Git使用：

Git init :创建本地仓库

通过命令：git remote add origin 远程库的HTTPS地址或者是SSH地址添加远程仓库

通过命令：git pull --rebase origin master获取远程仓库与本地同步（远程仓库只有有文件就需要进行这一个命令处理）此时test01文件夹中就有了远程仓库【README.md】文件

git push -u origin master （第一次需要-u）

配置用户名：git config --global user.name “github上注册的用户名”  
配置用户邮箱：git config --global user.email “GitHub上注册时的邮箱”

配置密钥：ssh-keygen -t rsa -C "GitHub上注册时的邮箱 "

常用标签

标题标签：h1-h6

段落标签：p

换行标签：br

大盒子标签：div

小盒子标签：span

文字省略号处理：

.container .hot .hot-list ul li .message {

  width: 250px;

  overflow: hidden;

  white-space: nowrap;

  text-overflow: ellipsis;

  font-size: 16px;

  color: #292828;

  margin: 15px 0 15px 10px;

}

a标签常用属性

 <a href="http://www.baidu.com " target="\_blank" >百度</a>

target=”\_blank” 在新窗口打开

target=”\_self” 在当前窗口打开

a标签herf里面常用的几个值

具体跳转的地址 herf=”http://www.baidu.com”

刷新：herf=””

回到顶部：herf=”#”

禁止跳转：herf=”javascript:;” absolute

锚点跳转：作用：点击某个a标签的时候，可以跳转到相应的模块

<div id="goods">模块</div>

    <a href="#goods">跳转到模块</a>

a标签的访问样式

a:link{text-decoration:none; } /\* 指正常的未被访问过的链接\*/

a:visited{text-decoration:none; }/\*指已经访问过的链接\*/

a:hover{text-decoration:none;}/\*指鼠标在链接\*/}

a:active{text-decoration:none;}/\* 指正在点的链接\*/

鼠标变小手：

cursor:pointer;

table表格标签：

tr：代表每一行

td：代表每一个单元格

th：表头标签

caption：表格标签

table相关属性：

align：对其方式：left , center , right

width：宽度

height：高度

cellpadding：内容到单元框的距离

cellspacing：单元格到单元格的距离

border：边框，border=”1”、border=””

colspan：跨列

rowspan：跨行

三大列表标签

有序列表：

<ol>

<li>吃饭</li>

<li>睡觉</li>

<li>打豆豆</li>

</ol>

无序列表：

<ul>

<li>吃饭</li>

<li>睡觉</li>

<li>打豆豆</li>

</ul>

自定义标签：

<dl>

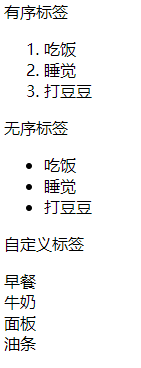
<dt>早餐</dt>

<dt>牛奶</dt>

<dt>面板</dt>

<dt>油条</dt>

</dl>



表单元素：

表单域：form

作用：包含在区域内，提交表单时会将区域内的信息提交给后台

表单域常见的属性：

|  |  |  |
| --- | --- | --- |
| 属性 | 属性值 | 含义 |
| action | url | 提交信息的地址 |
| method | post/get | 提交信息的方式 |
| name | 名称 | 用于区分一个页面中多个表单域 |

常用的表单元素

input类型的表单元素

button按钮

select下拉框表单元素

textarea文本域

input标签不同的type类型

|  |  |
| --- | --- |
| type | 含义 |
| text | 文本输入框 |
| password | 密码输入框 |
| radio | 单选 |
| button | 按钮 |
| file | 上传文件 |
| date | 日期 |
| color | 颜色面板 |
| range | 范围 |
| number | 数字 |
| submit | 提交 |
| reset | 重置 |

input常用的属性

|  |  |
| --- | --- |
| name | 规定input的值 |
| checked | 默认被选中 |
| maxlength | 最大字符长度 |
| placeholder | 描述输入字段预期值的提示信息 |
| autofocus | 自动获取焦点 |

button按钮:两种方式

 <input type="reset" value="清空">

    <input type="button" value="开始">

 <button>支付</button>

单选按钮功能实现：

请选择您的性别：<br />

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

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

文字关联功能实现：

为了增加用户体验，点击文字的时候，选中input

label：当点击文字的时候，可以让和它关联的表单元素被选中

<form action="" method="GET" name=""></form>

    请选择您的性别：<br />

    <label for="man">男</label><input type="radio" name="gender" value="男" id="man">

    <label for="woman">女</label><input type="radio" name="gender" value="女" id="woman">

select下拉框：

 <form action="">

       <select name="" id="">

           <option value="北京" >北京</option>

           <option value="北京" selected>重庆</option>

           <option value="北京" >山西</option>

           <option value="北京" disabled>河北</option>

       </select>

    </form>

textarea：文本域

cols:每行最多显示多少个

rows：能显示多少列

 <form action="">

            <textarea name="" id="" cols="10" rows="3"></textarea>

        </form>

字符实体：浏览器中，实体编号比实体名称兼容性强

|  |  |  |  |
| --- | --- | --- | --- |
| **显示结果** | **描述** | **实体名称** | **实体编号** |
|  | 空格 | &nbsp; | &#160; |
| < | 小于号 | &lt; | &#60; |
| > | 大于号 | &gt; | &#62; |
| & | 和号 | &amp; | &#38; |
| " | 引号 | &quot; | &#34; |
| ' | 撇号 | &apos; (IE不支持) | &#39; |
| ￠ | 分（cent） | &cent; | &#162; |
| £ | 镑（pound） | &pound; | &#163; |
| ¥ | 元（yen） | &yen; | &#165; |
| € | 欧元（euro） | &euro; | &#8364; |
| § | 小节 | &sect; | &#167; |
| © | 版权（copyright） | &copy; | &#169; |
| ® | 注册商标 | &reg; | &#174; |
| ™ | 商标 | &trade; | &#8482; |
| × | 乘号 | &times; | &#215; |
| ÷ | 除号 | &divide; | &#247; |

css（层叠样式表）：



选择器：具体的元素（选中谁）

{}：花括号里面存放选择中元素的样式（要给这个元素化怎样的妆容）

具体的样式：包含属性名和属性值，它们中间用“:”进行连接

生命语句和语句之间我们需要用“;”进行分割

字体相关样式：

font-family:”宋体”；多个字体用,隔开，优先加载左边的字体，中文样式用””，多个英文用单引号，单个单词，不用引号

font-size:30px 最小是12px，谷歌默认14px

font-weight:100-900 normal（400）:默认，bold（700）：粗的，bolder：更粗的，lighter：更细的

font-style： normal:正常 italic：倾斜

字体复合属性：字体大小和字体两项必须有

p {

            font: italic 700 30px "宋体";

        }

css常用的样式

css的文本属性：主要是定义文本的外观，它包含文本颜色、文本对齐、文本缩进、文本装饰线、行高

|  |  |
| --- | --- |
| 属性 | 作用 |
| color | 文本设置颜色：英文颜色单词，16禁止，RGB |
| text-align | 文本水平对齐方式：left、center、right |
| text-indent | 文本缩进（缩进种鸽子的距离2em） |
| text-decoration | 文本装饰线 |
| line-height | 行高，控制行与行直接的距离 |

color

div{

            color: brown;

            color: #001254;

            color: rgb(0,120,0);

        }

text-indent：em是相对单位：1em就是本元素的大小，2em就是2个本元素大小

text-decoration:

   a {

            text-decoration: none;

            text-decoration: underline;

            text-decoration: overline;

            text-decoration: line-through;

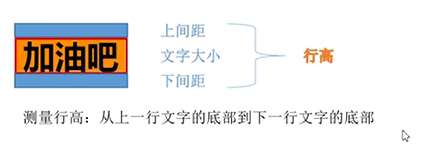
          }

line-height:行高

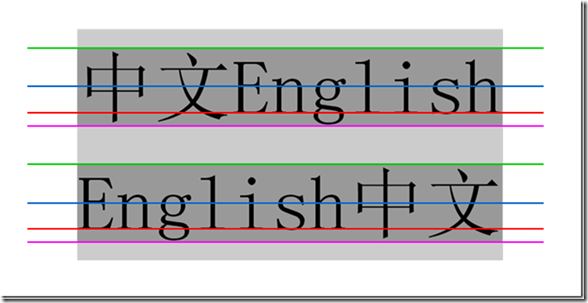
div{

line-height: 36px

}



line-height:行高是两行内容的基线之间的距离



文字垂直居中：

h1{

    height: 100px;

    line-height: 100px;

    background-color: salmon;

}

div边框的样式：

none (没有边框，无论边框宽度设为多大)  
  
　　dotted (点线式边框)  
  
　　dashed (破折线式边框)  
  
　　solid (直线式边框)  
  
　　double (双线式边框)  
  
　　groove (槽线式边框)  
  
　　ridge(脊线式边框)  
  
　　inset (内嵌效果的边框)  
  
　　outset (突起效果的边框)

border-color有四个值，顺时针顺序。三个值：顺时针，未设置的显示对边颜色

两个值，对边颜色显示。

border：复合属性：使用较多

顺序：border-width，border-style，border-color

块级元素水平居中：

margin 0 auto;

清除默认样式：\*通配符选择器

\*{

margin: 0;

padding: 0;

}

分组选择器：

ul,ol{

list-style: none;

}

ul,ol a标签格式清除

ul,ol{

        list-style: none;

    }

    a{

        text-decoration: none;

    }

css引入的四种方式：内嵌和外链使用较多

行内样式：

  <div style="width: 100px; height: 100px; background-color: lightgreen; "></div>

内嵌样式：

<style>

    div{

        width: 50px;

        height: 100px;

        background-color: lightpink;

    }

</style>

外链：

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

导入：

<style>

    @import url("index.css");

link和import的区别？

1. link除了可以加载css外，还能定义RSS，定义rel连接属性等其他作用；@import只能加载css。
2. 加载顺序：link 引用的css会在页面被加载的时候同时加载；@import引用的css会等到页面全部被下载完再被加载。
3. [兼容性的区别。@import是css2.1提出的，老的浏览器不支持。ie5以上的才能识别。link浏览器都支持。](mailto:兼容性的区别。@import是css2.1提出的，老的浏览器不支持。ie5以上的才能识别。link浏览器都支持。)
4. 使用javascript可以控制link，但@import却无法控制

css选择器：选择器可分为基础选择器和复合选择器

基础选择器：

通配符选择器：权重0

\*{

color: red;

}

标签选择器： 权重1

p{

color: red;

}

类选择器：权重10 class可设置多个名字

.p1{

width: 100px;

height: 100px;

background: green;

}

.f22{

}

<p class=”p1 f22”>p1</p>

<p class=”p1”>p2</p>

id选择器：权重100 id名必须唯一，不能重复

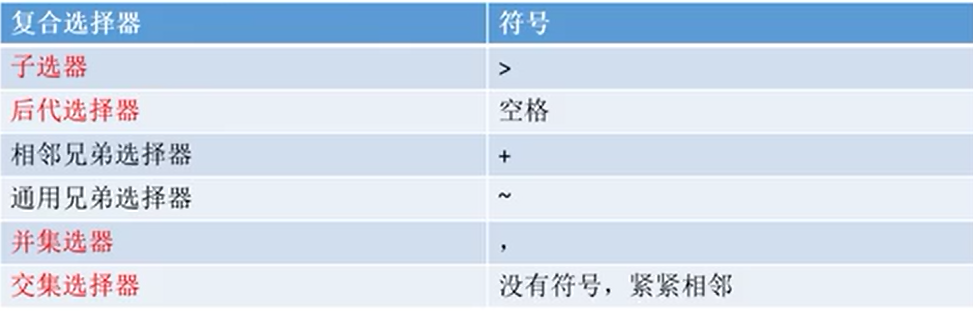
<style>

#box{color:red;}

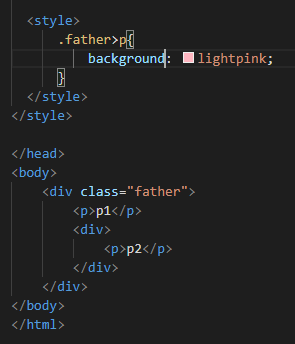
</style>

<div id=”box”></div>

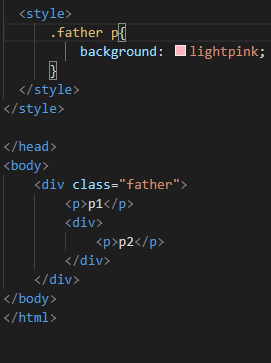
复合选择器：



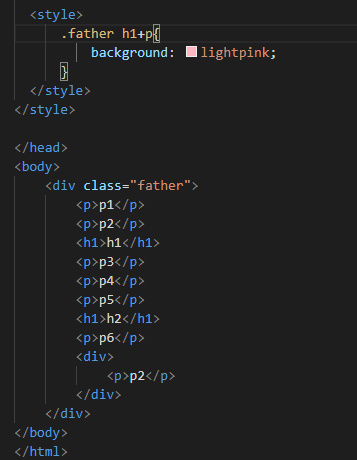
子选择器：> 权重：类选择器的权重+标签选择器（共计11）

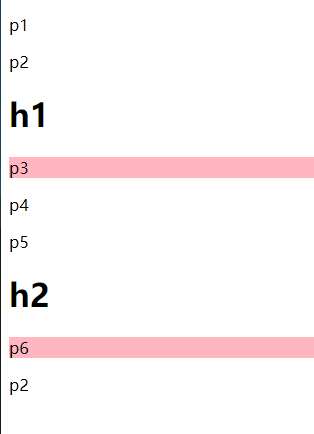


后代选择器：空格 权重也是相加

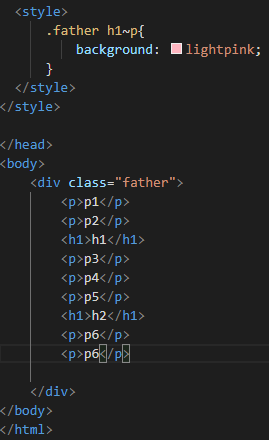


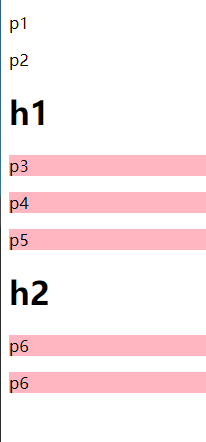
相邻兄弟选择器：+ 只能向下寻找，只找一个。权重相加



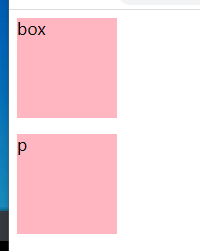


通用兄弟选择器：~ 只能向下寻找 只要是相邻的兄弟，挨着传染

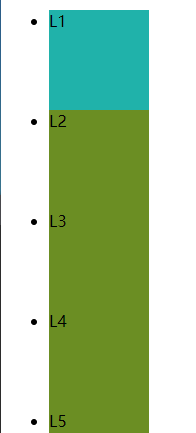




并集选择器：, 权重相加



交集选择器：没有符号，紧紧相邻 权重相加



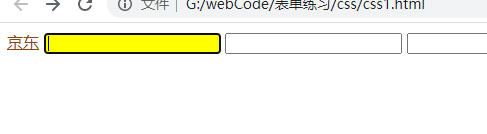
复合选择器之伪类选择器：

链接伪类选择器：设置的时候，要有一定的顺序



focus伪类选择器：





Css三大特性：

继承性：

继承是一种规则，允许样式不仅应用于特定的html标签元素，而且可以应用于后代元素。简单理解就是父元素设置的一些属性，子元素/后代可以不用设置，也可使用。

并不是所有的属性都可以继承，以color/font-/test-/line-开头的属性都可以继承

层叠性：

层叠性就是css处理冲突的一种能力，就是多种选择器选择了同一个元素，又对相同的属性设置了不同的属性值，这个时候就会发生层叠性。

1. 如果权重相同，则看代码先后顺序，后面的会覆盖前面的
2. 如果权重不同，谁的权重大，就听谁的

优先级：

!important（无穷大）>行内样式（1000）>id（100）>类（10）>标签（1）>通配符、继承（0）>

注意：如果该元素没有被直接选中，而是通过继承得到的属性，那么不管它的父元素权重多高，该元素的权重也是0

标签的分类

块级元素：

独占一行

可以设置宽、高、内边距、外边距

如果不设置宽度，可以继承父级的宽度

是一个容器或者盒子，里面可以放行内或者块级元素

常见的块级元素：div、p、h1-h6、ul、ol、li等

注意：文字类的元素内不能使用块级元素。

p标签主要用来存放文字，因此p里面不能放块级元素

h1-h6也是文字类的块级标签，里面也不能放块级元素

行内元素：行内块也是行内元素的一种

相邻的行内元素可以在一行上

设置宽、高无效、盒子的大小就是内容的大小

只能放文字或者其他的行内元素

常见的行内元素：a、span、b、strong、em等格式化标签

注意：a标签里面不能再放a标签

行内块元素：

行内块元素：行内块是特殊的行内元素

可以和其他的行内（行内块）元素共占一行，但是中间会出现缝隙

可以设置宽、高

盒子默认的大小就是本身内容的大小

常见的块级元素：img、input、table、video等

标签的互相转换：

转换为块级 display:block;

转换为行内 display:inline;

转换为行内块 display:inline-block

display还有一个值：none；让元素隐藏（完全隐藏这个元素）

盒模型

盒模型是一种写网页的思维模式，每一个元素都可以看成一个盒子，为了方便记忆，我们可以把它想象成手机盒，手机就是content（内容），白素填充物就是padding（填充），手机盒子就是边框border（边框）。多个手机盒子之间距离就是margin（边缘）

注意：行内元素不要用padding-top、padding-botton、margin-top、matgin-botton。因为行内元素设置了上下的宽度不准确，会比设置的距离小



盒模型的两种分类：

标准盒模型：box-sizing: content-box; （默认）

ie盒模型：box-sizing: border-box;

区别：标准盒模型中，代码里面缩写的width盒height指的是盒模型中content部分的宽和高

ie盒模型中，代码里面所写的width和height指的是总盒子的宽和高（在实战项目中经常用这个）

面试题之外边距合并：

外边距塌陷（穿透）

描述：有一个大盒子里面包含小盒子，如果大盒子没有设置padding-top或者是border-top，直接给小盒子设置margin-top，大盒子会跟着小盒子一起向下移动，这个现象就是margin值穿透问题。

解决办法：

给大盒子加border-top:1px solid transparent；

给大盒子加 overflow:hidden;

既然margin-top有这样的问题，我们可以给大盒子增加padding-top，来规避这个问题。

还有浮动固定定位、绝对定位、行内块等也可以解决这个问题

面试题值外边距合并：



两个块级元素转换成行内块存在的问题：会有一个空隙（原因是两行代码中间有一个回车）



解决办法：

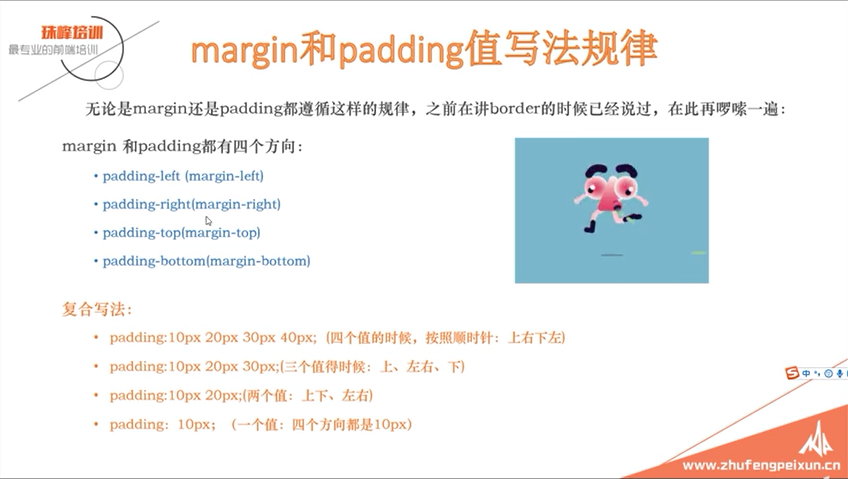
1. 消除换行符
2. 给父元素设置font-size:0（如果文字消失了，单独设置一下包含文字标签的font-size）

img存在的缝隙问题：

1. 把img转换为块级
2. 给父盒子设置一个固定的高度

margin和padding值写法规律

margin: 0 auto; (代表上下是0，左右是auto)



相关知识1：



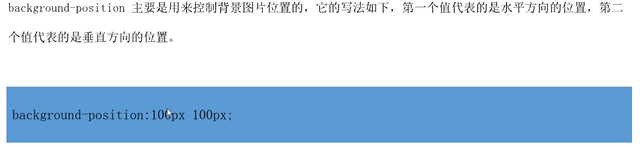
background-repeat：



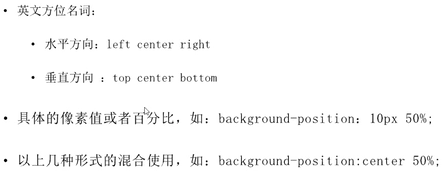
background-attachment：



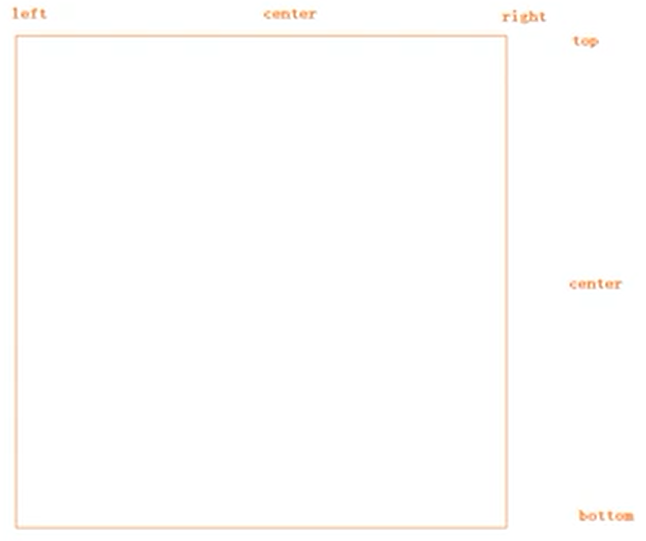
background-position：英文单词、像素、百分比都可以



background-position的值有多种表示形式：







background复合写法：有顺序

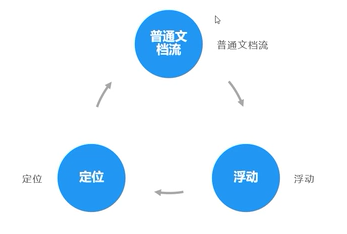
background：background-color（背景颜色） background-image（背景图片） background-repeat（平铺方式） background-attachment（背景是否跟着页面滚动而滚动） background-position（背景图片的位置）;

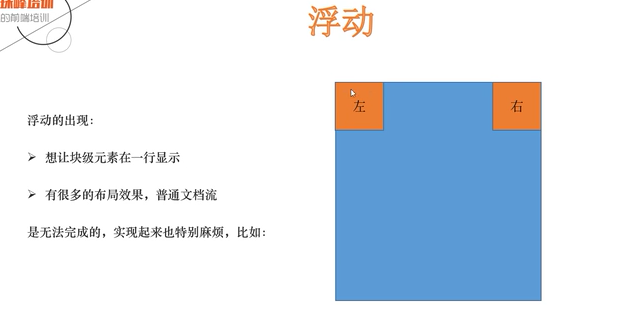
background相关知识2：

background-size：调节背景图片的尺寸



传统网页的布局方式：



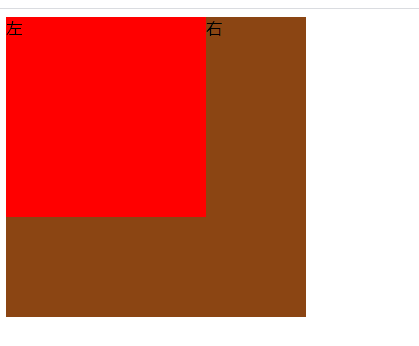


float：浮动



浮动的特点：

1.脱离了普通文档流。设置了浮动后，就飘了

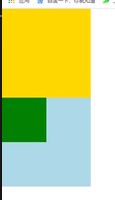
2.浮动元素具备行内块的特点

可以左右排列

即使是行内元素，浮动之后写大小也可以起作用

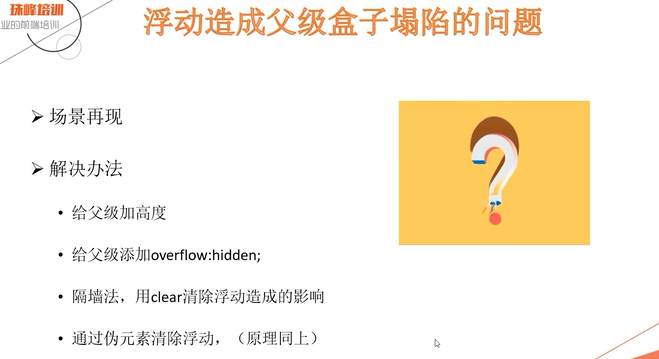
即使是块级元素，浮动之后，如果不写宽度，不再默认占满父盒子，而是自身内容的决定

浮动的元素碰到普通文档流的边缘就会停下来，浮动的元素之后影响他后面的普通文档流（对前面不造成影响）



浮动技巧：





1. 少量可以。大量就比较繁琐
2. overflow:hidden 溢出隐藏（溢出盒子的内容不显示）
3. 一出现这个问题就需要添加div，clear。繁琐



4.在父级加上这个样式，可用css

.clearfix::after {

            content: "";

            font-size: 0;

            display: block;

            height: 0;

            visibility: hidden;

            clear: both;

        }

        .clearfix {

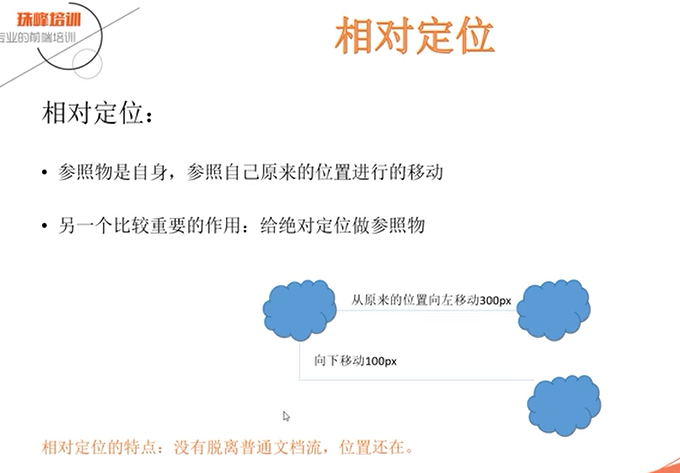
            \*zoom: 1;

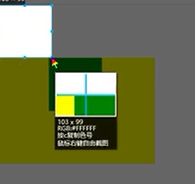
        }

定位：



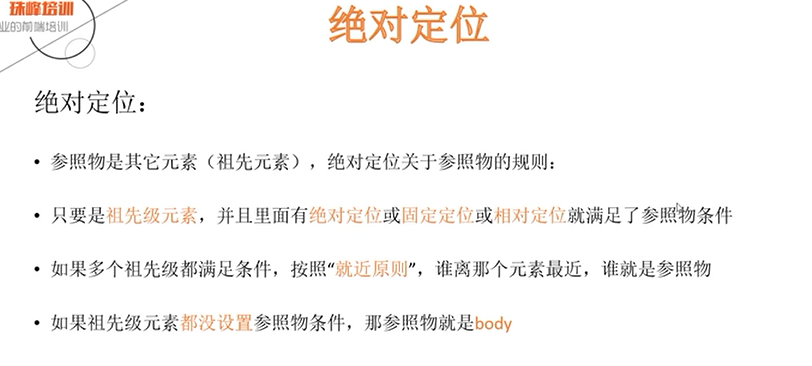
相对定位





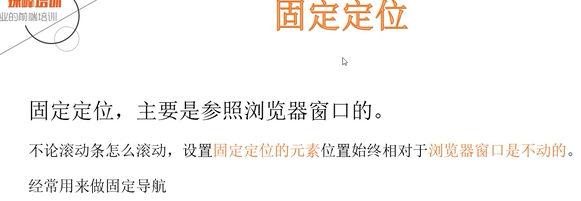
绝对定位：







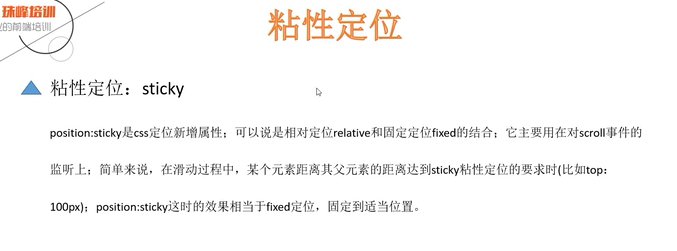
固定定位：



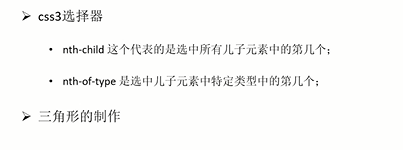
计算公式：让一个元素在屏幕中或者是一个盒子中水平垂直居中的公式



粘性定位：



css3选择器：





如何制作一个三角形：将不需要的颜色设置为透明

div {

            border: 50px solid red;

            /\* transparent:透明  \*/

            border-color: chartreuse gold blue transparent;

            width: 0;

        }

javascript

javascript中六种数据类型

String：字符串

Number：数值

Boolean：布尔值

Null：空值

Undefined：未定义

Object：对象

前5种属于基本数据类型，object属于引用数据类型

typeof可以检查变量类型

String：

用双引号或者单引号引起来。但是不能混用，如”team’。

引号不能嵌套

    var str="我说："今天天气很不错"!!";

但是可以这样使用

      var str="我说：'今天天气很不错'!!";

“\”表示转义字符

Number：

包含整数和浮点数（小数）

Number.MAX\_VALUE表示最大的数值（1.7976931348623157e+308）

如果Number的数字超过了最大值，则会返回一个Infinity（正无穷）。-Infinity（负无穷）。正负无穷都是Number类型

Number.MIN\_VALUE表示最小的数值（5e-324）

NaN：not a Number：如console.log("a"\*"a") 是string数据类型

js在整数运算基本可以保证精确。

如果js进行浮点数的计算，可能得到一个不精确的结果。

var a = 0.1+0.2

console.log(a)



不涉及钱的，可以舍去多余部分。千万不要用js进行精确度较高的运算。尽量放到服务器中计算

Boolean：布尔值，判断真假

Null：

只有一个值null，用来表示为空的对象，用typeof检测，会返回object。

undefined：

var a; console.log(a)。声明一个变量，但是不赋值，就会是undefined。typeof检查undefined还是返回undefined

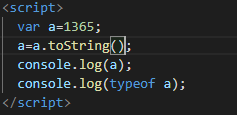
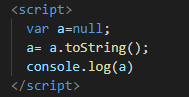
强制类型转换：

主要是指将其他的数据类型转换为String Number Boolean

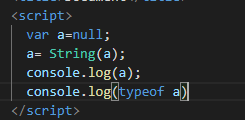
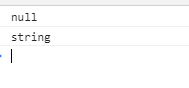
1.将其他的数据类型转换为String

方法1：toString()。该方法不会影响到原变量。但是该方法有局限性，如a=null或者undefined，会报错。因为这两个没有toString()方法。



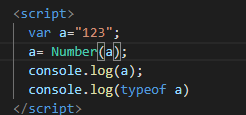
 

方法2：调用String()函数。使用String()函数做强制类型转换时，对于Number和Boolean实际上就是调用toString()方法。但是null和undefined就不会调用toString()方法，它会将null和undefined直接转换为字符串

3.将其他数据类型转换为Number

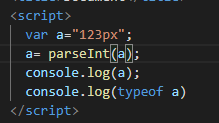
转换方式一、使用Number()函数。如果是纯数字的字符串，则直接将其转换为数字。如果字符串中有非数字内容，则转换为NaN。如果字符串是空串或者全是空格的字符串则转换为0。如果是Boolean值，则true为1，false为0。如果是null，则转换为0，undefined转Number则转换为NaN。

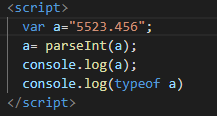


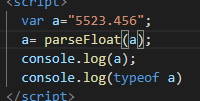
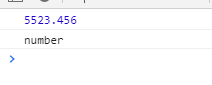
转换方式二、专门用来对付字符串

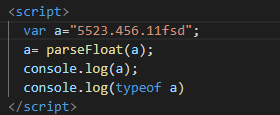
parseInt()把一个字符串转换为一个整数

pasefloat()把一个字符串转换为一个浮点数

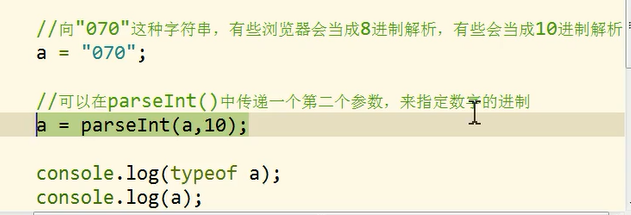
如果对非String使用parseInt或者parseFloat会先将其转换为String，然后在操作

其他进制的数字表示：

在js中表示16进制的数字，以0x开头

如果表示8进制数字，则以0开头

如果表示2进制数字，则以0b开头，但是不是所有的浏览器都支持



转换Boolean：

使用Boolean（）函数。

数字转布尔。除了0和NaN，都是true

字符串转布尔，处理空串，其余的都是true。

Null和undefined都是false

对象也会转换为true

运算符（操作符）：

通过运算符可以对一个或者多个值进行运算，并获取运算结果

比如typeof也是一个运算符，可以来获得一个类型

会将该值得类型以字符串类型返回

算数运算符：+-\*/%

对非number类型的值进行运算时，会将这些值转换为number然后在运算

任何值和NaN做运算都是NaN

“+”字符串相加，则会将两个字符串拼接 如 a = “123”+“456”。 a = “123456”

任何的值和字符串做加法运算，都转换先转换为字符串在拼串，可以用这个特点，可以将任意的数据类型转换为String，只需为任意数据类型+一个“”空串，由浏览器自动完成。实际上也是调换了String（）函数

“-”，“\*”，“/”。如果是数字和字符串进行减乘除，则会将字符串转成数字再运算，可以利用这个特性，做数值的类型转换，原理和使用Number（）函数一样

%取模运算（取余）

一元运算符：

只需要一个操作数

+ 正号

Var a = 123;

a = +a

正号不会对数值产生任何影响

* 负号

负号可以对数值进行负号取反

对于非number类型的值会先转换为number，再取反

可以对其他的数据类型使用+，可以将其转换为number，他的原理和使用number（）函数一样。

例如：var a = “18”; a = + a 结果会将a变为number类型，a的结果为数值18

自增 ++

通过自增可以使变量在自身的基础上增加1，对一个变量自增后，原变量的值会立即自增1。自增分为两种，后++（a++）和前++（++a）。无论是a++还是++a，都会立即使原变量的值自增1，不同的是a++和++a的值不同。a++的值等于原变量的值（自增前的值），++a的值等于自增后的值。

var a = 1; a++ 结果为2

自减

通过自减可以在原变量的基础上减1。前--，后--。原理同上

19

逻辑运算符：

js中提供了三种逻辑运算符

！ 非

- ！可以用来对一个值进行非运算，如果进行两次取反，值不会变化

如果对非布尔值进行非运算。则会先将该值转化为布尔值再进行运算。

所以我们可以利用该特点对一个其他数据类型转换为布尔值，可以用两个！！实现类型转换，且值不变

&& 与

可以对符号两侧的值进行与运算，并返回结果

运算规则—全真为真，有假为假

短路“与”，如果第一个值为false，则不会再检查第二个值

如果第一个值为true，则需要检查第二个值

|| 或

全假为假，有真为真。

短路“或”，如果第一个值为true。则不会检查第二个，如果第一个值为false，则需要检查第二个值

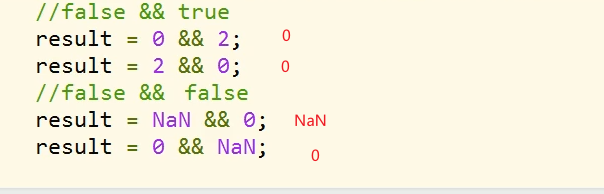
&& || 非布尔值情况

会先将其转换为布尔值，然后在运算。会反馈原值。

如：var result = 1 && 2; 结果会返回2。

如果两个值都为true。则返回第二个值。如果有false，则返回false

如果两个都是false，则会返回第一个值。



与运算，如果第一个值为true，则必然返回第二个值

如果第一个值为false，则必然返回第一个值

或运算：如果第一个值为true，则必然返回第一个值

如果第一个值为false，则必然返回第二个值

赋值运算符：

=

可以将右侧的值赋值给左侧的变量

+=

var a=10 a+=5 等价于a=a+5

-=，\*=，/=，%= :同“+=”类似

关系运算符：

通过关系运算符可以比较两个值之间的大小关系

如果关系成立，会返回true，如果关系不成立，则返回false

>左侧值大于右侧，则返回true，不成立则反馈false

<，=，>=,<= 同“>”类似（与数学的规则相同）

非数值的情况：

会将非数值转换为number进行比较。任何值和NaN作比较都是false

如果符号两端都是字符串，不会讲其转换成数字比较。而会分别比较字符串的Unicode编码。比较字符编码的时候，是一位一位比较。如果两位一样，则比较下一位。可以借用它来对英文排序，比较中文时，没有意义

如果是比较两个字符串的数字，可能会得到不可预期的结果。一定需要转型。可以在字符串数字前加一个“+”

如果：console.log(“1234613” > +“5”) 结果为true，如果不转型，结果为false

Unicode编码表：

在console输出一个Unicode字符，需要用\u 转义+四位编码（16进制）

在网页中使用Unicode编码：&#编码；这里的编码需要的是10进制，可以使用计算器转换

相等运算符：

比较两个值是否相等。如果相等返回true。否则返回false

使用==来做相等运算

--当使用==来做比较时，如果值的类型不同，则会自动进行类型的转换，将其转换成相同的类型再比较

Undefined衍生自null 。所以这两个值做相等判断时，会返回true

NaN不和任何值相等，包括本身

如果需要判断一个值或者变量是否是NaN。可以使用isNaN()函数。

如果是NaN，则反馈true。否则返回false

不相等：！=

用来判断两个值是否不相等。如果不相等返回true，否则反馈false

会对变量进行自动类型转换。如果转换后相等，也会返回false

===：全等：用来判断两个值是否全等，他是==类似，不同的是，它不会做自动类型转换，如果两个值的类型不同，直接返回false

！==：不全等。用来判断两个值是否不全等，和！=类似。不同的是，它不会做自动类型转换，如果两个值的类型不同，直接返回true

条件运算符（三元运算符）：

语法：？ ：

条件表达式？语句1：语句2；

执行流程：

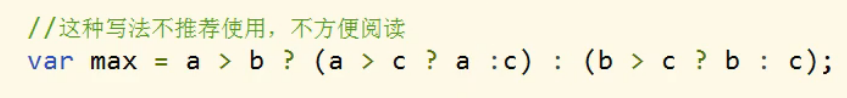
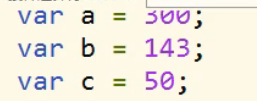
条件运算符在执行的时候，首先对条件表达式进行求值，

如果该值为true，则执行语句1.并返回执行结果

如果该值为false，则执行语句2，并返回执行结果

如果条件表达式的求值结果是非布尔值，会先转换为布尔值，再运算

如：var a=10; var b = 20; var max = a>b ? a: b; max结果为20



运算符的优先级：

，运算符：可以分割多个语句，一般可以在声明多个变量时使用

Var a ，b ，c; 等价于var a; var b ; var c;

和数学中一样，js中运算符也有优先级

如：先乘除后加减

Js中有一个运算符优先级表。在表中，越靠上优先级越高。优先级越高。越优先计算。如果优先级一样，则从左往右依次计算。不需要记忆。一般使用（）改变优先级



代码块：

在js中可以使用{}来为语句进行分组

同一个大块中的语句称为是一组语句，要么都执行，要么都不执行

一个{}中的语句，我们称之为代码块

在代码块后边，不用再编写；了

Js中的代码块只有分组作用，没有其他的作用

代码块的内容，在外部是完全可见的

流程控制语句：

分类：

1. 条件判断语句
2. 条件分支语句
3. 循环语句

条件判断语句：可以在语句执行之前判断，如果成立，执行，不成立，不执行

if语句

语法1：if（条件表达式）{

语句

}

条件表达式为true。则执行if后的语句。如果为false，则不执行if后的语句

if语句只能控制紧随其后的，如果需要管理多个，可使用代码块{}。实际开发中，尽量加上{}。If语句中，只会有一个代码块被执行，如果有一个代码块被执行，则立刻终止代码向下执行

语法2：if（）{

语句1

}else{

语句2

}

语法3：

if（）{

语句1

}else if{

语句2

}else if{

语句3

}else{

语句4

}

Prompt（）函数：可以弹出一个提示框，自带文本框。用户输入的内容可以定一个变量来接收这个内容。返回值是String类型

条件分支语句：switch 在执行时会依次将case后的表达式的值和switch后的表达式的值进行全等比较。如果相等，则执该case下的语句，case下的语句都会执行，但是遇到break（退出switch语句），会立刻停止执行。否则向下运行。如果所有case比较后，都为false，则会执行default语句。

语法：

switch(条件表达式){

case 表达式：

语句…

break；

case 表达式：

语句…

break；

case 表达式：

语句…

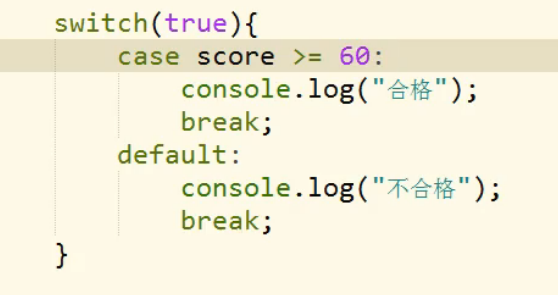
break；

case default：

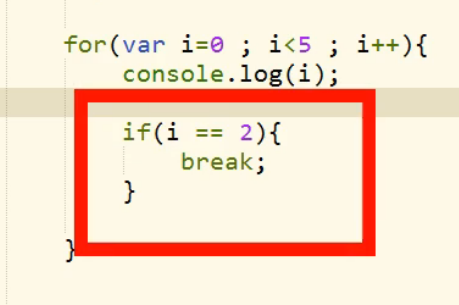
语句…

break；

}



Break 不能用来打断if，但是如果是在for循环内部。可以在if里，效果是打断for循环。Break会立即中断离他最近的循环

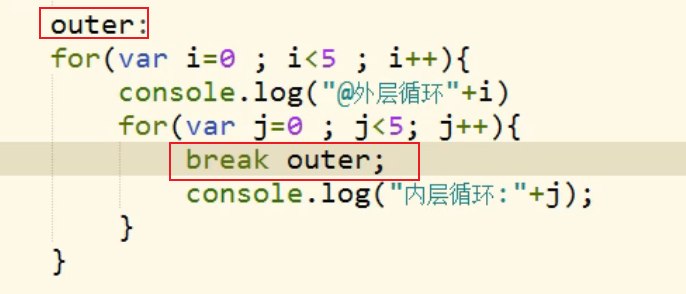


可以为循环语句创建一个label来表示当前的循环

Label:循环语句

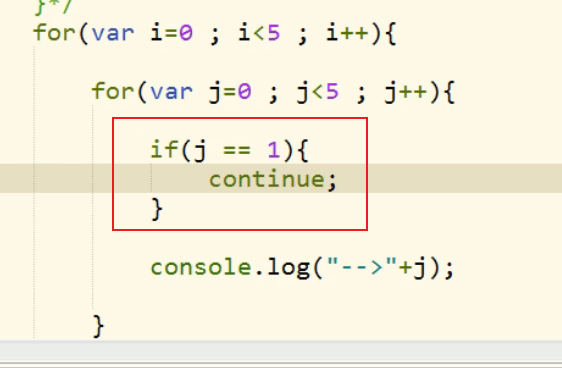
使用break语句时，可以在break后跟着一个label。

这样break会结束指定的循环，而不是最近的循环



Continue：跳过当次循环，只会跳过一次。对后续的循环不影响

同样也是离他最近的循环起作用。也可以使用label标签。



console.time(“名字”);可以开启一个计时器，它需要一个字符串作为参数，这个字符串将作为计时器的标识。

console.timeEnd(“名字”)：停止一个计时器。也需要一个名字作为参数。需要停止谁，就填谁

While:循环 通过循环语句可以反复的执行一段代码多次。先对条件表达式进行判断，如果为true，则执行语句。再判断表达式，为true，执行语句。依次类推，直到条件表达式为false

语法：

while(条件表达式){

语句…

}

do…while：和while类似，区别在于，while是先判断再执行，do。。。while是先执行一次（无论条件是否成立），再判断。

语法：

do{

语句

}while(条件表达式)

for 循环：执行顺序：1初始化变量，2判断条件表达式，为true，继续执行，为false，中止执行，3更新表达式执行完毕，4执行语句。继续重复2。直到判断表达式为false。

语法：

For(初始化表达式；条件表达式；更新表达式){

语句

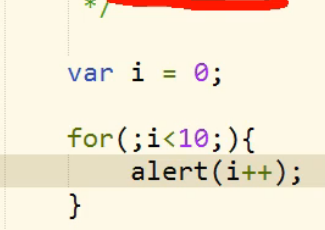
}

for（var i = 0;i<10;i++）{

语句

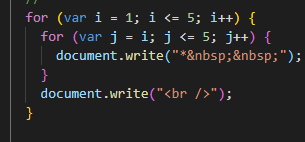
}

这样书写也可以



如果for循环中，不写任何表达式，只写两个分号。会变成死循环

嵌套for循环



对象：

只要看到的值不少5种基本数据类型，全都是对象

对象属于一种复合数据类型，在对象中可以保存不同的数据类型

对象的分类：

1. 由js标准中定义的对象，在任何的js实现中都可以使用

如：Math ,String Number Function Object……

1. 宿主对象

由js运行环境提供的对象，目前来讲主要指由浏览器提供的对象

比如BOM DOM 。 console document

1. 自定义对象

由开发人员自己创建的对象

对象的基本操作：

创建对象：使用new关键字调用的函数，是构造函数constructor

构造函数是专门用来创建对象的函数

var obj = new Object();

在对象中保存的值成为属性

语法：对象.属性名=属性值;

obj.name=”孙悟空”；

读取对象中的属性：

对象名.属性名，如果读取对象中没有的属性，不会报错，会返回undefined

修改对象的属性值：

对象.属性名 = 新值;

删除对象的属性：

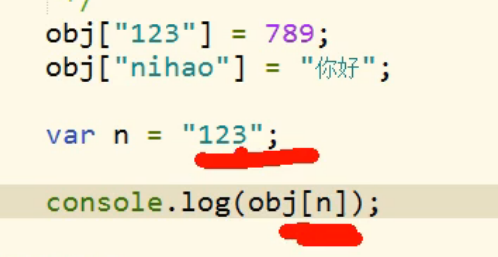
delete 对象.属性名

属性名和属性值：

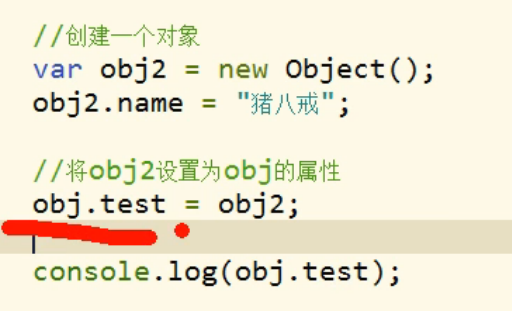
属性名：对象的属性名不强制要求遵守标识符规范，尽量遵循标识符规范。

如果需要使用特殊的属性名，不能采用.的方式。需要使用：

语法：obj [“123”] = 789; 读取时也需要采用这种方式 console.log(obj[“123”])。使用[]这种形式去操作属性，更加灵活。可以直接传递一个变量。

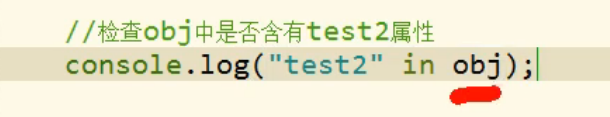


属性值：js对象的属性值可以是任意的数据类型也可以是一个对象。可以嵌套



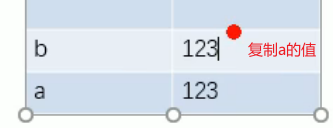
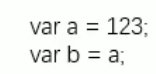
in运算符：通过该运算符可以检查一个对象中是否含有指定的属性，如果有，返回true，没有返回false

语法：”属性名” in 对象

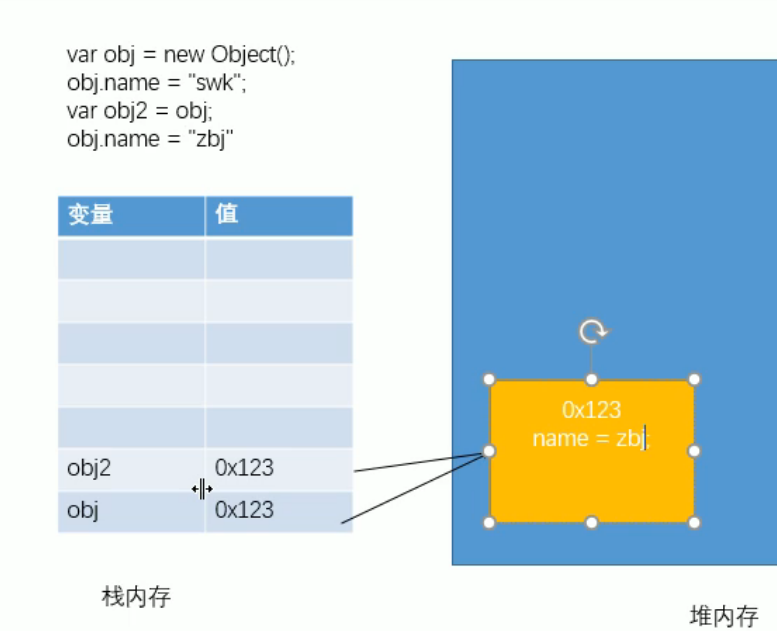


基本和引用数据类型：

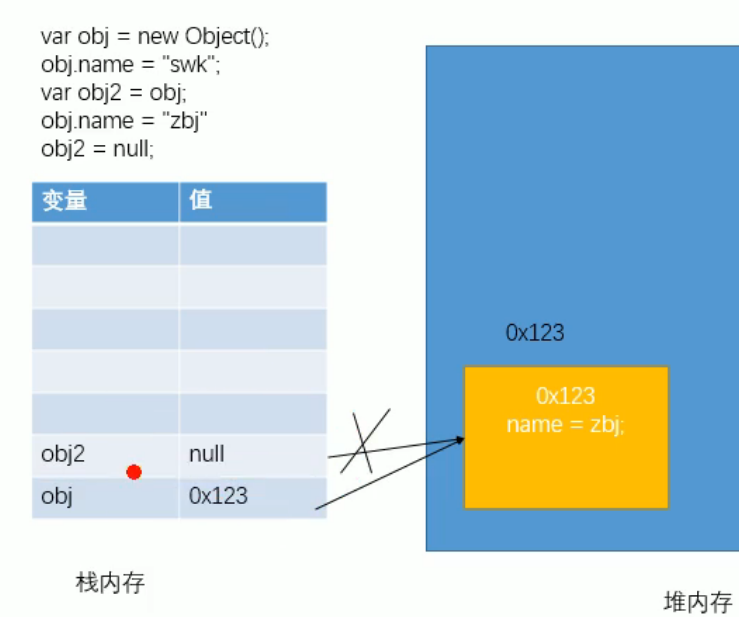
js中的变量都是保存在栈内存中的。基本数据类型的值直接在栈内存中存储。值与值之间是独立存在的，修改一个变量不会影响其他的变量。



对象是保存到堆内存中的，每创建一个新的对象，就会在堆内存中开辟一个新的空间，而变量保存的是对象的地址（对象的引用）。如果两个变量保存的是同一个对象引用。当其中一个变量的值改动， 另一个也会随之改动。

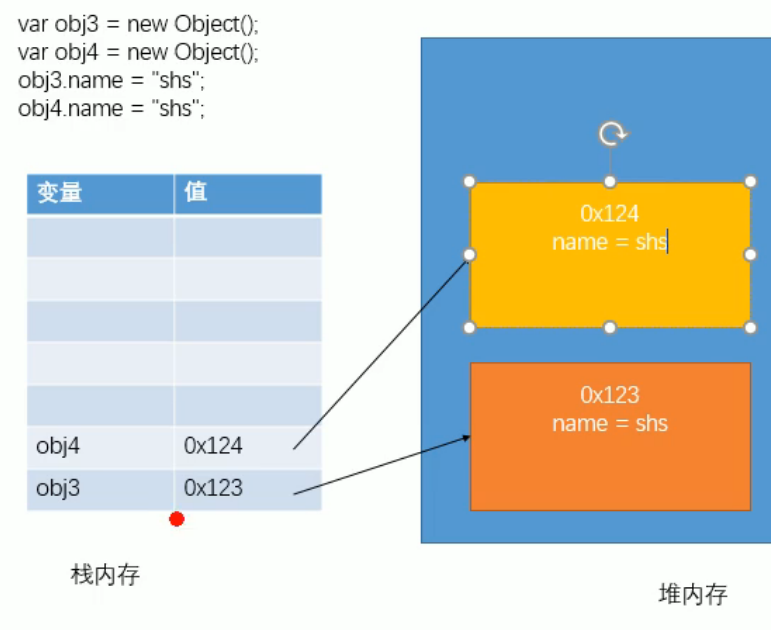


当修改一个对象的值时，对另一个对象是没有影响的。如下图所示：obj2的值变成了null，但是pbj还是有值的。



当比较两个基本数据类型时，就是比较值。

当比较两个引用数据类型时，是比较内存地址，就算两个对象一模一样，但是比较还是false



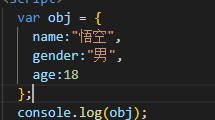
对象字面量：

使用对象字面量创建对象：

var obj = {};

使用对象字面量，可以在创建对象时，直接指定对象中的属性。对象字面量的属性名可以加引号，也可以不加，建议不加，但是需要使用一些特殊的属性名，则必须加。

属性名和属性值是一组一组的名值对结构，多个名值对用,隔开，末尾不写,



函数：函数也是一个对象，函数 中可以封装一些功能（代码）。在需要时可以执行这些功能，保存一些方法，代码。在需要时调用

创建一个函数对象：使用typeof检查一个函数对象时，会返回function

可以将要封装的代码以字符串的形式传递给构造函数。封装到函数中的代码不会立即执行。函数中的代码会在函数调用的时候执行。

var fun = new Function(“console.log(‘这是我的第一行函数’)”);

调用函数语法：函数对象()

当调用函数时，函数中封装的代码会按照顺序执行。

fun(); 执行结果为：这是我的第一个函数

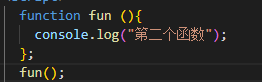
实际开发中很少使用构造函数创建一个函数对象。

实际开发中常用函数声明来创建一个函数。

语法：function 函数名([形参1，形参2，….形参n]){

语句

}

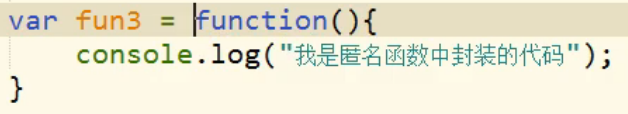


使用函数表达式来创建

var 函数名 = function([形参1，形参2，….形参n]){

语句

}

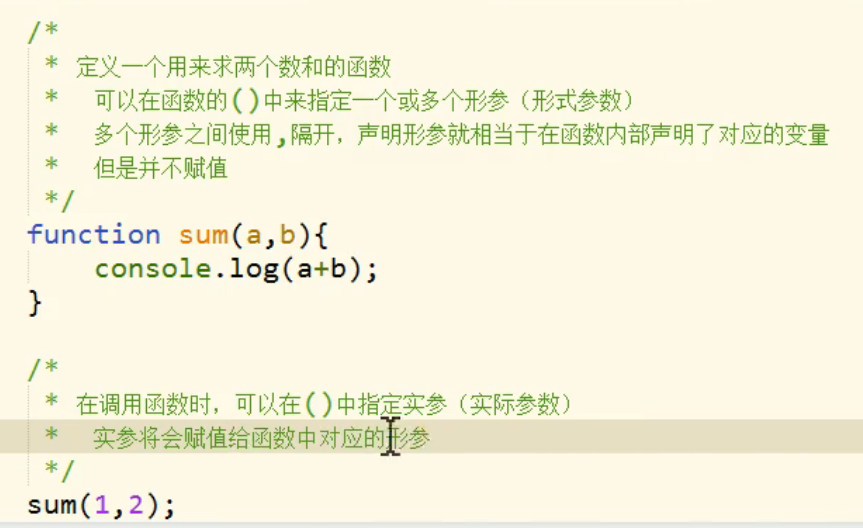


函数的参数：

调用函数时，解析器不会检查实参类型。所以要注意是否有可能接收到非法的参数，如果有可能，则需要对参数类型进行检查。

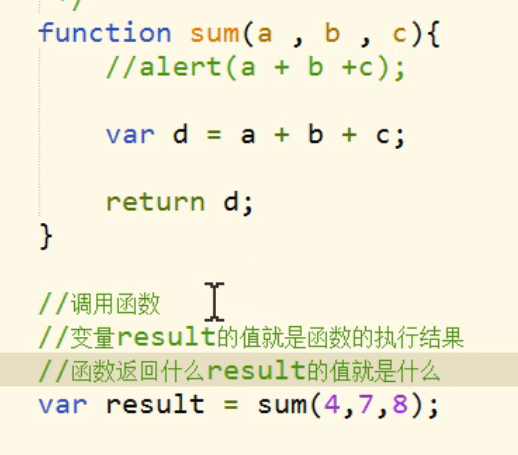
调用函数时，解析器也不会检查实参的数量。多余的实参，不会被赋值。如果实参数量少于形参数量，则没有对应实参的形参值是undefined。

函数的实参可以是任意数据类型。



函数返回值：

return 值;



在函数中，return后的语句都不会执行。

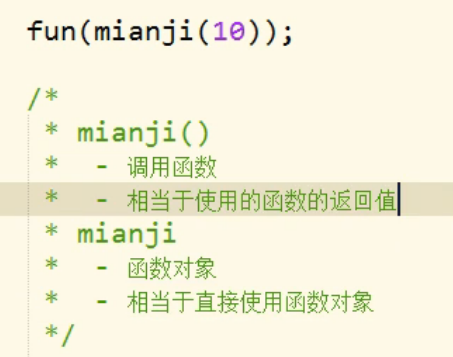
如果return后，不跟值，相当于返回undefined，如果return不写，也会返回undefined。return可以返回任意值。

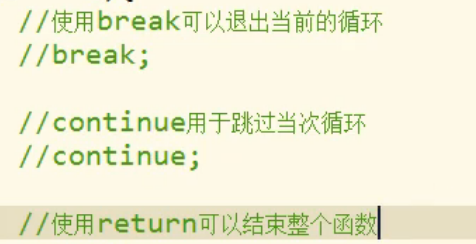
参数较多时，可以使用对象传入

方法中传入mianji和mianji（）的区别。

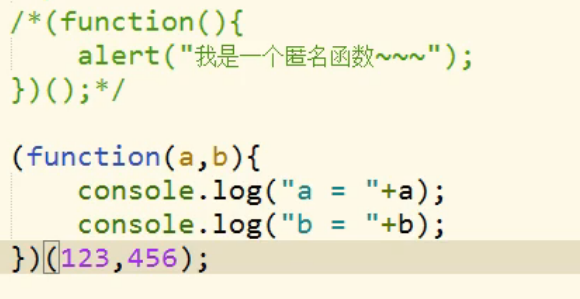
传入mianji的结果是输入这个mianji这个函数。

传入mianji(10)，输出的是这个mianji(314)这个函数的结果，也就是314

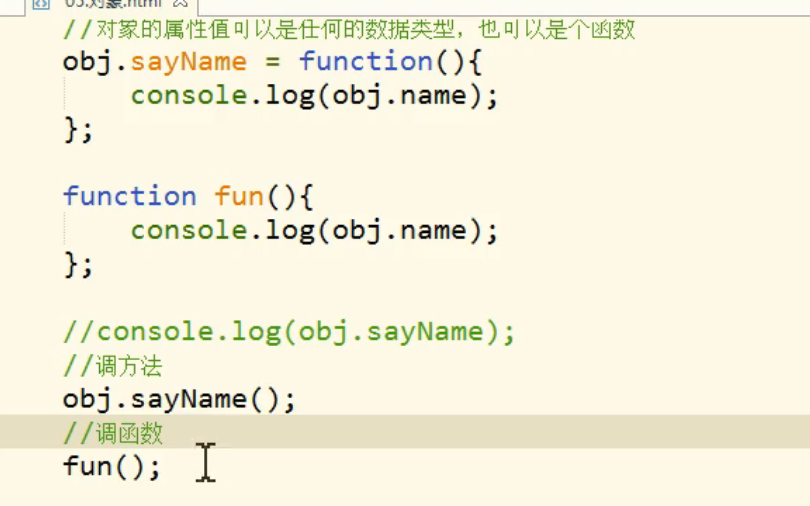




立即执行函数：函数定义完，立即被调用，只会执行一次。注意写法



调函数和调方法，只是名称上的区分，没有其他的区别。



枚举对象中的属性：

使用for … in语句

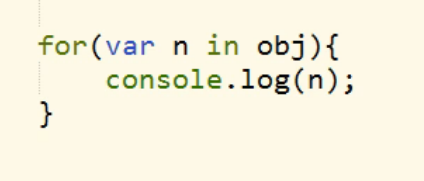
for …. in语句

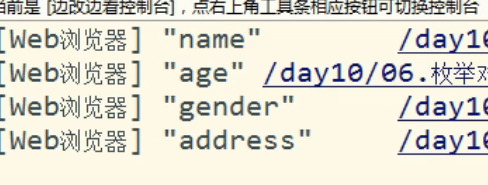
for(var 变量 in 对象){

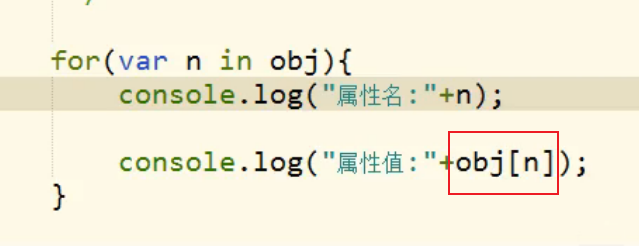
}

对象中有几个属性，循环体就会执行几次。

每次执行时，会将对象中的一个属性的名字赋值给变量







作用域：变量的作用范围

1. 全局作用域

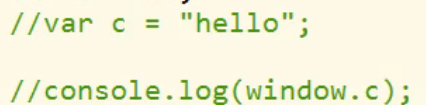
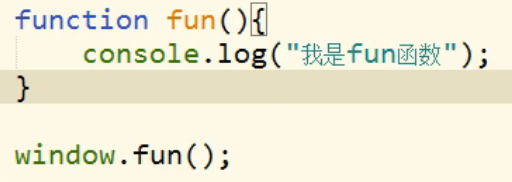
直接编写在script标签中的js代码。都在全局作用域中

作用范围：页面打开的时候创建。在页面关闭时销毁。

在页面的任意部分都可以访问到

在全局作用域中，有一个全局对象window（浏览器的窗口，由浏览器创建），我们可以直接使用。

在全局作用域中，创建的变量都会作为window对象的属性保存。创建的函数，都会作为window对象的方法保存



变量的声明提前。

--使用var关键字声明的变量，会在所有的代码执行之前被声明（但不会赋值）

但是如果声明变量时不使用var关键字，则变量不会被声明提前.

如果写了var关键字，则不会操作。console输出undefined，如果不写关键字，会报错

console.log(“a=”+a);

var a = 123;

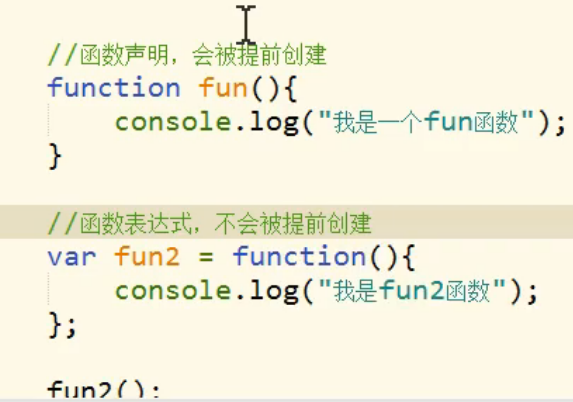
函数的声明提前：

--使用函数声明形式创建的函数function 函数（）{}

它会在所有的代码执行之前被创建，所以可以在函数声明前调用这个函数

--使用函数表达式创建的函数，不会被声明提前，所以不能在声明前调用

了解即可



1. 函数作用域：

调用函数时创建函数作用域，函数执行完毕以后，函数作用域销毁

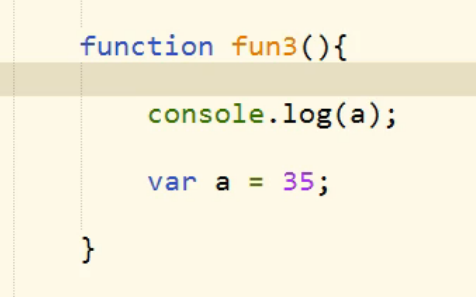
每调用一次函数，就会创建一个新的函数作用域，他们之间是互相独立的。

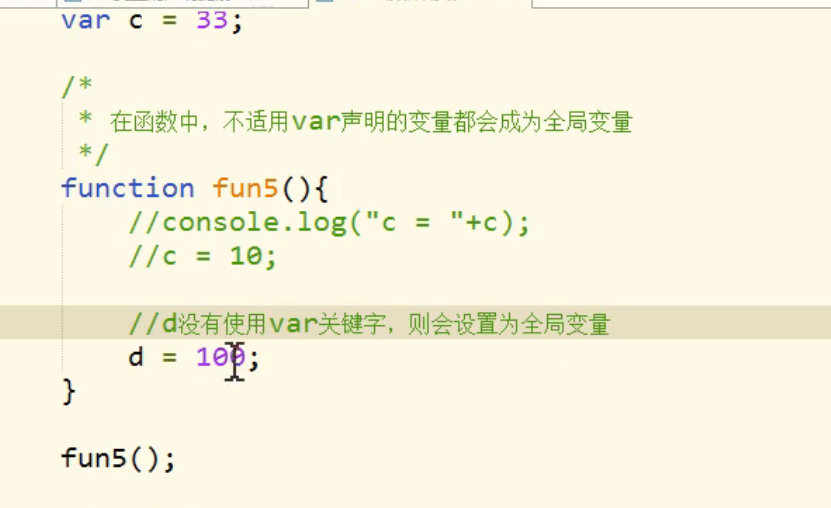
在函数作用域中，可以访问全局作用域的变量，在全局作用域中，无法访问到函数作用域的变量。

在函数作用域中操作变量时，会先在自身作用域范围中寻找，如果有，直接使用，如果没有，则向上一级作用域寻找。直到全局作用域，如果全局作用域中也没找到，则报错在函数中，需要调用全局变量，可以使用window.变量名

在函数作用域中，也有声明提前的特性。函数声明也会在函数中所有代码执行之前提前。

函数中，不使用var声明的变量，都会成为全局变量。





61