p**>  
</**body**>  
</**html**>

### 18.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**>

### 19.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**>

### 20.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**>

### 21.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

### 22.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

### 23.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

### 24.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。

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;}

### 25.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

### 26.浮动产生的问题

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，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

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

### 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');  
 })});

‘

’