**一、html介绍和常用的技巧**

**一、html**

HyperText Markup Language : 超文本标记语言

**二、点html什么意思？**

.mp4 .mp3 .exe .jpg .png .html ===》浏览器

**三、这是什么玩意**

<!DOCTYPE html> ===》声明帮助浏览器正确地显示网页

<html lang="en"> ===》根

<html lang="en">  
<head>  
 <meta charset="UTF-8">（出现乱码需要改为HTML5文件的格式）

</head>

<meta charset="UTF-8">

<body> ===》主体

</body>

</html>

**四、标记语言**

<英文单词或字母组合>

<body> 开始

</body> 结束

**五、标记属性**

<英文单词或字母组合 style class id ...>

\*\*\*\*\*

写代码在编辑工具中（sublime）

代码显示：浏览器

**一、html标签**

<p></p> ： 段落标签

h1-h6 ： 标题标签

strong : 加粗

em ： 斜体

del ： 删除

div 、span (非常重要) ===》结构化

**一、html标签**

ol--->li ： 有序列表

ol的属性：type="a";"A";"i"#排序的类型，类似于word里的list类目

<ol style="color:red">#添加颜色为红色

<li>西瓜</li>

<li>香蕉</li>

<li>苹果</li>

</ol>

ul---->li ： 无序列表

<ul style="color:yellow">

<li>西瓜</li>

<li>香蕉</li>

<li>苹果</li>

**一、img ： 图片**

标签属性：src（添加一个网页路径或者本地路径）、

eg:src="a.jpeg";<img src="https://timgsa.baidu.com/timg?image&amp" alt="hahha" title="09090">

alt（给图片添加一个本地文件的源数据）、

eg:alt='这是老张前xx'

title(给图片添加标题，鼠标停止时会显示)

eg:title='ooooooooooo'

**二、a ： 超链接(可以跳转页面)**

<a href="https://hao.qq.com/?unc=Af31026&s=o400493\_1"></a>

<a href="https://www.taobao.com/" target="\_blank">#点击图片跳转

<img src="https://timgsa.baidu.com/timg?image&amp" alt="hahha" title="09090">

</a>

标签属性：href（跳转到哪了？）、target='\_blank'（跳转到新页面）

**一、form表单**

<form action="https://www.baidu.com/">(登录后提交转到对应的网址)

文本：<input type=""> === <input type="text">

密码：<input type="password">

按钮：<input type='button' value='登录'>

提交：<input type='submit' value='登录'>

单选按钮：<input type="radio" name="select">##(name=“select”，表示浏览器能够识别出是单选)

多选按钮：<input type='checkbox'> 或者<input type='checkbox'name="">

</form>

**表格：table**

<meta charset="UTF-8">

<table>

<thead>

<tr>

<td>姓名</td>

</tr>

<tr>

<td>姓名</td>

</tr>

</thead>

<tbody>

<tr>

<td>掌上</td>

</tr>

<tr>

<td>掌上</td>

</tr>

</tbody>

</table>

**下拉框选择 ： select**

<select>

<option>北京</option>

<option>湖南</option>

<option>湖北</option>

<option>河北</option>

</select>

**&nbsp;<!--空格-->**

单标记

<br /> ===》换行

<hr /> ===》分割线

<div>

老张<br />前端

老张<hr />前端

</div>

注释：<!--这是我的解释--> ： 它不会执行

<!--

<div>1</div>

<div>2</div>

<div>3</div>

<div>4</div>

<div>5</div>

-->

**二、CSS常用技巧和功能**

**css引入样式：**

**一、行内样式**

<div style="background: #9fff5c;"></div>

<div style='background: red;'>这是一个div</div>

<p style='background: blue;'>这是一个p标记<p></p>

<em style="background: pink;">这是一个em</em>

**二、内部引入<!--在</head>的前面添加即可-->**

<style type="text/css">

div{

background: red;

}

p{

background: blue;

}

span{

background: pink;

}

</style>

**三、外部引入<!--在</head>的前面添加即可-->**

<link rel="stylesheet" type="text/css" href="style.css"><!--同级别目录-->

<link rel="stylesheet" type="text/css" href="css/style.css"><!--不同级别目录-->

</head>

<!--style.css文件内的内容为：-->

div{

background: red;

}

**一、id选择器(唯一)**

html<!--html内容-->:

<div id='dv'>333</div>

css<!--.css文件内的内容-->:

#dv{

background: blue;

}

**二、class选择器（多个）**

html:<!--将2、4行加上颜色-->

<ul>

<li>111</li>

<li class='lis'>222</li>

<li>333</li>

<li class='lis'>444</li>

</ul>

css:

.lis{

background: red;

}

**三、标签选择器**

html:

<p>111</p>

<p>222</p>

<p>333</p>

css:<!--页面上的所有标签-->

p{

background: red;

}

**四、通配符选择器<!--针对页面的所有内容-->**

\* ： 页面所有标签

\*{

background: #00ff00;

}

一、div,p<!-- 选取到页面所有的div和p标记-->

div,p{

background: blue;

}

<div>111</div>

<div>222</div>

<div>333</div>

<div>444</div>

<p>111</p>

<p>222</p>

<p>333</p>

二、子元素选择器

div p ： <!--选取到div内 "所有" 的p标记-->

div p{

background: pink;

}

<p>0000</p>

<div>

<p>111</p>

<p>222</p>

<p>333</p>

</div>

<p>9999</p>

div>p ：<!--选取到div内 "子集" 的p标记-->

div>span{

background: red;

}

<div>

<span>1111</span>

<span>2222</span>

<span>3333</span>

<span>4444</span>

<p>

<span>0000000</span>

</p>

</div>

三、兄弟元素选择器

div+span ：<!--选取到div标签 "同级" 往后的span标记-->

div+span{

background: red;

}

<span>000</span>

<span>000</span>

<div>

<span>xxxx</span>

<span>xxxx</span>

<span>xxxx</span>

</div>

<span>1aaa</span>

<span>2aaa</span>

一、属性选择器

[value] :<!--选取到标签的属性有value的-->

<!--(1和5的颜色会发生变化)-->

[value]{

background: red;

}

<input type="text" value="请输入手机号">

<input type="password" name="">

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

<input type="checkbox" name="">

<input type="submit" value="登录">

[value=a] :<!--选取到标签的属性值为a的元素-->

<!--结果：5的颜色会变化-->

[value=登录]{

background: red;

}

<input type="text" value="请输入手机号">

<input type="password" name="">

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

<input type="checkbox" name="">

<input type="submit" value="登录">

:hover <!--选择鼠标指针指向的链接-->

<!--当鼠标知道颜色或字体的区域颜色会发生变化-->

.box{

width:300px;

height: 300px;

background: red;

}

.box:hover{

background: blue;

}

p:hover{

color:pink;

}

<div class='box'></div>

<p>文字</p>

#### 一、css选择器权重：

!important > 行内样式 > id > class > 标签 > 通配

!important ==》 无穷大

行内 ==》1000

id ==》100

class|属性|伪类 ==》10

标签|伪元素 ==》1

通配 ==》0

<!--遇到权重相加值相等的情况，后者覆盖前者的颜色，最终显示后者颜色-->

#container2 div .op1{

background: blue;

}

#container2 .box1 p{

background: pink;

}

.container1 #box2 p{

background: red;

}

<div class='container1' id='container2'>

<div class='box1' id='box2'>

<p class='op1' class='op2'>

选取谁了?

</p>

</div>

</div>

一、css选择器权重：

行内样式：

<div style='background: red;'>这是一个div</div>

<p style='background: blue;'>这是一个p标记<p></p>

<em style="background: pink;">这是一个em</em>

!important > 行内样式 > id > class > 标签 > 通配<!--\* ： 页面所有标签-->

!important ==》 无穷大

行内 ==》1000

id ==》100

class|属性|伪类 ==》10

标签|伪元素 ==》1

通配 ==》0

<!--遇到权重相加值相等的情况，后者覆盖前者的颜色，最终显示后者颜色-->

#container2 div .op1{

background: blue;

}

#container2 .box1 p{

background: pink;

}

.container1 #box2 p{

background: red;

}

<div class='container1' id='container2'>

<div class='box1' id='box2'>

<p class='op1' class='op2'>

选取谁了?

</p>

</div>

</div>

##### 一、颜色

#box{

width:100px;

height: 100px;

background:red;

color:#0f0;

color:#0f0;

}

背景颜色：background-color、background:

background:red;

文字颜色：color

color:#0f0;

颜色值===》可以写成rgba(255,255,255)

/\*color:rgba(255,255,255);\*/

二、边框(border)

border:100px solid pink : 给元素所有边框设置1像素实线粉色

<div id="box"></div>

solid ==》实线

dashed==》虚线

border-left:1px solid red;

border-right:10px dashed blue;

border-top:20px solid pink;

border-bottom:10px dashed green;

border-bottom-color:blue; : 给下边框颜色值设置成xx颜色

-----------

width : 宽度

height : 高度

<!--更改边框的颜色可以得到不同的显示效果-->

#box{

width:0px;

height:0px;

border:100px solid red;

border-top-color:#fff;

border-left-color:#fff;

border-right-color:#fff;

border-bottom-color:pink;

}

<div id="box"></div>

#### 字体

一、字体大小（font-size）

font-size:50px

一般网站用的最多文字尺寸：12px、14px

\*\*\*\*

谷歌默认大小：16px;

最小值：12px

\*\*\*\*

二、加粗(font-weight)

默认值：normal

加粗：bold、数值（100-900）

三、斜体（font-style:italic）

<style type="text/css">

#box{

font-weight: 600 ;/\*|bold;\*/

font-style: italic;

}

</style>

四、文字位置（text-align）

居中：text-align:center

居左：text-align:left

居右：text-align:right

五、行高（line-height）

<style type="text/css">

#box{

width:500px;

height: 250px;

border:10px solid #ccc;

text-align: center;

line-height: 250px;

}

</style>

<div id="box">发动机发</div>

六、text-deoration

<!--属性规定添加到文本的修饰:上划线，下划线，中间穿线-->

h1 {text-decoration:overline}

h2 {text-decoration:line-through}

h3 {text-decoration:underline}

h4 {text-decoration:blink}

七、cursor<!--定义鼠标指针悬浮在元素上方显示的鼠标光标-->

<style type="text/css">

a,span{

text-decoration: none;

color:#999;

}

#box:hover{

background: orange;

}

a:hover{

color:#fff;

text-decoration:underline;

}

</style>

<div id='box'>

<a href="">游戏</a>

<span>/</span>

<a href="">动漫</a>

<span>/</span>

<a href="">影视</a>

</div>

#### 一、块级元素 :

display:block

独占一行

可以通过css改变宽高

例如：div、p、ul、ol、li、form...

二、行级元素 :

display: inline

内容决定大小

不可以通过css改变宽高

例如：span、em、strong...

三、行级块元素 :

display:inline-block

内容决定大小

可以通过css改变宽高

例如：img...

#### 设计技巧：

通常在开发之前都需要定义通配文件CSS，能够避免出现重复和麻烦，提升开发效率

Eg:hmtl:内容

<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="UTF-8">  
 <title></title>  
 <link rel="stylesheet" type="text/css" href="style.css">  
</head>  
<body>  
<p>这是p</p>  
<a href="">这是a</a>  
<ul>  
 <li class='textColor bgColor'>1</li>  
</ul>  
<ol>  
 <li>1</li>  
</ol>  
</body>  
</html>

CSS内容：

\*{  
 margin:0;  
 padding:0;  
}  
a{  
 text-decoration: none;  
}  
ul,ol{  
 list-style: none;  
}  
  
/\*文字颜色\*/  
.textColor{  
 color:orange;  
 po  
}  
/\*背景颜色\*/  
.bgColor{  
 background: #eee;  
}

一、盒模型：

盒子内容：width+height

盒子边框：border

<!--能够设置边框的颜色和线的粗细以及虚实线-->

盒子外边距：margin ==》 上、右、下、左

2个值： 20px（上下） 50px（左右）

3个值: 1px(上) 2px（左右） 3px（下）;

.box{  
width:100px;  
height: 100px;  
background: red;  
/\*margin-top:20px;  
margin-left:20px;  
margin-right:20px;  
margin-bottom:20px;\*/  
/\*margin:20px 50px;\*/  
/\*margin:1px 20px;\*/  
margin:10px 1px 3px;  
}

盒子内边距：padding ==》 上、右、下、左

2个值： 20px（上下） 50px（左右）

3个值: 1px(上) 2px（左右） 3px（下）;

.main{  
width:100px;  
height: 100px;  
background: red;  
padding:10px 20px 30px 40px;  
padding:20px 50px  
padding:1px 2px 3px  
}

#### 定位：position

默认值：static

inherit

fixed ==》相对于浏览器窗口定位

<!--固定头目录在顶端，滚动页面时不随着页面变化-->  
.header{  
width:100%;  
height: 80px;  
background: #fff;  
border-bottom:1px solid #ccc;  
position: fixed;  
top:0;  
left:0;  
}  
<!--固定一个盒子在浏览器的右下方，滚动时不随着页面变化-->  
.f{  
width:100px;  
height: 100px;  
background: #000;  
position: fixed;  
right:20px;  
bottom:20px;  
}

absolute ==》绝对定位

<!--指定一个固定区域在浏览器的的位置和大小-->  
.po{  
width:100px;  
height: 100px;  
background: red;  
position: absolute;  
left:100px;  
top:50px;  
}

relateive ==》相对定位

<!--指定一个相对固定区域在浏览器的的位置和大小-->  
<!--left : 从左到右 right : 从右到左 bottom: 从下到上 top : 从上到下-->  
.box{  
width:800px;  
height: 800px;  
background: blue;  
margin-top:10px;  
margin-left:10px;  
position: relative;  
}  
<div class='box'>  
 <div class='po'></div>  
</div>  
--------  
left : 从左到右  
right : 从右到左  
bottom: 从下到上  
top : 从上到下

#### 浮动：

float: left / right

<!--图片左边固定大小float-->

img{

width:100px;

height: 100px;

float:left;

}

<!--在插图作浮动之后在右边float-->

<img src='../timg.jpeg' style='float:right;'/>

-------------

float、position ===》自动转换：inline-block

显示出三个文本框在一行中的位置

\*{margin:0;padding:0;}

.container{

width:100%;

height: 100px;

border:1px solid #ccc;

position: relative;

}

.container>div{

float:left;

}

.container>.fav{

float: right;

}

.search{

<!--width:100px;-->

<!--height: 100px;-->

<!--background: red;-->

position: absolute;

left:50%;

top:0;

margin-left:-50px;

}

<div class='container'>

<div class='logo'>图片</div>

<div class='search'>搜索框</div>

<div class='fav'>收藏</div>

</div>

### 清除浮动：

1、浮动元素的最下面加入：

<!--clear:both（块元素）display:block-->  
\*{margin:0;padding:0;}  
.container{  
border:10px solid #ccc;  
}  
.item{  
width:100px;  
height: 100px;  
background: red;  
float:left;  
}  
span{  
clear:both;  
display: block;  
}  
 <div class='container'>  
 <div class='item'></div>  
 <div class='item'></div>  
 <div class='item'></div>  
 <span></span>

2、伪元素

<!--定义一个清除float的类再引用-->  
\*{margin:0;padding:0;}  
.container{  
border:10px solid #ccc;  
}  
.item{  
width:100px;  
height: 100px;  
background: red;  
float:left;  
}  
.clearfix:after{  
content:"";  
clear: both;  
display: block;  
}  
<div class='container clearfix'>  
 <div class='item'></div>  
 <div class='item'></div>  
 <div class='item'></div>  
</div>

一、文字溢出显示...

white-space:nowrap; 不换行

overflow:hidden; 超出隐藏

text-overflow: ellipsis; 显示...

\*{margin:0;padding:0;}  
.main{  
width:100px;  
border:1px solid #ccc;  
/\*文字显示不换行\*/  
white-space:nowrap;  
/\*文字过多自动超宽隐藏\*/  
overflow:hidden;  
/\*文本框显示为...\*/  
text-overflow: ellipsis;  
}  
<div class='main'>的方的空间发挥得很发达</div>

二、背景图片

background:url(地址) no-repeat(不重复)

\*{margin:0;padding:0;}  
/\*背景图片显示方式一\*/  
.item{  
width:1000px;  
height: 1000px;  
background: url('../timg.jpeg');  
/\*background-size:50% 50%;\*/  
background-repeat:no-repeat;  
}  
/\*背景图片显示方式二\*/  
.item1{  
width:1000px;  
height: 1000px;  
background: url('../timg.jpeg') no-repeat;  
}

项目设计技巧总结

为了定义一个类的内容居中显示：css中的宽度必须和margin：auto联合使用

Html:

<div class="header-main">  
 我的收藏  
</div>

CSS;

.header-main{  
 width: 1200px;  
 background: red; /\*显示颜色为使得看见效果\*/  
 margin: auto ;  
}

利用伪类去除掉 首页 的左边margin和 分享赚钱 的右边margin以及右边的边界线

.nav-item:last-child{  
 border-right: none;  
}  
.nav-item:first-child{  
 margin-left: 0;  
}  
.nav-item:last-child{  
 margin-right: 0;  
}

为了使得头目录固定位置不变需要添加相对位置，利用绝对位置函数使得nav的子目录目能够居中显示：

.header-main{  
 /\*width: 1200px;\*/  
/\*!\*background: red; !\*显示颜色为使得看见效果\*!\*!\*/  
/\*margin: auto ;  
 position: relative;  
}

.nav{  
 position: absolute;  
 right: 0;  
 top: 50%;  
 height: 30px;  
 margin-top: -15px;  
}

为了使得头部与banner部分的图片完全显示需要width和height改为100%、以及能够上下位置显示需要计算头部的高度来改变margin-top的值

<div>  
 <a class="banner">  
 <img src="img/banner.jpg" class="banner-img">  
 </a>  
</div>

.banner-img{  
 width: 100%;  
 height: 100%;  
 margin-top: 107px;  
}

固定二维码在浏览的位置

/\*扫描定位二维码\*/  
.pdd-fixed{  
 position: fixed; /\*固定二维码页面再历览器的位置\*/  
 bottom: 150px;  
 left: 150px;  
 width: 162px; /\*这三行固定二维码在左下角的位置和宽度\*/  
}  
.pdd-app{  
 margin: 8px;  
 padding: 20px 15px; /\*盒子的内边距为上下20px，左右15px\*/  
 border: 2px solid #ffedf1;  
}  
.pdd-app img{  
 width: 100%;  
 height: 100%;  
}  
.pdd-app-desc{ /\*设置文字的位置和大小\*/  
 line-height: 20px;  
 margin-top: 10px;  
 font-size:14px;  
}

防止页面拉动时出现头目录在页面之下的问题：赋予头目录较高的权限z-index: 999;即可

.header{  
 z-index: 999;  
 position: fixed;  
 top: 0;  
 left: 0;  
 width: 100%;  
 height: 103px;  
 border-bottom: 4px solid #ed435b;  
 background: #ffffff;  
}

如果出现所有的块不按照一行两个栏目来排列，则需要改变偶数个subject里的margin-right=0

.subject:nth-child(2n){  
 /\*让偶数个右边缘为0\*/  
 margin-right: 0;  
}

**一、元素大小**

offsetWidth

offsetHeight

\*\*注意：

1》 没有加入单位的（类型：数值number）

2》计算：

a)计算边框(外)（border）

b)计算内边距(padding)

<style>  
 #box{  
 width:300px;  
 height:300px;  
 background: blue;  
 border:20px solid #0c0c0c;  
 padding:10px;  
 margin:10px;  
 }  
</style>  
<body>  
<div id="box"></div>  
<script src="script.js"></script>  
</body>

Js:

var box = document.getElementById("box");  
// alert(box.offsetWidth); //300 number  
alert(box.offsetHeight);  
alert(box.offsetWidth);  
alert( getComputedStyle(box,null).width );/(盒子的宽度)

案例：实现box盒子随时间变换width的效果

<style>  
 #box{  
 width:300px;  
 height:300px;  
 background: blue;  
 border:5px solid #ff5500;  
 }  
</style>  
<body>  
<div id="box"></div>  
<script src="script.js"></script>  
</body>

Js:

// 写一个定时器， 每隔50毫秒，实现box盒子宽度增加XXpx  
setInterval(function(){  
 //box.style.width = box.offsetWidth-1+"px";  
 box.style.width = parseInt(getAttr(box,"width"))-2+"px";  
},50)  
function getAttr(dom,value){  
 if(dom.currentStyle){  
 return dom.currentStyle[value];  
 }else{  
 return getComputedStyle(dom,null)[value];  
 }  
}

二、元素偏移

offsetLeft

offsetTop

\*\*注意：

1》没有加入单位的（类型：数值number）

2》计算：

a)计算margin

3》定位的关系

1》如果没有定位,返回元素左侧距离浏览器左侧的偏移量

2》如果有定位.返回元素左侧距离和定义元素距离的偏移量

scrollTop

scrollLeft

<style>  
 \*{margin:0;padding:0}  
 #dv1{  
 width:300px;  
 background: pink;  
 height: 300px;  
 margin:50px;  
 position: relative;  
 }  
 #dv2{  
 width:100px;  
 height: 100px;  
 background: red;  
 position: relative;  
 left:60px;  
 top:30px;  
 }  
</style>  
<body>  
<div id="dv1">  
 <div id="dv2"></div>  
</div>  
<script src="script.js"></script>  
</body>

Js

var dv1 = document.getElementById("dv1");  
var dv2 = document.getElementById("dv2");  
//alert("left===>:"+dv1.offsetLeft);  
//alert("top===>:"+dv1.offsetTop);  
alert(dv2.offsetTop);  
alert(dv2.offsetLeft);

案例：实现读取scroll的数值

<style>  
 #box{  
 width:50px;  
 height: 70px;  
 overflow: scroll;  
 }  
</style>  
<body>  
<div id="box">  
 3i4y3igfhnkghro发蝴蝶结  
 3i4y3igfhnkghro发蝴蝶结  
 3i4y3igfhnkghro发蝴蝶结  
 3i4y3igfhnkghro发蝴蝶结  
 3i4y3igfhnkghro发蝴蝶结  
 3i4y3igfhnkghro发蝴蝶结  
 3i4y3igfhnkghro发蝴蝶结  
 3i4y3igfhnkghro发蝴蝶结  
 3i4y3igfhnkghro发蝴蝶结  
 v3i4y3igfhnkghro发蝴蝶结  
</div>  
<input type="button" id="btn" value="滚动">  
<script src="script.js"></script>  
</body>

Js:

var btn = document.getElementById("btn");  
btn.onclick = function(){  
 //alert(box.scrollLeft);  
 //alert(box.scrollTop);  
}

**一、获取宽度和高度(文档、可视区域)**

a)document.documentElement ===》文档 的 文档元素

b)document.body ===》文档 的 body

\*\*\*\*获取可视区域宽度和高度\*\*\*\*\*

clientWidth

clientHeight

<style>  
 \*{margin:0;padding:0}  
</style>  
<body style="height: 2000px;">  
<script src="scrpit.js"></script>  
</body>

Js:

// alert( document.documentElement.clientWidth ); //1920  
alert( document.body.clientWidth ); // 1904  
// alert( document.documentElement.clientHeight )//获取浏览器文档元素的宽度  
// alert( document.body.clientHeight )//获取设定值body 的宽度

二、标准模式和混杂模式

标准模式：document.documentElement

混杂模式：document.body

兼容处理：

**document.body || document.documentElement**

**案例：处理获取不同浏览器scroll滚动的数值**

<body style="height:2000px;">  
<input type="button" value="点击" id="btn">  
<script src="script.js"></script>  
</body>

Js:

var btn = document.getElementById("btn");  
btn.onclick = function(){  
 //alert( document.documentElement.scrollTop )  
 //alert( document.body.scrollTop )  
 var top = document.documentElement.scrollTop || document.body.scrollTop;  
 alert(top);  
}

**一、什么是事件：事情的起因**

二、事件流

1》事件冒泡：从内到外

<body>  
<div id="dv1" style="background: red;width:400px;height: 400px;">  
 <div id="dv2" style="background: blue;width:200px;height: 200px;"></div>  
</div>  
<script src="script.js"></script>  
</body>

Js:

var dv1 = document.getElementById("dv1");  
var dv2 = document.getElementById("dv2");  
dv1.onclick = function(){  
 alert(1);  
}  
dv2.onclick = function(){  
 alert(2);  
}

2》事件捕获：从外到内

**一、鼠标类事件**

Html:

<body>  
<div id="box">我是box</div>  
<script src="script.js"></script>  
</body>

1》onclick ====》单击事件

2》ondblclick===》双击事件

3》oncontextmenu ===》右击事件

Js:

var box = document.getElementById("box");  
box.onclick = function(){  
 alert(1);}  
box.ondblclick = function(){  
 alert("双击");  
}  
box.oncontextmenu = function(){  
 alert("右击");  
}

案例；鼠标事件4-8

Html：

<style>  
 #box{  
 width:300px;  
 height:300px;  
 background: red;  
 }  
</style>  
<body>  
<div id="box"></div>  
<script src="script.js"></script>  
</body>

Js:

4》onmouseover ====》鼠标移入（穿过）

5》onmouseout ====》鼠标移出

6》onmousemove ===》鼠标移入（一直会执行的代码）

var box = document.getElementById("box");  
box.onmouseover = function(){  
 alert("移入");  
}  
box.onmouseout = function(){  
 alert("移出");  
}  
box.onmousemove = function(){  
 alert(1);  
}

7》onmousedown ====》鼠标按下

8》onmouseup ===》鼠标释放(松开)

var box = document.getElementById("box");

box.onmousedown = function(){  
 alert("按下");  
}  
box.onmouseup = function(){  
 alert("放开手");  
}

**一、键盘类事件**

获取键盘上的值ASCALL：

document.onkeydown = function(e) {  
 alert(e.keyCode);  
}

1》onkeydown ===》键盘按下

2》onkeyup ===》键盘释放（松开）

document.onkeydown = function() {  
 alert("我的键盘已经按下了");  
}  
document.onkeyup = function(){  
 alert("释放键盘");  
}  
document.onkeydown = function(e){  
 if(e.keyCode==13){  
 alert("回车");  
 }  
}

\*\*\*\*\*\*\*\*\* keyCode ===>属性\*\*\*\*\*\*\*:得到键盘中的值

案例：实现给键特定的功能：

Html:

<style>  
 #box{  
 width:300px;  
 height:300px;  
 background: #ccc;  
 opacity: 0.5;  
 }  
</style>

Js:

//1>单机btn按钮 ====》出现一个弹出层  
//2>按下键盘中的esc键，删除掉弹出层  
var btn = document.getElementById("btn");  
btn.onclick = function(){  
 var oDiv = document.createElement("div");  
 oDiv.setAttribute("id","box");  
 document.body.appendChild(oDiv);  
 document.onkeydown = function(e){  
 if(e.keyCode==27){  
 document.body.removeChild(oDiv);  
 }  
 }  
}

案例：通过特定键改变实现盒子所在位置

Html:

<style>  
 #box{  
 width:300px;  
 height:300px;  
 background: olive;  
 position: absolute;  
 left:0;  
 top:0;  
 }  
</style>  
<body>  
<div id="box"></div>  
<script src="script.js"></script>  
</body>

Js:

var box = document.getElementById("box");  
document.onkeydown = function(e){  
 switch(e.keyCode){  
 case 37:  
 box.style.left = box.offsetLeft-10+"px";  
 break;  
 case 38:  
 box.style.top = box.offsetTop-10+"px";  
 break;  
 case 39:  
 box.style.left = box.offsetLeft+10+"px";  
 break;  
 case 40:  
 box.style.top = box.offsetTop+10+"px";  
 break;  
 }  
}

**一、其他类事件**

1》onload ===》加载（DOM加载完毕）

window.onload = function(){  
 var box = document.getElementById("box");  
 box.style.background = "red";  
}

2》onfocus ===》得到焦点

3》onblur ===》失去焦点

4》onresize ===》浏览器或者框架尺寸发生变化时触发

5》onscroll ===》滚动条发生变化时触发

Html:

<body style="height: 2000px;">  
<input type="text" id="txt">  
<script src="script.js"></script>  
</body>

Js:

var txt = document.getElementById("txt");  
txt.onfocus = function(){  
 txt.style.background = "red";  
}  
txt.onblur = function(){  
 txt.style.background = "blue";  
}  
window.onresize = function(){  
 alert(1);  
}  
window.onscroll = function(){  
 alert("滚动条发生变化了");  
}

6》onchange ===》值发生变化时触发

Html:

<body>  
<select name="" id="city">  
 <option value="">北京</option>  
 <option value="">天津</option>  
 <option value="">上海</option>  
</select>  
<script src="script.js"></script>  
</body>

Js:

var city = document.getElementById("city");  
city.onchange = function(){  
 alert("改变值了");  
}

**一、事件处理程序===》添加**

1》HTML事件处理程序

语法：把事件添加到了节点身上

缺点：HTML和js耦合(需要同时修改html和js才能实现功能的改变)

Html:

<body>  
<div id="box" onclick="alert( fn(20,10) )">我是box</div>  
<script src="script.js"></script>  
</body>

Js:

function fn(num1,num2){  
 return num1\*num2;  
}

2》DOM0级事件处理程序

语法：box.onclick = function(){}

缺点：同样的元素，同样的事件，后面的会覆盖前面

Html：

<body>  
<div id="box">我是box</div>  
<script src="script.js"></script>  
</body>

Js:

var box = document.getElementById("box");  
box.onclick = function(){  
 alert( fn(30,20) );  
}  
function fn(num1,num2){  
 return num1\*num2;  
}

box.onclick = function(){  
 alret(1);  
}  
box.onclick = function(){  
 alert(2);  
}

window.onload = function(){  
 alert(1);  
}  
window.onload = function(){  
 alert(2);  
}

3》DOM2级事件处理程序

语法：***box***.addEventListener(事件名称,函数,**false**|**true**)

参数1：不需要加on ===>click

参数3: false代表了冒泡

true代表了捕获

缺点：在IE（8或者8以下）不兼容

Html:

<body>  
<div id="box">我是box</div>  
<script src="script.js"></script>  
</body>

Js:

var box = document.getElementById("box");  
box.addEventListener("click",function(){  
 alert(1);  
},false);  
box.addEventListener("click",function(){  
 alert(2);  
},false);

4》IE事件处理程序  
 ***语法：box***.attachEvent(事件名称,函数)js

参数1》需要加入on

Html:

<body>  
<div id="box">我是box</div>  
<script src="script.js"></script>  
</body>

Js:

var box = document.getElementById("box");  
box.attachEvent("onclick",function(){  
 alert(1);  
})  
box.attachEvent("onclick",function(){  
 alert(2);  
})

5》跨浏览器事件处理程序(做兼容)

DOM0事件处理程序

DOM2事件处理程序

IE事件处理程序

案例：

Html:

<body>  
<div id="box">我是box</div>  
<script src="script.js"></script>  
</body>

Js：

var box = document.getElementById("box");  
function setEvent(dom,evt,fn){  
 if(dom.addEventListener){  
 dom.addEventListener(evt,fn,false);  
 }else if(dom.attachEvent){  
 dom.attachEvent("on"+evt,fn);  
 }else{  
 dom["on"+evt] = fn;  
 }  
}  
setEvent(box,"click",function(){  
 alert(1);  
})  
setEvent(box,"click",function(){  
 alert(2);  
})

**一、删除事件处理程序**

dom0级事件处理程序删除===》

语法：DOM对象.事件名称 = null

Html:

<body>  
<div id="box">我是box</div>  
<script src="script.js"></script>  
</body>

Js:

var box = document.getElementById("box");  
box.onclick = function(){  
 alert(1);  
}  
box.onclick = null;

dom2级事件处理程序删除===》

语法：DOM对象.removeEventListener(事件名称,函数,false|true)

参数1》事件名称不需要加入on

参数2》函数,添加和删除必须都是同一个函数，才有效

参数3》false===>冒泡 true===》捕获

Html:

<body>  
<div id="box">我是box</div>  
<script src="script.js"></script>  
</body>

Js:

var box = document.getElementById("box");  
function fn(){  
 alert(1);  
}  
box.addEventListener("click",fn,false);  
box.removeEventListener("click",fn,false);

IE事件处理程序删除===》

语法：DOM对象. attachEvent(事件名称,函数)

参数1》事件名称需要加入on

参数2》函数,添加和删除必须都是同一个函数，才有效

Html:

<body>  
<div id="box">我是box</div>  
<script src="script.js"></script>  
</body>

Js:

var box = document.getElementById("box");  
function fn(){  
 alert(222);  
}  
box.attachEvent("onclick",fn)  
box.detachEvent("onclick",fn)

案例：处理删除事件处理程序的兼容问题：

Html:

<body>  
<div id="box">我是box</div>  
<script src="script.js"></script>  
</body>

Js：

function removeEvent(dom,evt,fn){  
 if(dom.removeEventListener){  
 dom.removeEventListener(evt,fn,false);  
 }else if(dom.detachEvent){  
 dom.detachEvent("on"+evt,fn);  
 }else{  
 dom["on"+evt] = null;  
 }  
}

**一、事件对象的属性**

1》target(非IE)===》目标（返回对应的DOM节点）

2》srcElement（IE中）====》目标（返回对应的DOM节点）

处理兼容： 事件对象.target||事件对象.srcElement

Html:

<body>  
<div id="box">  
 我是box  
 <p>11111</p>  
 <span>22222</span>  
</div>  
<script src="script.js"></script>  
</body>

Js:

var box = document.getElementById("box");  
box.onclick = function(e){  
 var e = window.event||e;//处理IE和非IE浏览器的windows事件  
 //alert(e.target);  
 //alert(e.srcElement);  
 var tar = e.target || e.srcElement;  
 alert(tar);  
}

3》clientX ===》返回当前鼠标位置距离浏览器左侧的值

4》clientY ===》返回当前鼠标位置距离浏览器顶部的值

\*\*\*\*注意：返回的结果是没有加入单位的

Js:

document.onclick = function(e){  
 var e = window.event || e;  
 //alert(e.clientX);  
 alert(e.clientY);  
}

二、事件委托

原理：冒泡的原理

优点：

1》性能较好（提高）

2》即使后添加的也有效

Html:

<body>  
<input type="button" id="btn">  
<ul id="uls">  
 <li>1</li>  
 <li>2</li>  
 <li>3</li>  
</ul>  
<script src="script.js"></script>  
</body>

Js:

/\*我有一个快递（我要去拿） li  
 让张三兄弟帮我去拿一下快递 ul  
\*/  
var uls = document.getElementById("uls");  
var btn = document.getElementById("btn");//绑定按钮事件  
btn.onclick = function(){  
 var li = document.createElement("li");  
 uls.appendChild(li);  
}  
uls.onclick = function(e){  
 var e = window.event || e;  
 if(e.target.nodeName=="LI"){  
 e.target.style.background = "red";  
 }  
}  
/\*性能较差，功能添加，其属性不会添加\*/  
// var btn = document.getElementById("btn");  
// var lis = document.getElementsByTagName("li");//循环列表LI  
// var uls = document.getElementById("uls");  
// btn.onclick = function(){  
// var li = document.createElement("li");  
// uls.appendChild(li);  
// }  
// for(var i=0;i<lis.length;i++){  
// lis[i].onclick = function(){  
// this.style.background = "red";  
// }  
// }

**一、事件阻止冒泡**

语法：

事件对象.属性名称

非IE：e.stopPropagation

IE：e.**cancelBubble** = **true**;

案例：阻止冒泡兼容处理

Html：

<style>  
 #dv1{  
 width:300px;  
 height:300px;  
 background: red;  
 }  
 #dv2{  
 width:100px;  
 height:100px;  
 background: blue;  
 }  
</style>  
<body>  
<div id="dv1">  
 <div id="dv2"></div>  
</div>  
<script src="script.js"></script>  
</body>

Js:

var dv1 = document.getElementById("dv1"),  
 dv2 = document.getElementById("dv2");  
dv1.onclick = function(){  
 alert("dv1");  
}  
dv2.onclick = function(e){  
 var e = window.event || e;  
 //e.stopPropagation();  
 //e.cancelBubble = true;  
 if(e.stopPropagation){  
 e.stopPropagation();  
 }else{  
 e.cancelBubble = true;  
 }  
 alert("dv2");  
}

二、事件阻止默认行为

语法：

事件对象.属性名称

非IE：e.preventDefault()

IE：e.**returnValue** = **false**;

案例：阻止默认行为兼容处理

Html:

<body>  
<a href="http://www.baidu.com" id="oA">跳转页面</a>  
<div id="box">我是box</div>  
<script src="script.js"></script>  
</body>

Js:

var oA = document.getElementById("oA");  
oA.onclick = function(e){  
 var e = window.event || e;  
 e.preventDefault();  
}  
var box = document.getElementById("box");  
box.oncontextmenu = function(e){  
 var e = window.event || e;  
 //e.preventDefault();  
 e.returnValue = false;  
 alert(111);  
}  
document.oncontextmenu = function(e){  
 var e = window.event || e;  
 //e.preventDefault();  
 //e.returnValue = false;  
 if(e.preventDefault){  
 e.preventDefault();  
 }else{  
 e.returnValue = false;  
 }  
 alert(1111);  
}

案例：小人快跑：

Html:

<style>  
 #imgs{  
 position:absolute;  
 left:800px;  
 top:400px;  
 }  
</style>  
<body>  
<img src="images/down-0.png" alt="" id="imgs">  
<script src="script.js"></script>  
</body>

Js:

var imgs = document.getElementById("imgs");  
var index = 0;  
function fn(value){  
 index++;  
 if(index>4){  
 index=0;  
 }  
 imgs.src ="images/"+value+"-"+index+".png";  
}  
document.onkeydown = function(e){  
 var e= window.event|| e;  
 switch(e.keyCode){  
 case 37:  
 imgs.style.left = imgs.offsetLeft-10+"px";  
 fn("left") ;  
 break;  
 case 38:  
 imgs.style.top = imgs.offsetTop-10+"px";  
 fn("up");  
 break;  
 case 39:  
 imgs.style.left = imgs.offsetLeft+10+"px";  
 fn("right");  
 break;  
 case 40:  
 imgs.style.top = imgs.offsetTop+10+"px";  
 fn("down");  
 break;  
 }  
}

案例：图片放大：

Html:

<style>  
 \*{margin:0;padding: 0}  
 #box{//布局管理器的一个轻量级容器  
 width:1200px;  
 height: 700px;  
 border:1px solid #ccc;  
 margin:150px;  
 }  
 #min{  
 float:left;  
 width:400px;  
 height: 400px;  
 position: relative;  
 }  
 b{  
 position: absolute;  
 left:0;  
 top:0;  
 width:200px;  
 height: 200px;  
 background: blue;  
 opacity: 0.5;  
 display: none;  
 }  
 #max{  
 float:right;  
 width:400px;  
 height: 400px;  
 overflow: hidden;  
 display: none;  
 position: relative;  
 }  
 #imgs{  
 position: absolute;  
 left:0;  
 top:0;  
 }  
</style>  
<body style="height: 2000px">  
<div id="box">  
 <div id="min"><img src="imgs/min.jpg" alt=""><b></b></div>  
 <div id="max"><img src="imgs/max.jpg" alt="" id="imgs"></div>  
</div>  
<script src="script.js"></script>  
</body>

Js:

//1》鼠标移入到min这个盒子中，显示出来 放大图片的盒子和小阴影部分  
//2》阴影跟随鼠标移动====》  
var min = document.getElementById("min"),  
 max = document.getElementById("max"),  
 b = document.getElementsByTagName("b")[0],  
 imgs = document.getElementById("imgs");  
  
min.onmousemove = function(e){  
  
 b.style.display = "block";  
 max.style.display = "block";  
 var scrollLeft = document.body.scrollLeft || document.documentElement.scrollLeft;  
 var scrollTop = document.body.scrollTop || document.documentElement.scrollTop;  
 var x = e.clientX+scrollLeft-min.offsetLeft- b.offsetWidth/2;  
 var y = e.clientY+scrollTop-min.offsetTop - b.offsetHeight/2;  
 if(x<0){  
 x=0;  
 }else if(x>min.offsetWidth-b.offsetWidth){  
 x=min.offsetWidth-b.offsetWidth;  
 }  
 if(y<0){  
 y=0;  
 }else if(y>min.offsetHeight-b.offsetHeight){  
 y=min.offsetHeight-b.offsetHeight;  
 }  
 b.style.left =x+"px";  
 b.style.top =y+"px";  
   
 imgs.style.left = -2\*x+"px";  
 imgs.style.top = -2\*y+"px";  
}  
min.onmouseout = function(){  
 b.style.display = "none";  
 max.style.display = "none";  
}

案例：拖拽盒子改变位置而大小不变：

Html:

<style>  
 \*{margin:0;padding:0}  
 #box{  
 width:300px;  
 height: 300px;  
 background: blue;  
 position: absolute;  
 left:0px;  
 top:0px;  
 }  
</style>  
<body>  
<div id="box"></div>  
<script src="script.js"></script>  
</body>

Js:包含onmousedown onmousemove onmouseup

//1> onmousedown onmousemove onmouseup  
var box = document.getElementById("box");  
box.onmousedown = function(e){  
 var e = window.event || e;  
 var x = e.clientX - box.offsetLeft;  
 var y = e.clientY - box.offsetTop;  
 document.onmousemove = function(e){  
 var e = window.event || e;  
 box.style.left = e.clientX - x+"px";  
 box.style.top = e.clientY - y+"px";  
 }  
}  
box.onmouseup = function(){  
 document.onmousemove = null;  
}

案例：拖拽盒子改变其大小：

Html：

<style>  
 \*{margin:0;padding:0}  
 #dv1{  
 width:300px;  
 height: 300px;  
 background: blue;  
 position: relative;  
 }  
 #dv2{  
 width:50px;  
 height: 50px;  
 background: red;  
 position: absolute;  
 bottom:0px;  
 right:0px;  
 }  
</style>  
<body>  
<div id="dv1">  
 <div id="dv2"></div>  
</div>  
<script src="script.js"></script>  
</body>

Js:

var dv2 = document.getElementById("dv2"),  
 dv1 = document.getElementById("dv1");  
dv2.onmousedown = function(e){  
 var e = window.event || e;  
 var x = e.clientX;  
 var y = e.clientY;  
 var wdith = dv1.offsetWidth;  
 var height = dv1.offsetHeight;  
 document.onmousemove = function(e){  
 var e = window.event || e;  
 dv1.style.width =wdith + e.clientX-x+"px";  
 dv1.style.height =height + e.clientY-y+"px";  
 }  
}  
dv2.onmouseup = function(){  
 document.onmousemove = null;  
}

案例：飞机发射子弹：

Html：

<style>  
 img{  
 position:absolute;  
 left:0px;  
 top:0px;  
 }  
</style>  
<body>  
<img src="images/2.png" alt="" id="pos0" style="display: none">  
<img src="images/2.png" alt="" id="pos1" style="display: none">  
<img src="images/2.png" alt="" id="pos2" style="display: none">  
<img src="images/2.png" alt="" id="pos3" style="display: none">  
<img src="images/2.png" alt="" id="pos4" style="display: none">  
<img src="images/1.gif" alt="" id="img">  
<script src="script.js"></script>  
</body>

Js:

//1>规定飞机的走向 键盘事件，  
//2>发射子弹===》 子弹位置: left top  
var img = document.getElementById("img");  
  
document.onkeydown = function(e){  
 var e = window.event || e;  
 switch(e.keyCode){  
 case 37:  
 img.style.left = img.offsetLeft-10+"px";  
 break;  
 case 38:  
 img.style.top = img.offsetTop-10+"px";  
 break;  
 case 39:  
 img.style.left = img.offsetLeft+10+"px";  
 break;  
 case 40:  
 img.style.top = img.offsetTop+10+"px";  
 break;  
 case 32:  
 //发射子弹  
 game(img.offsetTop,img.offsetLeft+img.offsetWidth/2);  
 break;  
 }  
}  
function game(top,left){  
 for(var i=0;i<5;i++){  
 var pos = document.getElementById("pos"+i);  
 if(pos.style.display == "none"){  
 pos.style.display = "block";  
 pos.style.left = left+"px";  
 pos.style.top = top+"px";  
 break;  
 }  
 }  
}  
setInterval(function(){  
 for(var i=0;i<5;i++){  
 var pos = document.getElementById("pos"+i);  
 pos.style.top = pos.offsetTop-10+"px";  
 if(pos.offsetTop<-7){  
 pos.style.display = "none";  
 }  
 }  
},50);

案例：省市区连锁切换效果

Html:

<body>  
省:<select id="pro"></select>  
市:<select id="city"></select>  
区:<select id="county"></select>  
<script src="script.js"></script>  
</body>

Js：

var arr = [  
  
 "北京",[  
 "海淀",["海淀1","海淀2","海淀3"],  
 "朝阳",["朝阳1","朝阳2","朝阳3"]  
 ],  
 "山东",[  
 "青岛",["青岛1","青岛2","青岛3"],  
 "济南",["济南1","济南2","济南3"]  
 ],  
 "河北",[  
 "张家口",["张家口1","张家口2","张家口3"],  
 "石家庄",["石家庄1","石家庄2","石家庄3"]  
 ]  
];  
var pro = document.getElementById("pro");  
var city = document.getElementById("city");  
var county = document.getElementById("county");  
for(var i=0;i<arr.length;i++){  
 if(typeof arr[i]=="string"){  
 /\*var option = document.createElement("option");  
 option.innerHTML = arr[i];  
 pro.appendChild(option);\*/  
 pro.add( new Option(arr[i],arr[i]));  
 //在pro节点中， add ==>表示添加的意思 new Option表示创建出来option节点了。 参数1》节点中的值，参数2》vlaue值  
 }  
}  
pro.onchange = function(){  
 for(var i=0;i<arr.length;i++){  
 if(typeof arr[i]=="string"){  
 if(pro.value==arr[i]){  
 var brr = arr[i+1];  
 city.innerHTML = "";  
 for(var j=0;j<brr.length;j++){  
 if(typeof brr[j]=="string"){  
 city.add(new Option(brr[j],brr[j]));  
 }  
 }  
 }  
 }  
  
 }  
}  
city.onchange = function(){  
 for(var i=0;i<arr.length;i++){  
 if(typeof arr[i]!="string"){  
 var brr = arr[i];  
 for(var j=0;j<brr.length;j++){  
 if(typeof brr[j]=="string"){  
 if(brr[j]==city.value){  
 var crr = brr[j+1];  
 county.innerHTML = "";  
 for(var k=0;k<crr.length;k++){  
 county.add( new Option(crr[k],crr[k]) );  
 }  
 }  
 }  
 }  
 }  
 }  
}

购物车案例：

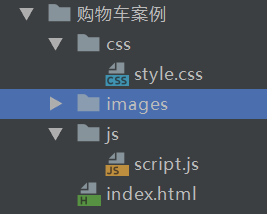
功能介绍：

//1》如果复选框选中，把对应的数量和价格，放入到结算中  
//2》结算功能，都在什么地方使用（数量增加或者减少）(删除)(单价\*数量=总价) ====》封装成一个函数实现  
//3》点击按钮,数量增加，小计值发生变化 删除  
//4》已选商品的添加===》取消选择

功能：获取333的方法：

<table id="tab">  
 <tbody>  
 <tr>  
 <td>333</td>  
 </tr>  
 </tbody>  
</table>  
<script>  
var tab = document.getElementById("tab");  
alert(tab.getElementsByTagName("tbody")[0].getElementsByTagName("tr")[0].getElementsByTagName("td")[0].innerHTML);  
alert(tab.tBodies[0].rows[0].cells[0].innerHTML);  
//tbody ===> tBodies  
//tr===>rows  
//td ==>cells  
</script>

目录介绍：



Index.html:

<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="UTF-8">  
 <title>demo</title>  
 <link rel="stylesheet" href="css/style.css"/>  
</head>  
<body>  
<table id="cartTable">  
 <thead>  
 <tr>  
 <th><label><input class="check-all check" type="checkbox"/>&nbsp;全选</label></th>  
 <th>商品</th>  
 <th>单价</th>  
 <th>数量</th>  
 <th>小计</th>  
 <th>操作</th>  
 </tr>  
 </thead>  
 <tbody>  
 <tr>  
 <td class="checkbox"><input class="check-one check" type="checkbox"/></td>  
 <td class="goods"><img src="images/1.jpg" alt=""/><span>Casio/卡西欧 EX-TR350</span></td>  
 <td class="price">5999.88</td>  
 <td class="count"><span class="reduce"></span><input class="count-input" type="text" value="1"/><span class="add">+</span></td>  
 <td class="subtotal">5999.88</td>  
 <td class="operation"><span class="delete">删除</span></td>  
 </tr>  
 <tr>  
 <td class="checkbox"><input class="check-one check" type="checkbox"/></td>  
 <td class="goods"><img src="images/2.jpg" alt=""/><span>Canon/佳能 PowerShot SX50 HS</span></td>  
 <td class="price">3888.50</td>  
 <td class="count"><span class="reduce"></span><input class="count-input" type="text" value="1"/><span class="add">+</span></td>  
 <td class="subtotal">3888.50</td>  
 <td class="operation"><span class="delete">删除</span></td>  
 </tr>  
 <tr>  
 <td class="checkbox"><input class="check-one check" type="checkbox"/></td>  
 <td class="goods"><img src="images/3.jpg" alt=""/><span>Sony/索尼 DSC-WX300</span></td>  
 <td class="price">1428.50</td>  
 <td class="count"><span class="reduce"></span><input class="count-input" type="text" value="1"/><span class="add">+</span></td>  
 <td class="subtotal">1428.50</td>  
 <td class="operation"><span class="delete">删除</span></td>  
 </tr>  
 <tr>  
 <td class="checkbox"><input class="check-one check" type="checkbox"/></td>  
 <td class="goods"><img src="images/4.jpg" alt=""/><span>Fujifilm/富士 instax mini 25</span></td>  
 <td class="price">640.60</td>  
 <td class="count"><span class="reduce"></span><input class="count-input" type="text" value="1"/><span class="add">+</span></td>  
 <td class="subtotal">640.60</td>  
 <td class="operation"><span class="delete">删除</span></td>  
 </tr>  
 </tbody>  
</table>  
<div class="foot" id="foot">  
 <label class="fl select-all"><input type="checkbox" class="check-all check"/>&nbsp;全选</label>  
 <a class="fl delete" id="deleteAll" href="javascript:;">删除</a>  
 <div class="fr closing">结 算</div>  
 <div class="fr total">合计：￥<span id="priceTotal">0.00</span></div>  
 <div class="fr selected" id="selected">已选商品<span id="selectedTotal">0</span>件<span class="arrow up">︽</span><span class="arrow down">︾</span></div>  
 <div class="selected-view">  
 <div id="selectedViewList" class="clearfix">  
 <div><img src="images/1.jpg"><span>取消选择</span></div>  
 </div>  
 <span class="arrow">◆<span>◆</span></span>  
 </div>  
</div>  
<script type="text/javascript" src="js/script.js"></script>  
</body>  
</html>

Script.js:

//全选class : check-all check  
//单选class : check-one check  
var check = getClass("check");  
var cartTable = document.getElementById("cartTable");  
var tr = cartTable.tBodies[0].rows;  
var selectedTotal = document.getElementById("selectedTotal");  
var priceTotal = document.getElementById("priceTotal");  
for(var i=0;i<check.length;i++){  
 check[i].onclick = function(){  
 if(this.className=="check-all check"){  
 for(var i=0;i<check.length;i++){  
 check[i].checked = this.checked;  
 }  
 }  
 getTotal();  
 }  
}  
//函数==》实现获取class名称的  
function getClass(check){  
 var arr = [];  
 var doms = document.getElementsByTagName("\*");  
 for(var i=0;i<doms.length;i++){  
 if(doms[i].className==check || doms[i].className.indexOf(" "+check)!=-1 || doms[i].className.indexOf(check+" ")!=-1){  
 arr.push(doms[i]);  
 }  
 }  
 return arr;  
}  
function getTotal(){  
 //结算函数功能  
 var select = 0;  
 var price =0;  
 for(var i=0;i<tr.length;i++){  
 //如果是复选框选中的状态，才能把值放入进去  
 if(tr[i].getElementsByTagName("input")[0].checked){  
 select+= parseInt(tr[i].getElementsByTagName("input")[1].value);  
 price+=parseFloat(tr[i].cells[4].innerHTML);  
 }  
 }  
 selectedTotal.innerHTML = select;  
 priceTotal.innerHTML = price.toFixed(2);  
}

Style.css:

\* {  
 /\*通配符样式,一般用于公共样式的书写,会全局定义\*/  
 margin: 0;  
 padding: 0;  
}  
a {  
 color: #666;  
 text-decoration: none;  
}  
body {  
 padding: 20px;  
 color: #666;  
}  
.fl{  
 float: left;  
}  
.fr {  
 float: right;  
}  
table {  
 border-collapse: collapse;/\*用于表格属性\*/  
 border-spacing: 0;  
 border: 0;  
 text-align: center;  
 width: 937px;  
}  
th, td {  
 border: 1px solid #CADEFF;  
}  
th {  
 background: #e2f2ff;  
 border-top: 3px solid #a7cbff;  
 height: 30px;  
}  
td {  
 padding: 10px;  
 color: #444;  
}  
tbody tr:hover {  
 background: RGB(238,246,255);  
}  
.checkbox {  
 width: 60px;  
}  
.goods {  
 width: 300px;  
}  
.goods span {  
 width: 180px;  
 margin-top: 20px;  
 text-align: left;  
 float: left;  
}  
.price {  
 width: 130px;  
}  
.count {  
 width: 90px;  
}  
.count .add, .count input, .count .reduce {  
 float: left;  
 margin-right: -1px;  
 position: relative;  
 z-index: 0;  
}  
.count .add, .count .reduce {  
 height: 23px;  
 width: 17px;  
 border: 1px solid #e5e5e5;  
 background: #f0f0f0;  
 text-align: center;  
 line-height: 23px;  
 color: #444;  
}  
.count .add:hover, .count .reduce:hover {  
 color: #f50;  
 z-index: 3;  
 border-color: #f60;  
 cursor: pointer;  
}  
.count input {  
 width: 50px;  
 height: 15px;  
 line-height: 15px;  
 border: 1px solid #aaa;  
 color: #343434;  
 text-align: center;  
 padding: 4px 0;  
 background-color: #fff;  
 z-index: 2;  
}  
.subtotal {  
 width: 150px;  
 color: red;  
 font-weight: bold;  
}  
.operation {  
 width: 80px;  
}  
.operation span:hover, a:hover {  
 cursor: pointer;  
 color: red;  
 text-decoration: underline;  
}  
img {  
 width: 100px;  
 height: 80px;  
 /\*border: 1px solid #ccc;\*/  
 margin-right: 10px;  
 float: left;  
}  
  
.foot {  
 width: 935px;  
 margin-top: 10px;  
 color: #666;  
 height: 48px;  
 border: 1px solid #c8c8c8;  
 background-color: #eaeaea;  
 background-image:linear-gradient(RGB(241,241,241),RGB(226,226,226));  
 position: relative;  
 z-index: 8;  
}  
.foot div, .foot a {  
 line-height: 48px;  
 height: 48px;  
}  
.foot .select-all {  
 width: 100px;  
 height: 48px;  
 line-height: 48px;  
 padding-left: 5px;  
 color: #666;  
}  
.foot .closing {  
 border-left: 1px solid #c8c8c8;  
 width: 100px;  
 text-align: center;  
 color: #000;  
 font-weight: bold;  
 background: RGB(238,238,238);  
 cursor: pointer;  
}  
.foot .total{  
 margin: 0 20px;  
 cursor: pointer;  
}  
.foot #priceTotal, .foot #selectedTotal {  
 color: red;  
 font-family: "Microsoft Yahei";  
 font-weight: bold;  
}  
.foot .selected {  
 cursor: pointer;  
}  
.foot .selected .arrow {  
 position: relative;  
 top:-3px;  
 margin-left: 3px;  
}  
.foot .selected .down {  
 position: relative;  
 top:3px;  
 display: none;  
}  
 .show .selected .down {  
 display: inline;  
}  
 .show .selected .up {  
 display: none;  
}  
.foot .selected:hover .arrow {  
 color: red;  
}  
.foot .selected-view {  
 width: 935px;  
 border: 1px solid #c8c8c8;  
 position: absolute;  
 height: auto;  
 background: #ffffff;  
 z-index: 9;  
 bottom: 48px;  
 left: -1px;  
 display:none;  
}  
.show .selected-view {  
 display: block;  
}  
.foot .selected-view div{  
 height: auto;  
}  
.foot .selected-view .arrow {  
 font-size: 16px;  
 line-height: 100%;  
 color:#c8c8c8;  
 position: absolute;  
 right: 330px;  
 bottom: -9px;  
}  
.foot .selected-view .arrow span {  
 color: #ffffff;  
 position: absolute;  
 left: 0px;  
 bottom: 1px;  
}  
#selectedViewList {  
 padding: 20px;  
 margin-bottom: -20px;  
}  
#selectedViewList div{  
 display: inline-block;  
 position: relative;  
 width: 100px;  
 height: 80px;  
 border: 1px solid #ccc;  
 margin: 10px;  
}  
#selectedViewList div span {  
 display: none;  
 color: #ffffff;  
 font-size: 12px;  
 position: absolute;  
 top: 0px;  
 right: 0px;  
 width: 60px;  
 height: 18px;  
 line-height: 18px;  
 text-align: center;  
 background: RGBA(0,0,0,.5);  
 cursor: pointer;  
}  
#selectedViewList div:hover span {  
 display: block;  
}

**正则表达式**

一、正则表达式是什么

规则表达式

二、正则表达式的应用场景

1》表单验证

2》解析url参数

3》筛选....

三、正则表达式的定义

1》字面量形式 ===》 //

2》构造函数形式 ===》new RegExp()

/\*  
var reg = /a/;  
alert(reg);  
\*/  
var reg = new RegExp("b");  
alert(reg);

\*\*\*\*\*区别

字面量形式内不能够放入变量|函数..

构造函数形式内可以放入变量|函数

四、正则表达式的方法

1》test：

功能：看内容是否与正则匹配

返回:布尔值,

匹配==》true

不匹配===>false

2》exec

功能：看内容是否与正则匹配

返回：匹配==》数组

不匹配==》null

\*\*\*\*\*在很多语言中，都有正则表达式:php java javascript...

/\*  
var reg = /a/;  
var str = "a";  
\*/  
//alert( reg.test(str) );  
//alert(typeof reg.exec(str) );  
var num1 = 1;  
//var reg1 = /"+num1+"/;  
var reg1 = new RegExp(num1);  
var str1 = "1";  
alert(reg1);  
alert( reg1.test(str1) );

一、正则表达式的元字符

[] ===》匹配括号内的任意“一个”字符

var reg = /[abc]/; //匹配中括号内的任意一个字符  
var str = "c";  
alert(reg.test(str));

[^] ===》除了括号内的任意字符

var reg1 = /[^abc]/;  
var str1 = "u";  
alert(reg1.test(str1));

[0-9] ===》匹配0至9的任意数字(范围可变)

var reg2 = /[0-9]/;  
var num2 = 5;  
alert(reg2.test(num2));

[a-z] ===》匹配a至z的任意小写字母

var reg3 = /[a-z]/;  
var reg3 = /[A-Z]/;  
var str3 = "A";  
alert(reg3.test(str3));

[A-Z] ===》匹配A至Z的任意大写字母（同上）

var reg4 = /[0-9a-zA-Z]/; //数字 大小写字母

（条件可以多种）

\d ===》匹配数字 ：[0-9]

\D ===》匹配非数字

var reg1 = /\d/;  
var reg1 = /\D/;  
var num1 = 5;  
alert( reg1.test(num1) );

\w ===》匹配数字、字母、下划线 : [0-9a-zA-Z\_]

\W ===》匹配非数字、字母、下划线

var reg2 = /\w/;  
var reg2 = /\W/;  
var str2 = "\_";  
alert(reg2.test(str2));

\s ===》匹配空格

\S ===》非空格

var reg3 = /\s/;  
var reg3 = /\S/;  
var str3 = " ";  
alert( reg3.test(str3) );

\b ===》匹配边界

var reg4 = /\ba/;  
var str4 = " a";  
alert( reg4.test(str4) );

. ===》除换行符以外的任意字符

var reg5 = /./;  
var str5 = "\n";  
alert( reg5.test(str5) );

二、转义符

\

var reg6 = /\./; //.有特殊含义、除了换行符以外的任意字符  
var str6 = ".";  
alert( reg6.test(str6) );

**一、正则表达式限定符**

^ ===》以字符串的起始(开始)

$===》以字符串的结束(结束)

var reg = /\d/; // \d ====>数字 没有限定符num中，只要有数字即可  
var num = "5a";  
alert(reg.test(num));  
var reg1 = /\w/; //数字 字母 下划线  
var str1 = "^\_^";  
alert( reg1.test(str1) );  
var reg2 = /^\d\d$/; // 以数字开始 以数字结束 ===》数字只有一位数  
var num2 = "55";  
alert( reg2.test(num2) );

二、正则表达式重复

{n} ====>匹配重复N次

var reg3 = /^\d{11}$/; // 手机号===>11位数  
var num3 = 1234567890;  
alert( reg3.test(num3) );

{n,} ====》匹配重复最少n次，最多不限

//匹配数字最少匹配到6次 , 最多不限  
var reg4 = /^\d{6,}$/;  
var str4 = "123454655555555555789";  
alert( reg4.test(str4) )

{n,m} ====》匹配重复最少n次，最多m次

//用户名最少6个字符，最多12个字符  
var reg5 = /^\w{6,12}$/;  
var str5 = "123ab\_\_44444a";  
alert( reg5.test(str5) );

\* ====》匹配重复最少0次，最多不限

var reg6 = /^\d\*$/;  
//匹配数字，最少可以没有，最多不限制  
var num6 = "";  
alert( reg6.test(num6) );

+ ====》匹配重复最少1次，最多不限

var reg7 = /^\d+$/;  
//匹配数字，最少一次，最多不限制  
var num7 = "1111111111111";  
alert( reg7.test(num7) );

? ====》匹配重复最少0次，最多一次

var reg8 = /^\d?$/;  
//匹配数字，最少0次，最多一次  
var num8 = "11";  
alert( reg8.test(num8) );

**一、选择符**

| ===》或

var reg1 = /com|cn/;  
var str1 = "com";  
alert( reg1.test(str1) );

二、修饰符

i ===》不区分大小写

var reg1 = /[A-Z]/i; // i===》自变量方法：不区分大小写  
var str1 = "b";  
alert( reg1.test(str1) );

var reg1 = new RegExp("[A-Z]","i");//构造函数的方法：不区分大小写  
var str1 = "b";  
alert( reg1.test(str1) );

g ===》全文匹配（全局模式）

var reg1 = /a/g;  
var str1 = "abc a a a";  
var str = str1.replace(reg1,"#");  
document.write(str);

var reg1 = /baidu/g;  
var str1 = "1.baidu\n2.baidu\n3.baidu";  
var str = str1.replace(reg1,"#");  
document.write(str);

m ===》换行匹配

var reg1 = /^\d/gm;//m:换行也能匹配  
var str1 = "1.baidu\n2.baidu\n3.baidu";  
var str = str1.replace(reg1,"#");  
document.write(str);  
  
var reg2 = /[a-z]/igm;  
var str2 = "a a a a A A";  
var str = str2.replace(reg2,"#");  
document.write(str);

**一、分组**

() ==》把内容组合起来

var reg1 = /^(abc)$/;  
var str1 = "abc";  
alert( reg1.test(str1) );

var reg2 = /^abc{3}$/; //C重复必须是3次  
var str2 = "abccc";  
alert( reg2.test(str2) );

var reg2 = /^(abc){3}$/; //abc重复必须是3次  
var str2 = "abcabcabc";  
alert( reg2.test(str2) );

二、字符类

RegExp.$1...$99

var reg1 = /(a)/;  
var str1 = "abc abc abc";  
reg1.test(str1);  
alert( RegExp.$1 );//匹配到第一个分组内的内容

注意：

1》要使用字符类，必须使用一下正则

2》要使用字符类返回结果，必须正则中有分组

var reg2 = /(baidu) (taobao) (jd)/;  
var str2 = "baidu taobao jd";  
reg2.exec(str2);  
//运行正则  
alert( RegExp.$3 );

var reg3 = /(taobao)/;  
var str3 = "我是taobao";  
var str = str3.replace(reg3,"<b style='color:red'>$1</b>");  
document.write(str);

**一、字符串匹配正则方法**

1》search

如果匹配函数下标位置

如果不匹配返回-1

\*\*\*\*\*\*不支持全局\*\*\*\*\*

var reg1 = /a/g;  
var str1 = "b a c a d";  
var sh = str1.search(reg1);  
document.write(sh);

2》match

如果匹配返回数组

如果不匹配返回null

var reg3 = /a/g;  
var str3 = "ab ac ad";  
var arr = str3.match(reg3);  
document.write(arr);

match和exec 区别

var reg2 = /a/g;  
var str2 = "ab ac ad";  
//var arr = reg2.exec(str2); //正则.exec这个方法==》不支持全局  
//document.write(arr);

var arr = null;  
while(arr=reg2.exec(str2)){  
 document.write(arr);  
}

match支持全局

exec不支持全局，可以用循环遍历出来

3》replace

var reg4 = /b/g;  
var str4 = "ab bb cb";  
var str = str4.replace(reg4,"\*");  
document.write(str);

语法：字符串.方法名称

案例：验证手机号 邮箱 身份证

Html:

<body>  
Tel:<input type="text" id="userTel"><span></span>  
<hr />  
用户名: <input type="text" id="userName"><span></span>  
<hr />  
邮箱: <input type="text" id="userEmail"><span></span>  
<hr />  
身份证: <input type="text" id="userS"><span></span>  
<script src="script.js"></script>  
</body>

Js:

var userTel = document.getElementById("userTel");  
//验证手机号的  
//手机号是11位数  
//手机号都是1开头的  
//手机号的第二位。 3 4 5 6 7 8  
userTel.onblur = function(){  
 var regTel = /^1[3-8]\d{9}$/;  
 if( regTel.test(this.value) ){  
 this.nextElementSibling.innerHTML = "手机号正确";  
 }else{  
 this.nextElementSibling.innerHTML = "手机号输入有误";  
 }  
}  
//验证用户名的  
//可以有： 数字 字母 下划线  
//位数必须是： 6-12  
var userName = document.getElementById("userName");  
userName.onblur = function(){  
 var regName = /^\w{6,12}$/;  
 if( regName.test(this.value) ){  
 this.nextElementSibling.innerHTML = "用户名正确";  
 }else{  
 this.nextElementSibling.innerHTML = "用户名输入有误";  
 }  
}  
//验证邮箱的  
//zhang\_san@163.com  
//zhangsan123@qq.cn  
//zhan1ab@126.cc  
// zhangsan\_123abC @ 163.com  
var userEmail = document.getElementById("userEmail");  
userEmail.onblur = function(){  
 var regEmail = /^\w+@\w{1,6}\.[0-9a-zA-Z]{2,5}$/;  
  
 if( regEmail.test(this.value) ){  
 this.nextElementSibling.innerHTML = "邮箱正确";  
 }else{  
 this.nextElementSibling.innerHTML = "邮箱输入有误";  
 }  
}  
//验证身份证的  
var userS = document.getElementById("userS");  
userS.onblur = function(){  
 var regS = /^\d{17}[\dX]$/;  
 if( regS.test(this.value) ){  
 this.nextElementSibling.innerHTML = "证件正确";  
 }else{  
 this.nextElementSibling.innerHTML = "证件输入有误";  
 }  
}

封装一个函数类似于trim的功能：

function t(str){  
 //把前空格和后空格，替换成 "空"  
 var reg = /^\s\*|\s\*$/g;  
 var str1 = str.replace(reg,"");  
 return str1;  
}  
console.log( t(" A B C ") );

//封装一个函数，获取class名称的  
//1》把页面上所有的dom节点，全部选取  
//2》循环所有节点，if判断，如果页面上节点的className==lis  
//3》把节点添加到数组中

function getClass(cName) {  
 var arr = [];  
 var doms = document.getElementsByTagName("\*");  
 var reg = new RegExp("\\b"+cName+"\\b");  
 for(var i=0;i<doms.length;i++){  
 if(reg.test(doms[i].className)){  
 arr.push(doms[i]);  
 }  
 }  
 return arr;  
}  
var li = getClass("lis");  
for(var i=0;i<li.length;i++){  
 li[i].style.background = "red";  
}

案例：封装一个函数,实现解析url参数

//封装一个函数,实现解析url参数  
function fn(c){  
 var str = "https://www.baidu.com/s?wd=%E4%BD%A0&rsv\_spt=1&rsv\_iqid=0xf18aea8a00073315&issp=1&f=8&rsv\_bp=0";  
 var reg = new RegExp(c+"=([^&]\*)");//匹配任意数量的 除了 & 以外的字符  
 reg.test(str);  
 return RegExp.$1;  
}  
document.write( fn("rsv\_bp") );//传递RSV\_bp的结果

补充：**var reg = new RegExp("(^|&)" + name + “=([^&]\*)(&|$)”, “i”);的详细介绍：**

匹配以 &name=开头 或者 name=开头 中间为任意多个除了 & 以外的字符 一旦遇到 & 或者 空白 就中止取值 **^n 的用法** **:** 匹配任何开头为n的字符串

**"(^|&)"+name 的含义** **:** 匹配以&name开头 或者 空白name 开头的字符串

(其实|前面有空白也就是什么都没有,也就是通配url中第一个参数或第二个参数的情况)

**[^abc]的用法** **:** 匹配除了abc以外的其他字符

**[^&]的含义** **:** 匹配除了 & 以外的任意字符

**[^&]\*的含义** **:** 匹配任意数量的 除了 & 以外的字符 (也就是说name=后面的值一旦有&则就不向后取值了)

**n$ 的用法** **:** 匹配任何结尾为 n 的字符串

**(&|$)的含义** **:** 匹配任何以 & 结尾 或者 以空白结尾的参数 (也就是说匹配了该参数在中间或者最后一个参数的情况)

**([^&]\*)(&|$)的含义** **:** 匹配 & 之外的以任意多个 参数值,一旦遇到 & 或者 空白 就中止

**最后就是第二个参数:**

**i** **:** 执行对大小写不敏感的匹配

**g** **:** 执行全局匹配（查找所有匹配而非在找到第一个匹配后停止）

**m** **:** 执行多行匹配

**一、基本类型和引用类型的值**

a)动态的属性

创建对象：

1》构造函数创建:new Object

2》字面量创建:{}

var obj = new Object();  
alert(obj);  
var obj1 = {};  
alert(obj1);

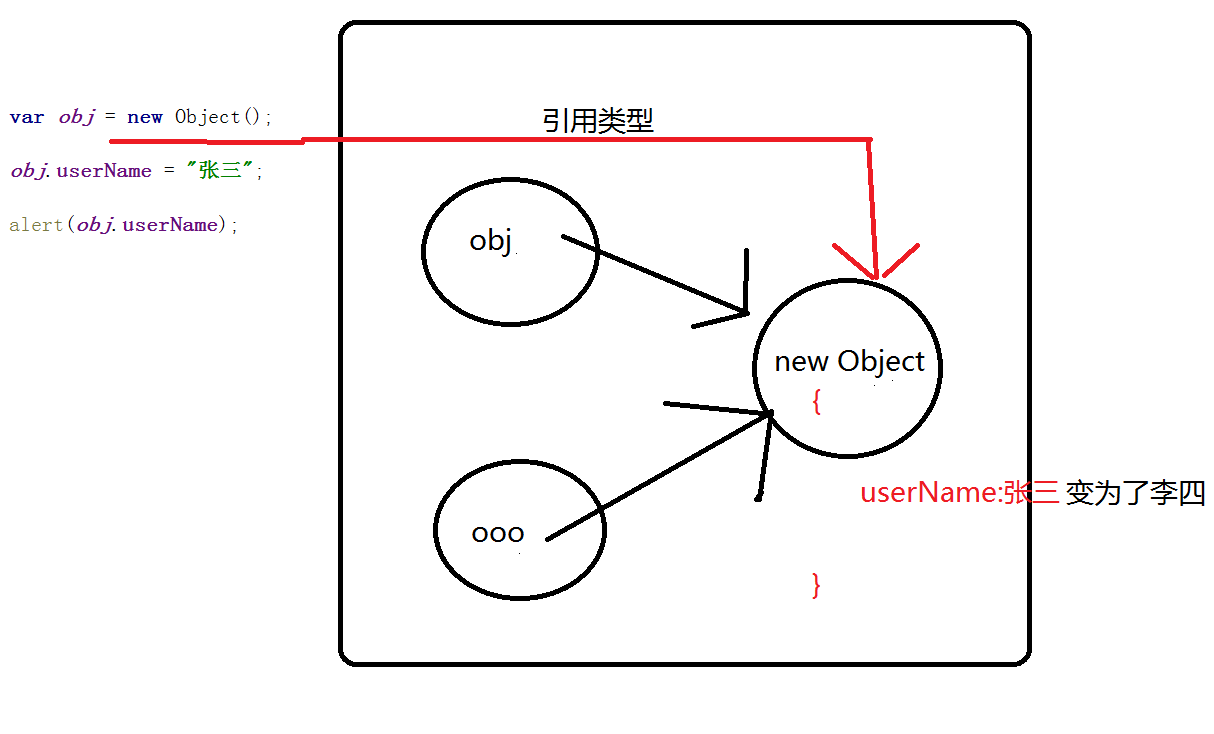
对象添加属性：对象.属性名称=值

对象添加方法：对象.方法名称=function(){}

var obj2 = new Object();  
obj2.userName = "张三"; //我给对象添加了一个属性（userName） 值：张三  
obj2.run = function(){  
 return "run方法";  
}  
alert( obj2.userName ); //对象.属性名称  
alert( obj2.run() );

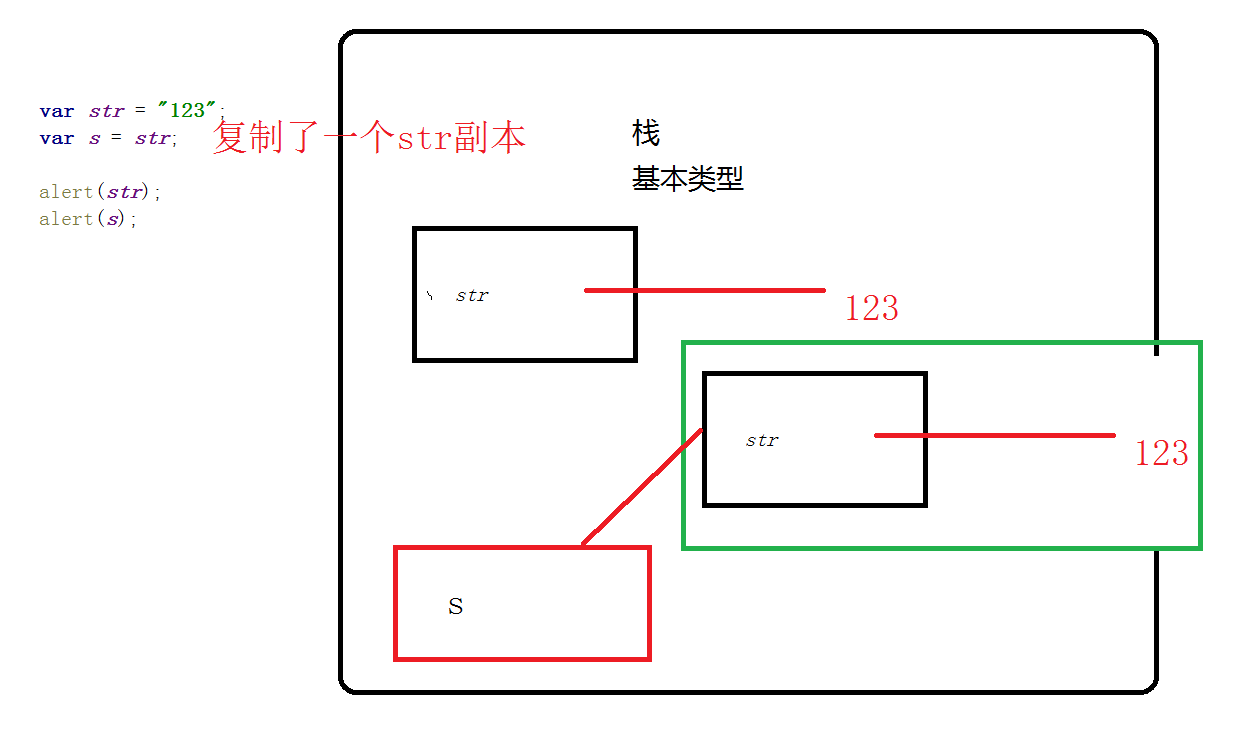
b)复制变量值

1》引用类型：如果是赋值的状态，引用类型是公用一个对象则互相影响



var obj = new Object();  
var ooo = obj;  
obj.userName = "张三";  
  
alert(obj.userName);  
alert(ooo.userName);  
ooo.userName = "李四";  
alert(ooo.userName);  
alert(obj.userName);

2》基本类型：如果是赋值状态，相当于是复制了一个副本（克隆版） 2个变量都是单独的则互相不影响



var str = "123";  
var s = str;  
alert(str); //123  
alert(s); //123  
  
s = "你好js";  
alert(s); //你好js  
alert(str); //123

c)传递参数

函数(参数) : 参数===》只能传递值

function fn(o){ //引用类型 ===>只能是值  
 o.userName = "张三";  
}  
  
var obj = new Object();  
fn(obj); //对象===》引用类型  
alert(obj.userName);  
  
function fn(o){  
 o.userName = "张三";  
 o = new Object();  
 o.userName = "李四";  
}  
var obj = new Object();  
fn(obj);  
alert(obj.userName+"1111111");

d)检测类型:

**instanceof===》返回布尔值**

如果匹配返回true

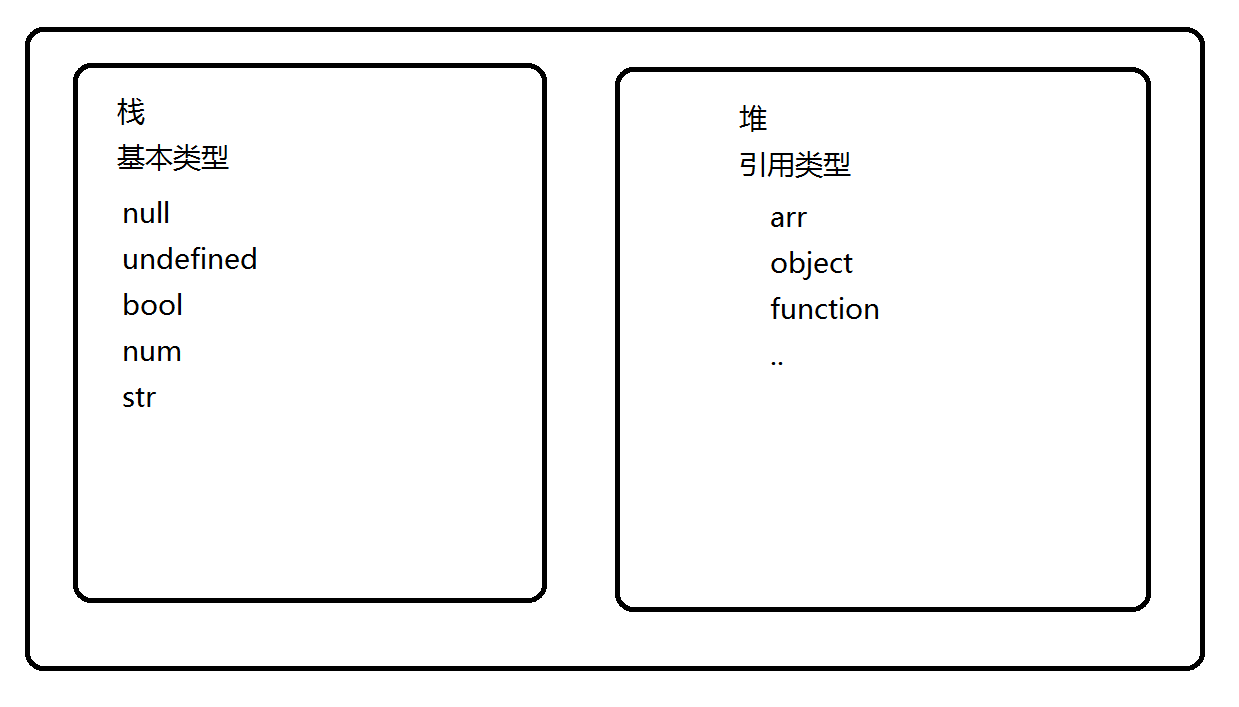
如果不匹配返回false

var arr = [1,2,3];  
alert(typeof arr);  
  
var str = new String();  
alert(typeof str);  
  
var arr = [1,2,3];  
var str = new String();  
  
alert(arr instanceof Array);  
alert(str instanceof String);  
  
if(arr instanceof Array){  
 alert("arr是数组");  
}else{  
 alert("arr不是数组");  
}  
  
if(str instanceof String){  
 alert("是字符串");  
}else{  
 alert("不是字符串");  
}

\*\*\*\*\*\*\*javascript存储\*\*\*\*\*\*\*\*\*\*

1》栈内存（基本类型）

2》堆内存（引用类型）



**一、JS中的作用域**

代码(变量|函数)在某一个地方的作用

1》在全局范围内，声明的变量和函数，其实都是window对象的属性或者方法

<script>  
 var a = 10;  
 alert(a);  
 alert(window.a);  
 //在全局范围内, 变量其实是window对象的一个属性  
 function fn(){  
 alert("fn");  
 }  
 fn();  
 window.fn(); //在全局范围内， 函数其实是window对象的一个方法  
</script>

2》例如函数中声明一个局部变量在全局中打不出来

<script>  
 function fn(){  
 var a = 10;  
 //a是一个局部的a ===》他的作用是在函数呢  
 alert(a);  
 }  
 fn();  
 alert(a);  
  
 // function fn(){  
 // window.a =10; //声明的a 是全局A(window.a)  
 // }  
 // fn();  
 // alert(a);

二、JS中的作用域链

内部有权访问外部的局部变量或者函数

外部不能访问内部的局部变量或者函数

<script>  
 /\* var a = 10;//外部  
 function fn(){  
 alert(a); //内部  
 }  
 fn();\*/  
 /\* var a = 10;  
 function fn(){  
 //222222222222222222  
 var a = 20;  
 return function(){  
 //3333333333333  
 return a;  
 }  
 }  
 alert( fn()() );\*/  
 function fn(){  
 var a = 20;  
 return function(){  
 var a = 10;  
 }  
 }  
 //fn()()  
 //alert( fn() );  
 fn()  
 fn()()  
 alert(a);  
</script>

<script>  
 /\* (function(){  
 var a = 10;  
 })();  
 alert(a); //a报错了\*/  
   
 (function(){  
 var a = b = 5;  
 })();  
 //alert(a); //局部A 其作用是在匿名函数中  
 alert(b); //全局b 其作用是在全局中  
</script>

**1》先找本层(本作用域)===>逐级向外查找**

<script>  
 var a = 10;  
 function fn(){  
 var a = 20;  
 console.log(a);  
 }  
 fn();  
</script>

2》优先级

变量--->函数--->参数---->提升

<script>  
 function fn(){  
 console.log(a);  
 var a = 10;  
 }  
 fn();  
</script>

函数（声明）===》不看执行顺序的

<script>  
 function fn(num){  
 var num = "变量";  
 console.log(num);  
 var num = 10;  
 function num(){  
 }  
 }  
 fn(5);

声明变量的事情

var a ===>局部a

window.a ====>全局a

a ====>属于局部a，也属于全局a

<script>  
 /\*function fn(){  
 var a = 10;  
 function a(){  
 return 111111111;  
 }  
 console.log(a);  
 }  
 fn();\*/  
 /\* function fn(){  
 var a = 10;  
 var a = 20;  
 console.log(a);  
 }  
 fn();\*/  
 function fn(){  
 function a(){  
 return 11111111;  
 }  
 function a(){  
 return 2222222;  
 }  
 console.log(a);  
 }  
 fn();  
</script>

function fn(){

a = 10; ==>属于内部，也属于外部

console.log(a);

var a; //提升后把上面a =10不在是属于内部和外部,只属于内部了

}

fn();

<script>  
 /\* function fn(){  
 var a = 10; // ===>本层找到的  
 window.a = 20; //====>全局a  
 a = 30; //属于内部===》属于全局  
 console.log(a);  
 }  
 fn();\*/  
 /\* function fn(){  
 window.a = 10;  
 console.log(a);  
 var a = 20;  
 }  
 fn();\*/  
 /\*function fn(){  
 var a = 10;  
 a = 30;  
 console.log(window.a);  
 }  
 fn();  
 console.log(a);\*/  
 function fn(){  
 window.a = 10;  
 a = 20; //如果碰到了57有局部a的情况， 55行本身a属于（内部和外部）===》但是现在，只属于内部了  
 console.log(window.a);  
 var a;  
 }  
 fn();  
 console.log(a);  
</script>

**一、JS 垃圾回收机制**

1》标记清除

最常用的一种垃圾回收机制

function fn(){  
 var a = 10;  
 alert(a);  
 }  
 fn();  
 //这里就应该把a销毁掉  
  
function fn(){  
 var a = 10; //标记一下 有a  
 alert(a); //还在 a干了弹出的事情  
 }  
 fn();  
 //没有再用到这个函数中的a了...在内存中，把a销毁了

2》引用计数

最不常用一种垃圾回收机制

function fn(o){  
 var arr = o; // +1  
 var brr = o; // +2  
 //不用了 -1  
 //不用了 -1  
 //到0的时候才能清除  
 }  
 var obj = new Object();  
 fn(obj);

3》手动清除

把不用的变量或者对象或者函数===》（赋值为）= null

function fn(){  
 var obj1 = new Object();  
 obj1.userName = "张三";  
 var obj2 = new Object();  
 obj2.userPwd = "123456";  
 alert(obj1);  
 obj1=null;  
}  
fn();

**一、递归函数**

function fn(num){  
 if(num==1){  
 return 1;  
 }else{  
 return num\*fn(num-1);  
 }  
}  
alert( fn(6) );

自己调用自己

function fn(num){  
 if(num==1){  
 return 1;  
 }else{  
 return num+fn(num-1);  
 }  
}  
alert( fn(5) );

二、闭包

function fn1(){  
 var num = 0;  
 return function(){  
 num++;  
 return num;  
 }  
}  
//alert( fn1()() );  
/\*  
alert( fn1()() ); //1  
alert( fn1()() ); //1  
\*/  
var fn = fn1();  
alert( fn() );//1  
alert( fn() );//2  
alert( fn() );//3

1》什么是闭包

a)函数中还有一个函数

b)for循环中还有一个函数

2》闭包的优点

内部函数有权访问，外部函数的局部变量

3》闭包的缺点

变量会驻留在内存中，导致存储变量不会立刻销毁

案例： 点击鼠标弹出对应的序号值：

Html:

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
</head>  
<body>  
<ul>  
 <li>1111111111</li>  
 <li>2222222222</li>  
 <li>3333333333</li>  
</ul>  
<script src="script.js"></script>  
</body>  
</html>

Js:

var lis = document.getElementsByTagName("li");  
for(var i=0;i<lis.length;i++){  
 lis[i].index = i;  
 lis[i].onclick = function(){  
 alert(this.index);  
 }  
 //i====》执行完毕了  
}

**一、模仿块级作用域**

JS没有块级作用域

语法：(function(){

})()

/\*for(var i=0;i<3;i++){  
}  
alert(i);\*/  
  
/\*(function(){  
 for(var i=0;i<3;i++){  
 }  
})();  
alert(i); //报错 ===>i没有定义\*/  
  
/\*(function(){  
 var jQuery = function(){  
 return "jQuery";  
 }  
 //把局部变为全局  
 window.$=window.jQuery=jQuery;  
})();  
alert( $() );\*/  
  
(function(){  
 var a = b =5;  
 window.a=a;  
})();  
alert(a);

**this指向**

一、关于this指向

var box = document.getElementById("box");  
box.onclick = function(){  
 alert(this); //this代表当前对象  
}  
  
//在全局中,变量其实就是window对象属性  
var str = "123";  
alert(window.str);  
  
//在全局中,函数其实就是window对象的方法  
function fn(){  
 alert(1);  
}  
window.fn();  
  
function fn(){  
 alert(this); //window  
}  
window.fn();

构造函数创建对象: new Object()

字面量创建对象：{}

var obj = new Object(); //构造函数创建对象  
obj.run = function(){ //给obj对象添加了一个run方法  
 return this;  
}  
console.log( obj.run() );  
  
var obj = {  
 run:function(){  
 return this;  
 }  
}  
console.log( obj.run() );

1》如果产生了函数中还有一个函数（闭包）.内部函数中的this指向于window

var obj = {  
 userName:"张三", //obj的userName属性  
 run:function(){ //obj的run方法  
 return this.userName;  
 }  
}  
alert(obj.run());  
  
var userName ="张三"; //window对象添加了一个userName属性值为 ==》张三  
 var obj = {  
 userName:"李四",  
 run:function(){  
 return this.userName;  
 }  
 }  
 alert( obj.run() );

var userName ="张三"; //window对象添加了一个userName属性值为 ==》张三  
var obj = {  
 userName:"李四",  
 run:function(){  
 //this===》obj  
 return function(){  
 //this===>window  
 return this.userName; //闭包。  
 };  
 }  
}  
alert( obj.run()() );

二、改变this指针（指向）

var obj = {  
 run:function(){  
 return this; //this===>obj  
 }  
}  
function fn( f ){  
 alert( f() );  
 alert( arguments[0]() ) //arguments===>对象 this代表了当前对象  
}  
fn( obj.run );

1>赋值

var userName ="张三"; //window对象添加了一个userName属性值为 ==》张三  
var obj = {  
 userName:"李四",  
 run:function(){  
 var That = this; //this==>obj  
 return function(){  
 return That.userName; // this===>window 返回的结果就是张三 。。。返回李四???  
 };  
 }  
}  
alert( obj.run()() );

2>call

var userName = "张三";  
var obj = {  
 userName:"李四",  
 run:function(){  
 return this.userName;//this===> obj;  
 }  
}  
//alert(obj.run()); //这里弹出了李四

//让obj中的run里面的this改变this指向 ， 指向于window  
alert( obj.run.call(window) );

3>apply()

定义：使用一个对象的方法。用一个对象替换当前对象

call和apply的功能是一模一样

var userName = "张三";  
var obj = {  
 userName:"李四",  
 run:function(){  
 //this===>obj  
 return function(){  
 //this===>window  
 return this.userName;  
 }  
 }  
}  
//alert( obj.run()() );  
alert( obj.run().call(obj) ); //让obj.run return的这个函数中的this不再指向于window了， obj\*  
alert( obj.run().apply(obj) );

语法：

方法.call(对象,参数1,参数2,参数3......)

方法.apply(对象,[参数1,参数2,参数....])

/\*  
function fn(a,b){ //fn这个函数有一个功能，实现了2个参数相加，求和  
 return a+b;  
}  
function myFn(){ //我自己的函数，什么功能都没有 ====>我可以去借用一下fn函数的功能  
 return fn.call(window,3,6);  
}  
alert( myFn() );  
\*/  
/\*function parent(num1,num2){  
 return num1+num2;  
}  
function Fn(a,b){  
 this.aaa = parent.call(this,a,b);  
}  
var obj = new Fn(5,6);  
console.log( obj.aaa );\*/  
function parent(num1,num2){  
 return num1+num2;  
}  
function calls(a,b){  
 return parent.call(window,a,b);  
}  
function applys(a,b){  
 return parent.apply(window,[a,b]);  
}  
alert( calls(8,8) );

call参数是一个一个挨着写的

apply是2个参数,第二个参数是一个大数组

/\*  
alert( Math.min(3,2,4,5,6,6) );  
\*/  
/\*  
var arr = [3,2,4,5,6];  
alert( Math.min(arr) );  
\*/  
var arr = [31,99,44,51,66];  
alert( Math.min.apply(window,arr) );  
alert( Math.max.apply(null,arr) ); //借用了Math对象的 max方法的功能===》实现了数组也可以求出最大值

**一、JavaScript函数式编程**

1》高阶函数：函数操作另外一个函数

a)ForEach功能(在适当位置添加元素)

/\*  
var arr = ["A","B",'C'];  
function fn(arr){  
 var str = "";  
 for(var i=0;i<arr.length;i++){  
 str+="2"+arr[i];  
 }  
 return str;  
}  
console.log( fn(arr) );  
\*/  
var arr = ["A","B","C"];  
function forEach(arr,fn){  
 for(var i=0;i<arr.length;i++){  
 fn(arr[i]);  
 }  
}  
forEach(arr,function(value){  
 console.log(value+"0");  
});  
forEach(arr,function(value){  
 console.log("0"+value);  
});

b)filter (过滤功能)

var arr = [2,3,4,5,6,7];  
function filter(arr,fn){  
 var brr = [];  
 for(var i=0;i<arr.length;i++){  
 if(fn(arr[i])){  
 brr.push(arr[i]);  
 }  
 }  
 return brr;  
}  
console.log( filter(arr,function(value){  
 return value>3;  
}) );  
console.log( filter(arr,function(value){  
 return value>5;  
}) );

2》规约函数：通过一个函数调用另一个函数，将数组转换为单一的值

a)Reduce ==》将数组转换为单一的值（相加功能）最后返回一个结果

function Reduce(add,num,array){  
 //add代表了函数（参数相加）  
 //num ==》 2  
 //array==》数组  
 forEach(array,function(value){  
 num = add(num,value);  
 console.log(num);  
 })  
 return num;  
}  
function add(num1,num2){ //参数相加  
 return num1+num2;  
}  
function forEach(arr,fn){ //把数组中的每一个值，循环出来,fn中传入实参  
 for(var i=0;i<arr.length;i++){  
 fn(arr[i]);  
 }  
}  
var arr = [1,2,3,4,5];  
alert( Reduce(add,2,arr) );

3》映射函数：会遍历数组，针对数组的每一个元素,调用指定的操作，然后将操作得出的值存储在另一个数组中，返回新的数组\

a)Map

function Map(fn,array){  
 var brr = [];  
 forEach(array,function(element){  
 brr.push(fn(element))  
 })  
 return brr;  
}  
function forEach(arr,fn){  
 for(var i=0;i<arr.length;i++){  
 fn(arr[i]);  
 }  
}  
console.log( Map(Math.round,[1.1,2.2,3.3]) );

**一、错误调试和处理**

1>try ...catch

语法：try{

有可能导致错误的代码

}catch(错误对象){

错误信息

}**finally{**

**不管代码对还是错，都会执行**

**}**

\*\*\*\*\*\*\*\*不会阻止下面代码的正常运行\*\*\*\*\*\*

function fn(obj){  
 try{  
 obj.userName = "张三";  
 }catch(e){  
 console.log(e.name);  
 console.log(e.message);  
 }finally{  
 console.log("不管上面的代码，对还是错,都会执行")  
 }  
}  
fn(null);  
alert(1);  
/\*  
var err = new Error();  
console.log(err.name);  
console.log(err.message);  
\*/

else:  
function fn(obj){  
 try{  
 obj.usreName = "张三";  
 }catch(error){  
 console.log(error="实参必须是对象,但是不能是null");  
 }  
}  
fn(null);

2>断点：排错

a)设置断点

**一、运动框架**

1》匀速运动

案例：鼠标移入盒子向左运动移除返回

Html：

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
</head>  
<style>  
 \*{margin:0;padding:0}  
 #box{  
 width:300px;  
 height: 300px;  
 background:blue;  
 position: absolute;  
 left:0;  
 top:0;  
 }  
</style>  
<body>  
<div id="box"></div>  
<script src="script.js"></script>  
<script>  
 var box = document.getElementById("box");  
 box.onmouseover = function(){  
 animate(box,300);  
 }  
 box.onmouseout = function(){  
 animate(box,0);  
 }  
</script>  
</body>  
</html>

Js:

var timer = null;  
function animate(dom,target){  
 clearInterval(timer);  
 timer = setInterval(function(){  
 if( dom.offsetLeft<target ){  
 var speed = 10;  
 }else{  
 var speed = -10;  
 }  
 if(target == dom.offsetLeft){  
 //停止定时器  
 clearInterval(timer);  
 }else{  
 dom.style.left = dom.offsetLeft + speed +"px";  
 }  
 },30)  
}

2》缓冲运动

案例：盒子靠近左边或者右边会作减速缓冲：

Html:

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
</head>  
<style>  
 \*{margin:0;padding:0}  
 #box{  
 width:300px;  
 height: 300px;  
 background:blue;  
 position: absolute;  
 left:0;  
 top:0;  
 }  
</style>  
<body>  
<div id="box"></div>  
<script src="script.js"></script>  
<script>  
 var box = document.getElementById("box");  
 box.onmouseover = function(){  
 animate(box,300);  
 };  
 box.onmouseout = function(){  
 animate(box,0);  
 }  
</script>  
</body>  
</html>

Js:

var timer = null;  
function animate(dom,target){  
 clearInterval(timer);  
 timer = setInterval(function(){  
 var speed = (target-dom.offsetLeft)/10;  
 /\*if(speed>0){  
 speed = Math.ceil(speed);  
 }else{  
 speed = Math.floor(speed);  
 }\*/  
 speed = speed>0 ?Math.ceil(speed):Math.floor(speed);/\* 速度大于0就向上取整，速度小于0就向下取整\*/  
 if(target == dom.offsetLeft){  
 //停止定时器  
 clearInterval(timer);  
 }else{  
 dom.style.left = dom.offsetLeft + speed +"px";  
 }  
 },30)  
}

3》多物体运动

案例：多物体点击哪个哪个就变换长度后自动还原

Html：

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
</head>  
<style>  
 \*{margin:0;padding:0}  
 ul li{  
 width:400px;  
 height: 200px;  
 background:blue;  
 margin-top:20px;  
 }  
</style>  
<body>  
<ul>  
 <li></li>  
 <li></li>  
 <li></li>  
</ul>  
<script src="script.js"></script>  
<script>  
var lis =document.getElementsByTagName("li");  
for(var i=0;i<lis.length;i++){  
 lis[i].onmouseover = function(){  
 animate(this,600);  
 }  
 lis[i].onmouseout = function(){  
 animate(this,400);  
 }  
}  
</script>  
</body>  
</html>

Js:

function animate(dom,target){  
 clearInterval(dom.timer);  
 dom.timer = setInterval(function(){  
 var speed = (target-dom.offsetWidth)/10;  
 speed = speed>0 ?Math.ceil(speed):Math.floor(speed);/\*速度大于0就向上取整，速度小于0就向下取整\*/  
 if(target == dom.offsetLeft){  
 //停止定时器  
 clearInterval(dom.timer);  
 }else{  
 dom.style.width = dom.offsetWidth + speed +"px";  
 }  
 },30)  
}

4》任意值运动

案例：

Html:

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
</head>  
<style>  
 \*{margin:0;padding:0}  
 ul li{  
 width:400px;  
 height: 200px;  
 background:blue;  
 margin-top:20px;  
 position: absolute;  
 left:0;  
 top:0;  
 opacity: 0.3;  
 filter:alpha(opacity:30);  
 }  
</style>  
<body>  
<ul>  
 <li></li>  
</ul>  
<script src="script.js"></script>  
<script>  
var lis =document.getElementsByTagName("li");  
for(var i=0;i<lis.length;i++){  
 lis[i].onmouseover = function(){  
 animate(this,700,"width");  
 }  
}  
</script>  
</body>  
</html>

Js:

function animate(dom,target,attr){  
 //dom ===>DOM节点  
 //target ===》目标值  
 //attr ===》属性名称  
 clearInterval(dom.timer);  
 dom.timer = setInterval(function(){  
 if(attr=="opacity"){  
 var objAttr= parseFloat(getAttr(dom,attr))\*100;  
 }else{  
 var objAttr = parseInt(getAttr(dom,attr));  
 }  
 var speed = (target-objAttr)/10;  
 speed = speed>0 ?Math.ceil(speed):Math.floor(speed);  
 if(target == objAttr){  
 //停止定时器  
 clearInterval(dom.timer);  
 }else{  
 if(attr=="opacity"){  
 dom.style.filter = "alpha(opacity:"+objAttr + speed+")";  
 dom.style[attr] = (objAttr + speed)/100;  
 }else{  
 dom.style[attr] = objAttr + speed +"px";  
 }  
 }  
 },30)  
}  
function getAttr(dom,attr){  
 if(dom.currentStyle){  
 return dom.currentStyle[attr];  
 }else{  
 return getComputedStyle(dom,null)[attr];  
 }  
}

5》链式运动

案例：实现物体执行回到原点周期运动

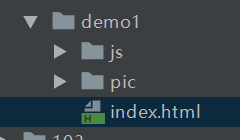
Html:

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
</head>  
<style>  
 \*{margin:0;padding:0}  
 #box{  
 width:300px;  
 height: 300px;  
 background:blue;  
 position: absolute;  
 left:0;  
 top:0;  
 opacity: 0.3;  
 filter:alpha(opacity:30);  
 }  
</style>  
<body>  
<div id="box"></div>  
<script src="script.js"></script>  
<script>  
var box =document.getElementById("box");  
box.onmouseover = function(){  
 animate(box,700,"left",function(){  
 animate(box,500,"top",function(){  
 animate(box,0,"left",function(){  
 animate(box,0,"top")  
 })  
 });  
 });  
}  
</script>  
</body>  
</html>

Js:

function animate(dom,target,attr,fn){  
 //dom ===>DOM节点  
 //target ===》目标值  
 //attr ===》属性名称  
 //fn ===》回调函数  
 clearInterval(dom.timer);  
 dom.timer = setInterval(function(){  
 if(attr=="opacity"){  
 var objAttr= parseFloat(getAttr(dom,attr))\*100;  
 }else{  
 var objAttr = parseInt(getAttr(dom,attr));  
 }  
 var speed = (target-objAttr)/10;  
 speed = speed>0 ?Math.ceil(speed):Math.floor(speed);  
 if(target == objAttr){  
 //停止定时器  
 clearInterval(dom.timer);  
 if(fn)fn();  
 }else{  
 if(attr=="opacity"){  
 dom.style.filter = "alpha(opacity:"+objAttr + speed+")";  
 dom.style[attr] = (objAttr + speed)/100;  
 }else{  
 dom.style[attr] = objAttr + speed +"px";  
 }  
 }  
 },30)  
}  
function getAttr(dom,attr){  
 if(dom.currentStyle){  
 return dom.currentStyle[attr];  
 }else{  
 return getComputedStyle(dom,null)[attr];  
 }  
}

6》同时运动

案例购物车：

Html:  
<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
</head>  
<style>  
 \*{margin:0;padding:0}  
 ul{  
 list-style: none;  
 width:970px;  
 height:270px;  
 position: relative;  
 }  
 #main{  
 width:970px;  
 height:270px;  
 margin:150px auto;  
 position: relative;  
 overflow: hidden;  
 }  
 ul li{  
 width:0px;  
 height: 0px;  
 position: absolute;  
 left:485px;  
 top:135px;  
 background:#000;  
 }  
 ul li img{  
 width:100%;  
 height:100%;  
 opacity: 0.5;  
 }  
 ul li.li1{  
 width:640px;  
 height:270px;  
 left:165px;  
 top:0px;  
 z-index: 10;  
 }  
 ul li.li1 img{  
 opacity: 1;  
 }  
 ul li.li2{  
 width:530px;  
 height:224px;  
 top:23px;  
 z-index: 9;  
 left:440px;  
 }  
 ul li.li3{  
 width:530px;  
 height:100px;  
 top:85px;  
 right:0px;  
 z-index: 8;  
 }  
 ul li.li9{  
 width:530px;  
 height:100px;  
 top:85px;  
 left:0px;  
 z-index: 8;  
 }  
 ul li.li10{  
 z-index: 9;  
 width:530px;  
 height:224px;  
 top:23px;  
 left:0px;  
 }  
 #btn div{  
 position: absolute;  
 font-size:50px;  
 color:#fff;  
 width:50px;  
 height:50px;  
 top:110px;  
 z-index: 999;  
 text-align: center;  
 }  
  
 #left{  
 left:0px;  
 }  
 #right{  
 right:0px;  
 }  
</style>  
<body>  
  
<div id="main">  
  
 <ul>  
 <li class="li1"><img src="pic/01.jpg" alt=""></li>  
 <li class="li2"><img src="pic/02.jpg" alt=""></li>  
 <li class="li3"><img src="pic/03.jpg" alt=""></li>  
 <li><img src="pic/04.jpg" alt=""></li>  
 <li><img src="pic/05.jpg" alt=""></li>  
 <li><img src="pic/06.jpg" alt=""></li>  
 <li><img src="pic/07.jpg" alt=""></li>  
 <li><img src="pic/08.jpg" alt=""></li>  
 <li class="li9"><img src="pic/09.jpg" alt=""></li>  
 <li class="li10"><img src="pic/10.jpg" alt=""></li>  
 </ul>  
  
 <div id="btn">  
 <div id="left"> < </div>  
 <div id="right"> > </div>  
 </div>  
  
</div>  
  
<script src="js/animate.js"></script>  
<script src="js/script.js"></script>  
</body>  
</html>

Js1:

function animate(dom,obj,fn){  
 clearInterval(dom.timer);  
 dom.timer = setInterval(function(){  
 var bool = true;  
 for(var k in obj) {  
 if(k=="opacity"){  
 var objAttr= parseFloat(getAttr(dom,k))\*100;  
 }else{  
 var objAttr = parseInt(getAttr(dom,k));  
 }  
 var speed = (obj[k]-objAttr)/10;  
 speed = speed>0 ?Math.ceil(speed):Math.floor(speed);  
 if(objAttr!=obj[k]){  
 bool=false;  
 }  
 if(k=="opacity"){  
 dom.style.filter = "alpha(opacity:"+objAttr + speed+")";  
 dom.style[k] = (objAttr + speed)/100;  
 }else{  
 dom.style[k] = objAttr + speed +"px";  
 }  
 }  
 if(bool){  
 clearInterval(dom.timer);  
 if(fn){  
 fn.call(dom);  
 }  
 }  
 },30)  
}  
function getAttr(dom,attr){  
 if(dom.currentStyle){  
 return dom.currentStyle[attr];  
 }else{  
 return getComputedStyle(dom,null)[attr];  
 }  
}

js:2

var btn = document.getElementById("btn");  
var oDiv = btn.getElementsByTagName("div");  
var date = new Date(); //只能在刷新页面的时候，执行了一次  
for(var i=0;i<oDiv.length;i++){  
 oDiv[i].index = i;  
 oDiv[i].onclick = function(){  
 if(new Date()-date>1200){  
 fn(this.index);  
 date=new Date();  
 }  
 }  
}  
var lis = document.getElementsByTagName("li");  
function fn(idx){  
 var arrW=[],arrH=[],arrL=[],arrT=[],arrZ=[],arrO=[];  
 for(var i=0;i<lis.length;i++){  
 arrW[i] = parseInt(getAttr(lis[i],"width"));  
 arrH[i] = parseInt(getAttr(lis[i],"height"));  
 arrL[i] = parseInt(getAttr(lis[i],"left"));  
 arrT[i] = parseInt(getAttr(lis[i],"top"));  
 arrZ[i] = parseInt(getAttr(lis[i],"zIndex"));  
 arrO[i] = parseFloat(getAttr(lis[i].getElementsByTagName("img")[0],"opacity"));  
 }  
 for(var i=0;i<lis.length;i++){  
 if(idx==1){  
 //右侧按钮  
 var index = i-1;  
 if(index<0){  
 index=9;  
 }  
 }else{  
 //左侧  
 var index = i+1;  
 if(index>lis.length-1){  
 index=0;  
 }  
 }  
 lis[i].style.zIndex = arrZ[index];  
 animate(lis[i].getElementsByTagName("img")[0],{  
 "opacity":arrO[index]\*100  
 })  
 animate(lis[i],{  
 "width":arrW[index],  
 "height":arrH[index],  
 "left":arrL[index],  
 "top":arrT[index]  
 })  
 }  
}  
function getAttr(dom,attr){  
 if(dom.currentStyle){  
 return dom.currentStyle[attr];  
 }else{  
 return getComputedStyle(dom,null)[attr];  
 }  
}

**一、JS中的对象**

1》理解对象

<script>  
 //document.write(box); //节点对象 ===》DOM对象  
 //box.nodeName  
 var arr = new Array();  
 var str = new String();  
 var date = new Date();  
 var reg = new RegExp();  
 var img = new Image();  
 var fn = new Function();  
 var obj = new Object();  
 var num = new Number();  
 var op = new Option();  
 console.log(typeof window);  
</script>

把对象比喻成一个人，

人有什么？性别，年龄，姓名....

人能干什么？打游戏，打代码，做饭....

<script>  
 var arr = new Array();  
 arr.length  
 arr.join()  
 arr.push()  
 arr.slice()  
 出生===》张三 : new Array(); 继承父亲的遗产  
 性别姓名出生日期打代码玩游戏  
</script>

a)继承性

<script>  
 var obj1 = { //obj就比喻成一个人了  
 "userName":"张三"  
 };  
 var obj2 = { //obj就比喻成一个人了  
 "userName":"李四"  
 };  
 document.write( obj2.userName );  
 /\* var obj3 = {  
 "chifan":"馒头"  
 }  
 var obj4=obj3;  
 obj4.chifan = "面条";  
 document.write(obj3.chifan+"===>"+obj4.chifan);  
 \*/  
 /\*var obj5= {  
 "userName":"张三",  
 "aa bb":"AAAAAAAAAAAA",  
 "get-id":"iiiiiiiiiiiiiiii",  
 0:123,  
 arr:[1,2,3],  
 obj:{  
 userName:"李四"  
 }  
 };  
 document.write( obj5.obj.userName )

document.write( obj5["aa bb"])  
document.write( obj5[0])  
document.write( obj5.arr)

\*/  
</script>

b)引用类型

2》对象用来干什么

1》创建对象

a)字面量形式 : {}

<script>  
 // var obj = {  
 // userName:"张三",  
 // "get id":"box"  
 // };  
 // obj.age = 18;  
 // document.write(obj.userName);  
 // document.write(obj.age);  
 // document.write(obj["get id"]);  
 // var oLi = {  
 // 0:"li1",  
 // 1:"li2",  
 // 2:"li3"  
 // };  
 // // document.write(oLi.0);  
 // document.write(oLi[0]);  
 var lis = document.getElementsByTagName("li");  
 console.log(lis);  
</script>

b)构造函数形式：new XXXX

<script>  
 /\* console.log(Array); ===>函数  
 var arr = new Array();\*/  
 /\*function fn(){  
 }  
 var str = new fn();\*/  
 /\* console.log(Object)\*/  
 var obj = Object.create(Array.prototype); //创建了一个新对象， ===》要了一下数组对象的属性和方法  
 console.log(obj.push());  
</script>

c)Object.create从原型(prototype)创建对象

2》设置对象属性和方法

方法就是:function(){}

<script>  
/\* var obj = {  
 userName:"张三",  
 userAge:18,  
 run:function(){  
 return "我是run方法";  
 }  
 };  
 document.write( obj.run() );\*/  
 var obj = {  
 userName:"张三",  
 userAge:18,  
 run:function(){  
 return this.userName;  
 }  
 };  
 document.write( obj.run() );  
</script>

3》查找对象属性和方法

对象.属性名称|方法名称

对象[属性名称|属性方法]

4》删除对象属性和方法

语法：

delete 对象.属性|方法名称

<script>  
 var obj = {  
 userName:"张三",  
 run:function(){  
 return 111111111111111111111111;  
 }  
 };  
 document.write(obj.run());  
 delete obj.run;  
 document.write(obj.run());  
 /\* document.write(obj.userName);  
 delete obj.userName;  
 document.write(obj.userName);\*/  
</script>

5》检测对象

检测对象的属性是否存在

<script>  
 var obj = {  
 name:"张三" //对象本身的属性  
 };  
 /\*\_\_defineGetter\_\_: \_\_defineGetter\_\_()  
 \_\_defineSetter\_\_: \_\_defineSetter\_\_()  
 \_\_lookupGetter\_\_: \_\_lookupGetter\_\_()  
 \_\_lookupSetter\_\_: \_\_lookupSetter\_\_()  
 constructor: Object()  
 hasOwnProperty: hasOwnProperty()  
 isPrototypeOf: isPrototypeOf()  
 propertyIsEnumerable: propertyIsEnumerable()  
 toLocaleString: toLocaleString()  
 toString: toString()  
 valueOf: valueOf()  
 get \_\_proto\_\_: get \_\_proto\_\_()  
 set \_\_proto\_\_: set \_\_proto\_\_()\*/  
 //对象的原型身上的属性或者方法  
</script>

1>in

语法：属性 in 对象

返回：布尔值

2>hasOwnProperty

语法：对象.hasOwnProperty(属性名称)

返回：布尔值

3>propertyIsEnumerable

语法：对象. propertyIsEnumerable (属性名称)

返回：布尔值

<script>  
 var obj = {  
 name:"张三",  
 age:18  
  
 };  
 /\*console.log( "name" in obj ); //对象身上的  
 console.log( "name123" in obj );//对象没有的  
 console.log("toString" in obj);//对象原型身上的属性 \*/  
 /\*console.log( obj.hasOwnProperty("name") ); // true  
 console.log( obj.hasOwnProperty("name123") );//false  
 console.log( obj.hasOwnProperty("toString") );\*/ //false  
 /\*console.log( obj.propertyIsEnumerable("name") ); //true  
 console.log( obj.propertyIsEnumerable("name123") ); //false  
 console.log( obj.propertyIsEnumerable("toString") ); //false\*/  
 Object.defineProperty(obj,"name",{  
 enumerable:false  
 });  
</script>

\*\*\*\*\*hasOwnProperty和propertyIsEnumerable区别\*\*\*\*\*

hasOwnProperty不可枚举也返回true

propertyIsEnumerable不可枚举返回false

6》枚举对象

对象的属性枚举

for in 遍历出来的是可枚举的

for in 遍历不出来是不可枚举的

<script>  
 var obj = {  
 name:"张三",  
 age:18  
 };  
 //name age 是对象身上添加的  
 //toString constructor.....都是在对象的原型身上  
 Object.defineProperty(obj,"name",{  
 enumerable:false  
 });  
 for(var k in obj){  
 console.log(k);  
 };  
 console.log(obj);  
</script>

Object.defineProperty(对象,属性名称,{

**enumerable**:**false**

}) ===>让对象的属性不可枚举

7》对象的属性只读和只写

get ===》读

set ===》写

<script>  
 /\*var arr = [1,2,3];  
 Array.prototype.length = 10;  
 console.log(arr.\_\_proto\_\_);  
 console.log(arr.length);\*/  
 var obj = {  
 name:"张三",  
 get age(){  
 return this.name;  
 },  
 set age(val){  
 this.name = val;  
 }  
 };  
 obj.age = "李四"; //写  
 console.log(obj.age); //读  
</script>

**一、序列化对象**

<script>  
 var obj ={  
 name:"张三",  
 age:18,  
 run:function(){  
 return "run方法";  
 }  
 };  
 console.log(typeof obj);  
 var str = JSON.stringify(obj);  
 console.log(typeof str);  
 var o = JSON.parse(str);  
 console.log(o);  
 console.log(typeof o);  
</script>

可以把对象转换为字符串：JSON.stringify();

可以把字符串转换为对象：JSON.parse();

**一、对象模式**

1》工厂模式

Js:

//现实中的工厂===>生产一瓶西瓜汁  
//1》西瓜 ====>new Object();  
//2》榨汁 ====>对象的属性  
//3》装瓶,贴商标.... ====>对象的方法  
//4》大货车拉出厂子卖  
function createObject(){  
 var obj = new Object();  
 obj.name = "西瓜汁"; //属性  
 obj.run = function(){//方法  
 return "贴了商标了";  
 };  
 return obj;//出厂==》卖  
}  
var o1 = new createObject();  
var o2 = new createObject();  
/\*console.log(o1.name);  
console.log(o2.name);\*/  
/\*console.log(o1.run == o2.run);\*/

a)现实中的工厂差不多

function createObject(){  
 var obj = new Object();  
 obj.name = "你好js工厂模式";  
 obj.run = function(){  
 return "赶紧出厂，顾客需要";  
 };  
 return obj;  
}  
var obj1 = new createObject();  
var obj2 = new createObject();  
console.log(Array);

b)在函数内部创建了一个新的空对象，最后返回这个对象

function CreateFn(){ //首字母要大写  
 this.str = "构造函数";  
 this.run = function(){  
 return "js构造函数方法";  
 }  
}  
//1>其实构造函数就是普通函数，只不过首字母需要大写了，为了让别人知道，这个函数是一个构造函数  
//2>函数内部写变量不在是var str ===> this.str;  
var obj1= new CreateFn();  
var obj2= new CreateFn();  
/\*console.log(obj1.str);  
console.log(obj2.str);\*/  
console.log(obj1.run == obj2.run);

c)缺点：虽然new的是共同的一个函数,但是其中的方法做对比，返回false,也就意思说，new的函数，中的方法，不是共享（不是同一个地址）.

2》构造函数模式

a)和工厂模式区别：

构造函数中，不需要在内部创建对象（给这个对象添加属性，方法，也不用return 对象）

b)如果要写构造函数模式，函数的首字母需要大写

new Array()

new String()

new Date()

new RegExp();

c)缺点：虽然new的是共同的一个函数,但是其中的方法做对比，返回false,也就意思说，new的函数，中的方法，不是共享（不是同一个地址）.

3》原型模式

1>每当创建一个函数，都有一个prototype(原型)的属性

2>原型（prototype）这个属性的指针,指向于一个对象，而这个对象的用途可以由特定类型的所有实例“共享”的方法和属性.

3>原型（prototype）是共享所有的属性和方法,也就是,如果new了2个函数（实例化），他们的方法是共享的，做对比，返回true，共同使用一个地址

function fn(){  
 fn.prototype.userName = "张三";  
 fn.prototype.arr = [1,2,3];  
 //把fn的原型的属性(userName)共享给对象了  
 fn.prototype.run = function(){  
 return "这是一个共享的方法";  
 }  
}  
var obj1 = new fn();  
var obj2 = new fn();  
/\*  
console.log(obj1.userName);  
console.log(obj2.userName);\*/  
//console.log(obj1.run == obj2.run);  
/\*  
obj1.userName = "李四";  
console.log(obj1.userName);  
console.log(obj2.userName);  
\*/  
obj1.arr[1] = "你好数组";  
console.log(obj1.arr);  
console.log(obj2.arr);

a)缺点 ：因为是共享的方法，公用一个地址,如果后期一个修改，所有对象的方法就都修改了

4》组合使用构造器和原型模式(推荐按照需要的共享和不共享)

a>构造函数:定义实例属性

function CreateFn(){  
 this.userName = "张三";  
 this.age = 18;  
 this.arr = [1,2,3];  
  
};  
CreateFn.prototype.run = function(){  
 return "我是共享的";  
};  
var obj1 = new CreateFn();  
var obj2 = new CreateFn();  
console.log(obj1.run == obj2.run);  
obj1.arr[1] = "李四";  
console.log(obj1.arr);  
console.log(obj2.arr);

b>原型:定义和共享属性

优点：每一个实例都会有自己的一份实例属性，但又同时共享着方法

function CreateFn(name,age){  
 this.userName = name;  
 this.age = age;  
 this.arr = [1,2,3];  
};  
CreateFn.prototype.run = function(){  
 return "我是共享的";  
};  
var obj1 = new CreateFn("张三",18);  
var obj2 = new CreateFn("李四",20);  
console.log(obj1.arr[2]);  
console.log(obj2.userName);

5》动态原型模式

是将所有的信息（属性和方法）都封装在构造函数中，而通过构造函数来初始化原型

解决：第一个对象实例化的时候就初始化原型，后面的就不需要初始化

function CreateFn(){  
 this.userName = "张三";  
 if( typeof this.run !="function" ){  
 console.log(1);  
 alert(1);  
 CreateFn.prototype.run = function(){  
 return "共享的run方法";  
 }  
 }  
}  
var obj1 = new CreateFn();  
var obj2 = new CreateFn();

**一、继承**

1》原型链

用法：把实例的父类给子函数的原型

Child.prototype=new Parent(); //原型链===》核心//原型链===》核心语句 1》不能传递参数2》arr是引用属性，一个改变，互相影响

缺点：

1》因为修改了obj1.arr后obj2.arr也变了，因为来自原型对象的引用属性里所有实例共享的

2》创建子类实例时，无法向父类构造函数传递参数

function Parent(){//父  
 this.userName = "父函数";  
 this.arr = [1,2,3];  
}  
function Child(age){//子  
 this.age = age;  
}  
Child.prototype=new Parent(); //原型链===》核心  
var obj1 = new Child(18);  
var obj2 = new Child();  
//console.log(obj1.userName);  
obj1.arr[0] = "张三";  
console.log(obj1.arr);  
console.log(obj2.arr);

2》借用构造函数（无共享功能）

Parent.call(this,name,arr); //借用构造函数 ====》 核心代码 //借用构造函数===>核心语句 1》不能复用

核心：借父类的构造函数来增强子类实例，就是说，相当于复制了一份父类的属性或者方法给子类了

优点：

a)解决了子类实例共享父类引用属性的问题

b)创建子类实例时，可以向父类构造函数传递参数

function Parent(name,arr){  
 this.userName = name;  
 this.arr = arr;  
 this.run = function(){  
 return 1111;  
 }  
}  
function Child(name,arr){  
 this.age = 18;  
 Parent.call(this,name,arr); //借用构造函数 ====》 核心代码  
}  
var obj1 = new Child("张三",[4,5,6]);  
var obj2 = new Child("张三",[4,5,6]);  
/\*console.log(obj1.userName);  
console.log(obj2.userName);\*/  
obj1.arr[0] = "你好构造函数";  
console.log( obj1.arr );  
console.log( obj2.arr );

缺点：

a)无法实现复用,每一个子类实例都有一个新的run函数，如果实例的对象多了，内存消耗过大

3》继承方式:

a)组合继承（最常用的方式）

优点：

1》不存在引用属性共享的问题

2》可传递参数

3》方法可复用

缺点：（小瑕疵）

子类原型上有一份多余的父类实例的属性

function Parent(name,arr){  
 this.userName = name;  
 this.arr = arr;  
}  
Parent.prototype.run = function(){  
 return "我是run方法";  
};  
function Child(name,arr){  
 Parent.call(this,name,arr); //借用构造函数===>核心语句 1》不能复用  
}  
Child.prototype=new Parent(); //原型链===》核心语句 1》不能传递参数2》arr是引用属性，一个改变，互相影响  
var obj1 = new Child("张三",[1,2,3]);  
var obj2 = new Child("李四",[1,2,3]);

b)原型式继承

核心：用一个函数，生出来一个新的对象

优点：从已有对象繁衍出新的对象，不需要创建自定义类型

缺点：

1》原型的引用属性会互相影响(公用一个地址)

2》无法实现代码复用，属性是后添加的，都没用到函数封装

/\*  
function fn(obj){ //用来生新对象的  
 function F(){}//构造函数  
 F.prototype=obj; //F新的对象  
 return new F();  
}  
function Child(){ //构造函数  
 this.val = 1;  
 this.age = 18;  
 this.arr = [1,2,3];  
}  
var sub = new Child(); //实例化child函数了  
var obj1 = fn(sub);  
var obj2 = fn(sub);  
obj1.arr[1] = "张三";  
console.log(obj1.arr);  
console.log(obj2.arr);  
\*/  
function fn(obj){ //用来生新对象的  
 function F(){}//构造函数  
 F.prototype=obj; //F新的对象  
 return new F();  
}  
var obj = {  
 name:"张三",  
 age:187  
};  
obj1.aa = "李四";  
obj2.bb = "你好js";  
var obj1 = fn(obj);  
var obj2 = fn(obj);  
alert(obj1.name);

c)寄生式继承

核心：创建新对象===》增强对象（添加属性或者方法）===》返回对象

function fn(obj){  
 var F = function(){};  
 F.prototype=obj;  
 return new F();  
}  
//原型式  
function getSub(obj){  
//寄生核心  
 var clone = fn(obj);  
 //新对象  
 clone.attr1 = "class";  
 clone.attr2 = "id";  
 clone.brr = [1,2,3,];  
 //增强  
 return clone;  
}  
var o = {  
 name:"张三",  
 age:18,  
 arr:[1,2,3]  
  
};  
var obj1 = new getSub(o);  
var obj2 = new getSub(o);  
obj1.brr[0] = "张三";  
console.log(obj1.brr);  
console.log(obj2.brr);

d)寄生组合式继承

优点：修正组合继承的缺点，只使用一次构造函数

缺点：写法繁琐

function fn(obj){  
 var F = function(){};  
 F.prototype=obj;  
 return new F();  
}  
function Sub(){  
 this.str = "张三";  
 //基本属性  
 this.arr = [1,2,3];  
 //引用属性  
}  
Sub.prototype.arr = [1,2,3];  
Sub.prototype.run = function(){  
 return "共享的run方法";  
};  
function SubType(){  
 Sub.call(this); //核心代码 ===》借用sub构造函数  
}  
var obj1 = fn(Sub.prototype);//核心===》传递原型  
obj1.constructor = SubType; //构造函数指向  
Sub.prototype = obj1; //实例对象赋值给sub原型  
var obj2 = new SubType();  
var obj3 = new SubType();  
obj2.arr[0] = "张三";  
console.log(obj2.arr);  
console.log(obj3.arr);  
console.log(obj2.run == obj3.run);

案例：tab利用面对过程的方式实现效果的切换

Html：

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
</head>  
<style>  
 \*{margin:0;padding:0}  
 ul{  
 list-style: none;noen}  
 #uls{width:300px; height:50px;}  
 #uls li{  
 float:left;  
 width:150px;  
 height:50px;  
 line-height: 50px;  
 text-align: center;  
 }  
 .active{  
 background: #e6e6e6;  
 }  
 div ul{  
 width:300px;  
 height:300px;  
 text-align: center;  
 line-height: 50px;  
 }  
 div ul li{  
 width:300px;  
 height:50px;  
 float:left;  
 }  
 .box2{display: none}  
</style>  
<body>  
  
<ul id="uls">  
 <li class="active">信息公告</li>  
 <li>服务公告</li>  
</ul>  
<div class="box1">  
 <ul>  
 <li>20万图书每满100减40</li>  
 <li>夏季防蚊祛痱3件5折</li>  
 <li>当当优品庆六一 0利 4轮秒杀</li>  
 <li>童装童鞋全场满200减100</li>  
 <li>六一礼品季童书1.9-4.9折封</li>  
 </ul>  
</div>  
<div class="box2">  
 <ul>  
 <li>积分支付上线了，直接当钱用</li>  
 <li> 当当阅读器 海量电子书随时读</li>  
 <li>端午撞六一，礼品卡钜惠</li>  
 <li>自营商品满39包邮啦！</li>  
 <li>关于假冒客服诈骗的声明</li>  
 </ul>  
</div>  
<hr>  
<script src="tab.js"></script>  
</body>  
</html>

js：

//1》基本的排版+效果实现(面向过程)  
//2》把面向过程的代码，修改成面向对象  
var uls = document.getElementById("uls"),  
 lis = uls.getElementsByTagName("li"),  
 oDiv = document.getElementsByTagName("div");  
for(var i=0;i<lis.length;i++){  
 lis[i].index = i;  
 lis[i].onmouseover = function(){  
 for(var j=0;j<oDiv.length;j++){  
 oDiv[j].style.display = "none";  
 lis[j].className="";  
 }  
 oDiv[this.index].style.display = "block";  
 lis[this.index].className="active";  
 }  
}

案例:利用面对对象的方法实现tab切换效果，常用于多次调用同样样式和效果的功能

Html：  
<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
</head>  
<style>  
 \*{margin:0;padding:0}  
 ul{  
 list-style: none;noen}  
 #uls{width:300px; height:50px;}  
 #uls li{  
 float:left;  
 width:150px;  
 height:50px;  
 line-height: 50px;  
 text-align: center;  
 }  
 .active{  
 background: #e6e6e6;  
 }  
 div ul{  
 width:300px;  
 height:300px;  
 text-align: center;  
 line-height: 50px;  
 }  
 div ul li{  
 width:300px;  
 height:50px;  
 float:left;  
 }  
 .box2{display: none}  
</style>  
<body>  
<ul id="uls">  
 <li class="active">信息公告</li>  
 <li>服务公告</li>  
</ul>  
  
<div id="box">  
 <div class="box1">  
 <ul>  
 <li>20万图书每满100减40</li>  
 <li>夏季防蚊祛痱3件5折</li>  
 <li>当当优品庆六一 0利 4轮秒杀</li>  
 <li>童装童鞋全场满200减100</li>  
 <li>六一礼品季童书1.9-4.9折封</li>  
 </ul>  
 </div>  
 <div class="box2">  
 <ul>  
 <li>积分支付上线了，直接当钱用</li>  
 <li> 当当阅读器 海量电子书随时读</li>  
 <li>端午撞六一，礼品卡钜惠</li>  
 <li>自营商品满39包邮啦！</li>  
 <li>关于假冒客服诈骗的声明</li>  
 </ul>  
 </div>  
</div>  
<hr>  
<div id="btn">  
 <button>案例分析</button>  
 <button>联系我们</button>  
</div>  
<div id="item">  
 <ul>  
 <li>案例分析1</li>  
 <li>案例分析2</li>  
 <li>案例分析3</li>  
 </ul>  
 <ul style="display: none">  
 <li>联系我们1</li>  
 <li>联系我们2</li>  
 <li>联系我们3</li>  
 </ul>  
</div>  
<script src="tab.js"></script>  
</body>  
</html>

Js：

function Tab(options){  
 this.id = document.getElementById(options.id);  
 this.idChild = this.id.getElementsByTagName(options.idChild);  
 this.passiveId = document.getElementById(options.passiveId);  
 this.passiveChild = this.passiveId.getElementsByTagName(options.passiveChild);  
 this.events = options.events;  
 this.init();  
}  
Tab.prototype = {  
 init:function(){  
 //效果的开始  
 var That = this;  
 for(var i=0;i<this.idChild.length;i++){  
 this.idChild[i].index = i;  
 this.idChild[i][this.events] = function(){  
 That.change(this.index);  
 }  
 }  
 },  
 change:function(i){  
 for(var j=0;j<this.passiveChild.length;j++){  
 this.passiveChild[j].style.display = "none";  
 this.idChild[j].className="";  
 }  
 this.passiveChild[i].style.display = "block";  
 this.idChild[i].className="active";  
 }  
};  
new Tab({  
 id:"uls",  
 idChild:"li",  
 //谁加入的事件  
 passiveId:"box",  
 passiveChild:"div",  
 //元素显示  
 events:"onmouseover"  
 //事件名称  
});  
new Tab({  
 id:"btn",  
 idChild:"button",  
 //谁加入的事件  
 passiveId:"item",  
 passiveChild:"ul",  
 //元素显示  
 events:"onclick"  
 //事件名称  
});  
/\*  
 \*1》先写入了构造函数  
 \*2》构造函数中传入实参（对象）====》按照节点不同，展示不同，事件不同，按需传入的参数  
 \*3》把面向过程代码放入到init方法中， 方法的分布，一般都是按照功能来细致划分的  
 \*4》需要修改了有：节点名称（this.）,注意this指向问题修改，this.index的修改，因为this现在代表不是节点而是实例化对象，所有index是不存在的，要做为参数传递过来  
 \*/

案例：面对过程的scroll效果

Heml:

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
</head>  
<style>  
 \*{margin:0;padding:0}  
 ul{list-style: none}  
 #box1{  
 width:300px;  
 height:100px;  
 border:1px solid #ccc;  
 }  
 #box2{  
 width:100px;  
 height:300px;  
 border:1px solid #ccc;  
 }  
 ul li{  
 float:left;  
 width:100px;  
 height:100px;  
 }  
  
</style>  
<body>  
  
  
<div id="box1">  
 <ul>  
 <li>1</li>  
 <li>2</li>  
 <li>3</li>  
 <li>4</li>  
 <li>5</li>  
 </ul>  
</div>  
<br>  
<br>  
<hr>  
<div id="box2">  
 <ul>  
 <li>1</li>  
 <li>2</li>  
 <li>3</li>  
 <li>4</li>  
 <li>5</li>  
 </ul>  
</div>  
<script src="scroll.js"></script>  
</body>  
</html>

Js：

var box1 = document.getElementById("box1");  
box1.style.overflow = "hidden";  
var ul = box1.getElementsByTagName("ul")[0];  
var li = ul.getElementsByTagName("li");  
var li\_num = li.length;  
var li\_width = parseInt(getAttr());  
ul.style.width = li\_num\*li\_width+"px";  
var index = 0;  
setInterval(function(){  
 index--;  
 ul.style.marginLeft = index+"px";  
 if(Math.abs(index)>=li\_width){  
 ul.appendChild( ul.firstElementChild );  
 ul.style.marginLeft = "0px";  
 index=0;  
 }  
},50);  
  
function getAttr(){  
 if(li[0].currentStyle){  
 return li[0].currentStyle["width"];  
 }else{  
 return getComputedStyle(li[0],null)["width"];  
 }  
}

案例：面对对象的scroll效果

Html：

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
</head>  
<style>  
 \*{margin:0;padding:0}  
 ul{list-style: none}  
 #box1{  
 width:300px;  
 height:100px;  
 border:1px solid #ccc;  
 }  
 #box2{  
 width:100px;  
 height:300px;  
 border:1px solid #ccc;  
 }  
 ul li{  
 float:left;  
 width:100px;  
 height:100px;  
 }  
</style>  
<body>  
<div id="box1">  
 <ul>  
 <li>1</li>  
 <li>2</li>  
 <li>3</li>  
 <li>4</li>  
 <li>5</li>  
 </ul>  
</div>  
<br>  
<br>  
<hr>  
<div id="box2">  
 <ul>  
 <li>1</li>  
 <li>2</li>  
 <li>3</li>  
 <li>4</li>  
 <li>5</li>  
 </ul>  
</div>  
<script src="scroll.js"></script>  
</body>  
</html>

Js:

function Scroll(options){  
 this.el = document.getElementById( options.el );  
 //得到参数  
 this.ul = this.el.getElementsByTagName("ul")[0];  
 //获取ul  
 this.li = this.ul.getElementsByTagName("li");  
 //获取li  
 this.direction = options.direction;  
 //无缝走向  
 this.speed = options.speed;  
 //速度值  
 this.index = 0;  
 this.init();  
}  
Scroll.prototype = {  
 init:function(){  
 //效果的开始  
 this.li\_num = this.li.length;  
 //1>给父元素 设置溢出隐藏  
 this.el.style.overflow = "hidden";  
 if(this.direction=="x"){  
 //宽度  
 this.change("width","marginLeft");  
 }else if(this.direction=="y"){  
 //高度  
 this.change("height","marginTop");  
 }  
 },  
 change:function(attr1,attr2){  
 //定时器  
 var That = this;  
 this.li\_attr = parseInt(this.getAttr(attr1));  
 this.ul.style[attr1] = this.li\_num\*this.li\_attr+"px";  
 setInterval(function(){  
 That.index--;  
 That.ul.style[attr2] = That.index+"px";  
 if(Math.abs(That.index)>=That.li\_attr){  
 That.ul.appendChild( That.ul.firstElementChild );  
 That.ul.style[attr2] = "0px";  
 That.index=0;  
 }  
 },this.speed);  
 },  
 getAttr:function(attr){  
 //获取样式值的  
 if(this.li[0].currentStyle){  
 return this.li[0].currentStyle[attr];  
 }else{  
 return getComputedStyle(this.li[0],null)[attr];  
 }  
 }  
};  
new Scroll({  
 el:"box1",  
 direction:"x",  
 speed:10  
});  
new Scroll({  
 el:"box2",  
 direction:"y",  
 speed:30  
});

**一、JSON**

1》什么是JSON

轻量级交换数据的一种格式

2》JSON中存储的值

a)简单值

数字类型

字符串类型

....

特殊字符,必须加入双引号

b)数组

<script>  
 var obj = {  
 "arr":[1,2,3],  
 o:{  
 data:["a","b"]  
 },  
 "brr":[  
 "a",  
 {  
 userName:"张三"  
 }  
 ]  
};  
 /\*console.log(obj.arr);  
 console.log( obj.o.data[1] );\*/  
 console.log(obj.brr[1].userName);  
</script>

c)对象

<script>  
/\*var obj = {  
 userName:true  
};  
console.log(obj.userName);\*/  
var obj = {  
 "margin-Left":"200px",  
 "marginLeft":"300px",  
 "a b":"有空格了",  
 "if":"这是if"  
};  
/\*console.log(obj."margin-Left");\*/  
console.log( obj["margin-Left"] );  
console.log( obj.marginLeft );  
console.log( obj["a b"] );  
console.log( obj.if );  
</script>

3》序列化

字符串转换为对象:JSON.parse()

对象转换为字符串:JSON.stringify

\*\*\*\*\*\*\*\*只要是在.json文件中\*\*\*\*\*\*

Html:

<script>  
 var obj = {  
 "userName":"张三"  
 }  
 var str = JSON.stringify(obj);  
 console.log( str );  
 var o = JSON.parse(str);  
 console.log(o);  
</script>

Json:

{  
 "userName":"张三",  
 "arr":[  
 {},  
 {},  
 {}  
 ],  
 "obj":{  
 "userName":"张三"  
 }  
}

key==》必须要加入引号

**一、什么是Ajax(异步javascript 和 xml )**

一种创建交互网页应用一门技术

二、Ajax应用场景

实时更新（地图）、表单验证...

三、优缺点

优点:

1>实现局部更新(无刷新状态)

2>减轻服务器端压力

...

缺点：

1>破坏了浏览器的前进和后退功能

2>一个ajax请求多了，页面出现加载慢的情况

3>搜索引擎的支持程度比较弱

4>ajax安全性问题

.....

四、如果要使用ajax必须要有后台环境的支持(服务器端)

安装php环境：appserv

安装完成后，需要在浏览器

localhost:8888[端口] 回车

localhost:8888[端口]===》www目录下了

**一、HTTP请求**

GET：用于获取数据。GET是在URL上传递数据。存储量较少，安全低。

POST：用于上传数据（表单）。POST安全性一般，容量几乎是无限

二、ajax

1》创建ajax对象

IE6以上：new XMLHttpRequest();

IE6：new ActiveXObject("Microsoft.XMLHTTP");

2》链接服务器

ajax对象.open("请求方式[get|post]",url,异步|同步)

参数3》true===>异步

false===>同步

3》发送请求

ajax对象.send()

4》接收返回值

onreadyStatechange事件:当请求被发送到服务器时，我们需要执行一些基于响应的操作.当readyState改变的时候，就会触发这个执行

readyState:请求的状态，返回的状态码(0-4)

0（未初始化）：open还没有调用

1(载入)：已经调用了send()正在发送请求

2（载入完成）：send()方法完成，已经收到了全部的响应内容

3（解析）：正在解析响应内容

4（完成）：响应内容解析完成，可以在客户端用了

status:返回请求结果（码）

200-》成功

404-》未找到

5\*\*开头->服务器错误

responseText:返回内容

案例1：

a.html：

你好我是a页面

Html:

<script>  
 //1》创建AJAX对象  
 var xhr = null;  
 if(window.XMLHttpRequest){  
 xhr = new XMLHttpRequest();  
 }else{  
 xhr = new ActiveXObject("Microsoft.XMLHTTP");  
 }  
 //2》链接服务器  
 xhr.open("GET","a.html",true);  
 //3》发送请求  
 xhr.send(null);  
 //4》接收返回值  
 xhr.onreadystatechange = function(){  
 // alert(xhr.readyState);  
 //alert(xhr.responseText);  
 if(xhr.readyState==4){  
 alert(xhr.responseText);  
 }  
 }  
</script>

案例2：

<script>  
 //1》创建AJAX对象  
 var xhr = null;  
 if(window.XMLHttpRequest){  
 xhr = new XMLHttpRequest();  
 }else{  
 xhr = new ActiveXObject("Microsoft.XMLHTTP");  
 }  
 //2》链接服务器  
 xhr.open("GET","package.json",true);  
 //3》发送请求  
 xhr.send(null);  
 //4》接收返回值  
 xhr.onreadystatechange = function(){  
 if(xhr.readyState==4 && xhr.status==200){  
 var obj = JSON.parse(xhr.responseText);  
 console.log(obj.name+"===>"+obj.age);  
 }  
 }  
</script>

Package.json:

{  
 "name": "张三",  
 "age":18  
}

案例：封装Ajax：

Html:

<script>  
 function ajax(options){  
 var xhr = null;  
 var params = formsParams(options.data);  
 //1》创建对象  
 if(window.XMLHttpRequest){  
 xhr = new XMLHttpRequest();  
 }else{  
 xhr = new ActiveXObject("Mincrosoft.XMLHTTP");  
 }  
 //2>open链接  
 if(options.type=="GET"){  
 xhr.open(options.type,options.url+"?"+params,options.async);  
 xhr.send(null);  
 }else if(options.type=="POST"){  
 xhr.open(options.type,options.url,options.async);  
 xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");  
 xhr.send(params);  
 }  
 xhr.onreadystatechange = function(){  
 if(xhr.readyState==4 && xhr.status==200){  
 options.success(xhr.responseText);  
 }  
 };  
 function formsParams(data){  
 var arr = [];  
 for(var i in data){  
 arr.push( i+"="+data[i] );  
 }  
 return arr.join("&");  
 }  
 }  
 ajax({  
 url:"a.php",//url--->地址  
 type:"POST", //type--->请求方式  
 async:true,//异步==>true 同步==》false  
 data:{name:"张三",age:18},  
 success:function(data){  
 console.log(data);  
 }  
 })  
</script>

A.php:

<?php  
 echo $\_POST["name"]."===>".$\_POST["age"];  
?>

案例：AJAX对象表单验证：

Ajax.js:

function ajax(options){  
 var xhr = null;  
 var params = formsParams(options.data);  
 //1》创建对象  
 if(window.XMLHttpRequest){  
 xhr = new XMLHttpRequest();  
 }else{  
 xhr = new ActiveXObject("Microsoft.XMLHTTP");  
 }  
 //2>open链接  
 if(options.type=="GET"){  
 xhr.open(options.type,options.url+"?"+params,options.async);  
 xhr.send(null);  
 }else if(options.type=="POST"){  
 xhr.open(options.type,options.url,options.async);  
 xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");  
 xhr.send(params);  
 }  
 xhr.onreadystatechange = function(){  
 if(xhr.readyState==4 && xhr.status==200){  
 options.success(xhr.responseText);  
 }  
 };  
 function formsParams(data){  
 var arr = [];  
 for(var i in data){  
 arr.push( i+"="+data[i] );  
 }  
 return arr.join("&");  
 }  
}

html：

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
</head>  
<body>  
用户名：<input type="text" id="t1"><span>\*这里输入账号</span>  
<hr>  
密码：<input type="text" id="t2"><span>\*这里输入密码</span>  
<script src="ajax.js"></script>  
<script>  
 var t1 = document.getElementById("t1");  
 t1.onblur = function(){  
 ajax({  
 url:"login.php",  
 type:"GET",  
 async:true,  
 data:{name:t1.value},  
 success:function(data){  
 if(data==1){  
 t1.nextElementSibling.innerHTML = "该用户名已经有了";  
 }else{  
 t1.nextElementSibling.innerHTML = "恭喜您用户名可以注册";  
 }  
 }  
 })  
 }  
</script>  
</body>  
</html>

Login.php:

<?php  
 //数据中查出来的数据和当前输入的value做比较，如果有提示no，如果没有，OK  
 $uName = $\_GET["name"];  
 $link = mysql\_connect("localhost","root","root");  
 mysql\_select\_db("admin",$link);  
 $sql = "select \* from user where uName='{$uName}'";  
 $result = mysql\_query($sql);  
 $num = mysql\_num\_rows($result);  
 if($num==1){  
 echo 1;  
 }else{  
 echo 0;  
 }  
?>

**一、JSONP（跨域）**

1>同源策略：同一个ip段

Html:

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
</head>  
<body>  
<script src="ajax.js"></script>  
<script>  
 ajax({  
 url:"https://www.baidu.com/s?ie=utf-8&csq=1&pstg=20&mod=2&isbd=1&cqid=8d83f40200002205&istc=500&ver=0Qt8HtMHOuXaje7bo\_TUy39X0LZyZSuVDI\_&chk=592e26eb&isid=F4AB416ED4368962&ie=utf-8&f=8&rsv\_bp=0&rsv\_idx=1&tn=baidu&wd=a&rsv\_pq=9d4324ab00001d7f&rsv\_t=ca8bdmKortQTE5bC%2Fnn9RRIvbeY%2FnsWwx%2FA0rzTeCLSC4ZII6NMZsbmpkqY&rqlang=cn&rsv\_enter=0&rsv\_sug3=2&rsv\_sug1=1&rsv\_slog=btn\_click&inputT=2440&rsv\_sug4=2598&f4s=1&\_ck=948.1.81.72.19.733.30&rsv\_isid=1468\_21115\_22749\_22074&isnop=1&rsv\_stat=-2&rsv\_bp=1",  
 type:"GET",  
 async:true,  
 success:function(data){  
 console.log(data);  
 }  
 })  
</script>  
</body>  
</html>

Ajax.js:

function ajax(options){  
 var xhr = null;  
 var params = formsParams(options.data);  
 //1》创建对象  
 if(window.XMLHttpRequest){  
 xhr = new XMLHttpRequest();  
 }else{  
 xhr = new ActiveXObject("Mincrosoft.XMLHTTP");  
 }  
 //2>open链接  
 if(options.type=="GET"){  
 xhr.open(options.type,options.url+"?"+params,options.async);  
 xhr.send(null);  
 }else if(options.type=="POST"){  
 xhr.open(options.type,options.url,options.async);  
 xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");  
 xhr.send(params);  
 }  
 xhr.onreadystatechange = function(){  
 if(xhr.readyState==4 && xhr.status==200){  
 options.success(xhr.responseText);  
 }  
 };  
 function formsParams(data){  
 var arr = [];  
 for(var i in data){  
 arr.push( i+"="+data[i] );  
 }  
 return arr.join("&");  
 }  
}

2>跨域src属性

案例：使用SRC 调用高德地图

<!doctype html>  
<head>  
 <meta charset="utf-8">  
 <meta http-equiv="X-UA-Compatible" content="chrome=1">  
 <meta name="viewport" content="initial-scale=1.0, user-scalable=no, width=device-width">  
 <style type="text/css">  
 body,html,#container{  
 width:100%;  
 height: 100%;  
 margin: 0px;  
 }  
 </style>  
 <title>快速入门</title>  
</head>  
<body>  
<div id="container" tabindex="0"></div>  
<script type="text/javascript" src="http://webapi.amap.com/maps?v=1.3&key=a2f9f8042ecf820c8abe73a99de2805a"></script>  
<script type="text/javascript">  
 var map = new AMap.Map('container',{  
 resizeEnable: true,  
 zoom: 10,  
 center: [116.480983, 40.0958]  
 });  
</script>  
</body>  
</html>

**一、JS设计模式**

1》单体模式

<script>  
 /\*function Obj(){  
 this.name = "张三";  
 this.run = null;  
 }  
 Obj.prototype.getName = function(){  
 return this.name;  
 };  
 function getRun(){  
 //可以实例化(new)，但是只能实例化一次  
 if(!this.run){  
 this.run = new Obj();  
 }  
 return this.run;  
 }  
 var a = getRun();  
 var b = getRun();  
 console.log(a.name);  
 console.log(b.name);  
 console.log(a===b);\*/  
 //--------------------------------------  
 function Obj(){  
 this.name = "张三";  
 }  
 Obj.prototype.getName = function(){  
 return this.name;  
 };  
 var getRun = (function(){  
 var run = null;  
 return function(){  
 if(!run){  
 run = new Obj();  
 }  
 return run;  
 }  
 })();  
 var a = getRun();  
 var b = getRun();  
 console.log("====");  
 console.log(a.name);  
 console.log(b.name);  
 console.log(a===b);  
  
</script>

理解：单体模式是提供了一种代码组织为一个逻辑单元的手段，这个逻辑单元中的代码可以通过单一变量进行访问。

<script>  
//用一个函数创建div节点，放入到body中， 函数===>专一，创建div  
 var CreateDiv = function(html){  
 this.html = html;  
 this.init();  
 };  
 CreateDiv.prototype.init = function(){  
 var oDiv = document.createElement("div");  
 oDiv.innerHTML = this.html;  
 document.body.appendChild(oDiv);  
 };  
 var proxyMode = (function(){  
 var run;  
 return function(){  
 if(!run){  
 run = new CreateDiv("这是一个单体");  
 }  
 return run;  
 }  
 })();  
 proxyMode();  
</script>

优点：

a)可以划分命名空间

b)使代码阅读性，维护性更好

c)可以实例化(new)，但是只能实例化一次

案例：单体实例化一次

<body>  
 <button id="btn">按钮</button>  
<script>  
 /\*function createDiv(){  
 var oDiv = document.createElement("div");  
 oDiv.innerHTML = "这就是一个格子";  
 oDiv.style.display = "none";  
 document.body.appendChild(oDiv);  
 return oDiv;  
 }  
 document.getElementById("btn").onclick = function(){  
 var w = createDiv();  
 w.style.display = "block";  
 }\*/  
 var createDiv = (function(){  
 var oDiv;  
 return function(){  
 if(!oDiv){  
 oDiv = document.createElement("div");  
 oDiv.innerHTML = "这就是一个格子";  
 oDiv.style.display = "none";  
 document.body.appendChild(oDiv);  
 }  
 return oDiv;  
 }  
 })();  
 document.getElementById("btn").onclick = function(){  
 var w = createDiv();  
 w.style.display = "block";  
 }  
</script>  
</body>

2》观察者模式（订阅模式）

发布===>订阅

理解：定义了一个对象一对多的关系，让多个观察者同时监听一个大对象，当一个大对象发生变化时，所有的依赖他的对象都得到了通知

<script>  
 var obj = {};//发布者  
 obj.list =[];//发布列表  
 obj.listen = function(fn){//增加订阅者 ..,fn是一个回调函数  
 obj.list.push(fn);  
 };  
 obj.message = function(){//发布消息  
 for(var i= 0,fn;fn=this.list[i++];){  
 fn.apply(this,arguments);  
 }  
 };  
 obj.listen(function(color,size){//小张订阅的消息  
 console.log("颜色是:"+color);  
 console.log("尺寸是:"+size);  
 }) ;  
 obj.listen = function(color,size){////小李订阅的消息...  
 console.log("颜色是:"+color);  
 console.log("尺寸是:"+size);  
 };  
 obj.message("红色",40);  
 obj.message("黑色",88);  
</script>

案例2：

<script>  
 var obj = {};  
 obj.list =[];  
 obj.listen = function(key,fn){  
 if(!this.list[key]){  
 this.list[key] = [];  
 }  
 this.list[key].push(fn);  
 }  
 obj.message = function(){  
 var key = Array.prototype.shift.call(arguments);//取出消息类型名称  
 var fns = this.list[key]; //取出消息对应回调函数集合  
 //如果没有订阅这个消息，直接返回  
 if(!fns || fns.length==0){  
 return;  
 }  
 for(var i= 0,fn;fn=fns[i++];){  
 fn.apply(this,arguments);//arguments发布消息附带参数  
 }  
 };  
 obj.listen("红色",function(size){  
 console.log("尺寸"+size);  
 });  
 obj.listen("黑色",function(size){  
 console.log("尺寸"+size);  
 });  
 obj.message("红色",40);  
 obj.message("黑色",42);  
</script>

2》观察者模式（订阅模式）

发布===>订阅

理解：定义了一个对象一对多的关系，让多个观察者同时监听一个大对象，当一个大对象发生变化时，所有的依赖他的对象都得到了通知

3》享元模式

案例:一次添加for循环实现10000次

<script>  
 function Iphone(model,screen,date,color){  
 this.model = model;  
 this.screen = screen;  
 this.date = date;  
 this.color = color;  
 }  
 var phones = [];  
 for(var i=0;i<10000;i++){  
 phones.push(new Iphone("iphone7s","5.0","2017-8-8","红色"));  
 }  
 console.log(phones);  
</script>

功能:优化程序的性能

<script>  
 //型号，屏幕大小，出厂日期， color  
 function Iphone(model,screen,date){  
 this.model = model;  
 this.screen = screen;  
 this.date = date;  
 }  
 var Fatory = (function(){  
 var iphones = {};  
 return {  
 get:function(model,screen,date){  
 var key = model+screen+date;  
 if(!iphones[key]){  
 iphones[key] = new Iphone(model,screen,date);  
 }  
 return iphones[key];  
 }  
 }  
 })();  
 function Iphones(model,screen,date,color){  
 this.weigth = Fatory.get(model,screen,date);  
 this.color = color;  
 }  
 var arrIphone = [];  
 for(var i=0;i<10000;i++){  
 arrIphone.push(new Iphones("iphone7s","5.0","8-8","红色"));  
 }  
 console.log(arrIphone);  
</script>

134：

console.log(jQuery);  
console.log($);

以上两中方法均为相同效果

;(function(){  
 var jQuery = function() {  
 return 111;  
 };  
 window.jQuery = window.$ = jQuery;  
})();

135:简单框架封装：  
<body>  
<div id="box"></div>  
<ul>  
 <li>1</li>  
 <li>2</li>  
 <li>3</li>  
 <li>4</li>  
</ul>  
<script src="myjq.js"></script>  
<script>  
 $("li").b();  
</script>  
</body>

函数的使用：  
;(function(){  
 var jQuery = function(selected) {  
 return new jQuery.prototype.init(selected); //返回一个对象  
 };  
 var markArray = function(arr,that){  
 var brr = that;  
 for(var i=0;i<arr.length;i++){  
 brr[i]=arr[i]  
 }  
 brr.length=arr.length;  
 return brr;  
 };  
 jQuery.prototype = {  
 init:function(selected){  
 //这是一个方法===》选择元素的  
 var dom = document.querySelectorAll(selected);  
 return markArray(dom,this); //返回一个对象  
 },  
 a:function(){  
 alert("aaaaa");  
 },  
 b:function(){  
 alert("bbbbb");  
 }  
 };  
 jQuery.prototype.init.prototype= jQuery.prototype;  
 window.jQuery = window.$ = jQuery;  
})();

136：函数的连续调用：

<body>  
<div id="box"></div>  
<ul>  
 <li>1</li>  
 <li>2</li>  
 <li>3</li>  
 <li>4</li>  
</ul>  
<script src="myjq.js"></script>  
<script>  
 $("li").a().b().c();  
</script>  
</body>

Js:

;(function(){  
 var jQuery = function(selected) {  
 return new jQuery.prototype.init(selected); //返回一个对象  
 };  
 var markArray = function(arr,that){  
 var brr = that;  
 for(var i=0;i<arr.length;i++){  
 brr[i]=arr[i]  
 }  
 brr.length=arr.length;  
 return brr;  
 };  
 jQuery.prototype = {  
 init:function(selected){  
 //这是一个方法===》选择元素的  
 var dom = document.querySelectorAll(selected);  
 return markArray(dom,this); //返回一个对象  
 },  
 a:function(){  
 alert("aaaaa");  
 return this;  
 },  
 b:function(){  
 alert("bbbbb");  
 return this;  
 },  
 c:function(){  
 alert("ccccc")  
 }  
 };  
 jQuery.prototype.init.prototype= jQuery.prototype;  
 window.jQuery = window.$ = jQuery;  
})();

137:函数封装“

<body>  
<div id="box">111111111</div>  
<ul>  
 <li>1</li>  
 <li>2</li>  
 <li>3</li>  
 <li>4</li>  
 <li>5</li>  
</ul>  
<script src="myjq.js"></script>  
<script>  
 /\*$("#box").hover(function(){  
 console.log($(this))  
 },function(){  
 alert("鼠标移出");  
 })\*/  
 /\*$("#box").css("background","red");  
 alert( $("#box").css("background") );\*/  
 $("ul li").first().css("background","red");  
 $("ul li").last().css("background","red");  
 $("ul li").eq(2).css("background","blue");  
</script>  
</body>

Js:  
;(function(){  
 var jQuery = function(selected) {  
 return new jQuery.prototype.init(selected); //返回一个对象  
 };  
 var markArray = function(arr,that){  
 var brr = that;  
 for(var i=0;i<arr.length;i++){  
 brr[i]=arr[i]  
 }  
 brr.length=arr.length;  
 return brr;  
 };  
 jQuery.prototype = {  
 init:function(selected){  
 //这是一个方法===》选择元素的  
 var dom = null;  
 if(typeof selected!="string"){  
 dom = [selected];  
 }else{  
 dom = document.querySelectorAll(selected);  
 }  
 return markArray(dom,this); //返回一个对象  
 },  
 hover:function(over,out){  
 this[0].onmouseover = over;  
 this[0].onmouseout = out;  
 return this;  
 },  
 css:function(attr,val){  
 var arg = arguments.length;  
 if(arg==2){  
 for(var i=0;i<this.length;i++){  
 this[i].style[attr] = val;  
 }  
 }else{  
 return getComputedStyle(this[0],null)[attr];  
 }  
 return this;  
 },  
 first:function(){  
 return $(this[0]);  
 },  
 last:function(){  
 return $(this[this.length-1]);  
 },  
 eq:function(num){  
 return $(this[num]);  
 }  
 };  
 jQuery.prototype.init.prototype= jQuery.prototype;  
 window.jQuery = window.$ = jQuery;  
})();