HTML + CSS + JavaScript学习笔记总结

郑素华 计机153 201510224831

目 录

[1 HTML 4](#_Toc511569726)

[1.1基础 4](#_Toc511569727)

[1.2元素 4](#_Toc511569728)

[1.3属性 4](#_Toc511569729)

[1.4标题title 5](#_Toc511569730)

[1.5段落 5](#_Toc511569731)

[1.6文本格式化 5](#_Toc511569732)

[1.7链接 5](#_Toc511569733)

[1.8头部 5](#_Toc511569734)

[1.9 CSS 5](#_Toc511569735)

[1.10图像 6](#_Toc511569736)

[1.11表格 6](#_Toc511569737)

[1.12列表 9](#_Toc511569738)

[1.13区块 9](#_Toc511569739)

[1.14表单 9](#_Toc511569740)

[1.15框架 11](#_Toc511569741)

[1.16脚本 11](#_Toc511569742)

[1.17字符实体 11](#_Toc511569743)

[1.18 URL 12](#_Toc511569744)

[2. CSS 12](#_Toc511569745)

[2.1 id与class选择器 12](#_Toc511569746)

[2.2 CSS创建 12](#_Toc511569747)

[2.3 background 13](#_Toc511569748)

[2.4文本格式 13](#_Toc511569749)

[2.5字体 14](#_Toc511569750)

[2.6 CSS链接 15](#_Toc511569751)

[2.7样式列表 15](#_Toc511569752)

[2.8表格 15](#_Toc511569753)

[2.9盒子模型 16](#_Toc511569754)

[2.10边框 16](#_Toc511569755)

[2.11 CSS轮廓（outline）：是绘制于元素周围的一条线，位于边框边缘的外围，可起到突出元素的作用 17](#_Toc511569756)

[2.12 margin（外边距） 17](#_Toc511569757)

[2.13padding（内边距） 18](#_Toc511569758)

[2.14尺寸 18](#_Toc511569759)

[2.15 display（显示）与visibility（可见性） 18](#_Toc511569760)

[2.16 position定位 18](#_Toc511569761)

[2.17 float浮动 19](#_Toc511569762)

[2.18布局 19](#_Toc511569763)

[2.19组合选择符 20](#_Toc511569764)

[2.20伪类 22](#_Toc511569765)

[2.21伪元素 25](#_Toc511569766)

[3. JavaScript 25](#_Toc511569767)

[3.1输出 25](#_Toc511569768)

[3.2语法 25](#_Toc511569769)

[3.3语句 27](#_Toc511569770)

[3.4变量 28](#_Toc511569771)

[3.5数据类型 29](#_Toc511569772)

[3.6对象 30](#_Toc511569773)

[3.7函数：是由事件驱动的或者当它被调用时执行的可重复使用的代码块 31](#_Toc511569774)

[3.8作用域 31](#_Toc511569775)

[3.9事件：是发生在HTML元素的事情 31](#_Toc511569776)

[3.10字符串：用于存储和处理文本 32](#_Toc511569777)

[3.11运算符 32](#_Toc511569778)

[3.12比较 32](#_Toc511569779)

[3.13条件语句 32](#_Toc511569780)

[3.14 switch语句：基于不同的条件来执行不同的动作 32](#_Toc511569781)

[3.15 for循环：可以将代码块执行指定的次数 32](#_Toc511569782)

[3.16 while：当指定条件为true，循环就可以一直执行代码块 33](#_Toc511569783)

[3.17 break和continue 33](#_Toc511569784)

[3.18 typeof：来检测变量的数据类型 33](#_Toc511569785)

[3.19类型转换 33](#_Toc511569786)

[3.20正则表达式（Regular Expression）：是由一个字符序列形成的搜索模式 36](#_Toc511569787)

[3.21错误 42](#_Toc511569788)

[3.22变量提升 42](#_Toc511569789)

[3.23严格模式 42](#_Toc511569790)

[3.24使用误区 44](#_Toc511569791)

[3.25表单验证 46](#_Toc511569792)

# 1 HTML

## 1.1基础

### 1.1.1标题

HTML的标题有<h1>~<h6>

### 1.1.2段落

<p></p>

### 1.1.3链接

<a href = “#”></a>

1.1.4图像

<img src = “图像的路径” alt = “”/>

1.2元素

（1）不要忘记结束标签

（2）HTML空元素：没有内容的HTML元素，空元素是在开始标签中关闭的，如<br>

（3）在XHTML、XML以及未来版本的HTML中，所有元素都必须被关闭

（4）即使<br>在所有浏览器中都是有效的，但使用<br/>是更长远的保障

（5）HTML标签对大小写不敏感，但推荐使用小写标签

1.3属性

（1）属性总是以名称/值对的形式出现

（2）常见属性

|  |  |
| --- | --- |
| **属性** | **描述** |
| class | 为html元素定义一个或多个类名（classname）(类名从样式文件引入) |
| id | 定义元素的唯一id |
| style | 规定元素的行内样式（inline style） |
| title | 描述了元素的额外信息 (作为工具条使用) |

（3） class属性可以多用class = “”(引号里面可以填入多个class属性)

Id属性只能单独设置id = “”（只能填写一个，多个无效）

1.4标题title

（1）用户可以通过标题来快速浏览网页，

（2）<hr>标签在页面中创建水平线

（2）HTML注释：<!—注释内容-->

1.5段落

<p></P>

1.6文本格式化

（1）<b>：粗体

（2）<i>：斜体

（3）通常<strong>替换加粗标签<b>来使用，<em>替换<i>标签使用，<strong>和<em>意味着要呈现的文本是很重要。

1.7链接

（1）<a href = “url”></a>

（2）Target定义被链接的文档在何处显示 \_blank (重新打开一个窗口)

1.8头部

<head></head>标签元素包含了所有头部标签元素，可以添加在头部区域的元素标签为：title, style, meta, script, link等等。

1.9 CSS

（1）可以通过以下方式把css添加到HTML中

* 内联样式- 在HTML元素中使用"style"**属性**

<p style="color:blue;margin-left:20px;">This is a paragraph.</p>

* 内部样式表 -在HTML文档头部 <head> 区域使用<style>**元素** 来包含CSS

<head>  
<style type="text/css">  
body {background-color:yellow;}  
p {color:blue;}  
</style>  
</head>

* 外部引用 - 使用外部 CSS**文件**

<head>  
<link rel="stylesheet" type="text/css" href="mystyle.css">  
</head>

Take care: 最好的方式是通过外部引用CSS文件

（2）一些样式

* font-family：字体
* font-size：文字大小
* color：字体颜色
* background-color：背景颜色
* text-align：文字对齐方式

1.10图像

（1）定义图像的语法

<img src="*url*" alt="*some\_text*">

（2）<img>是空标签，只包含属性，没有闭合标签

（3）HTML图像标签

<img> 定义图像

<map> 定义图像地图

<area> 定义图像地图中的可点区域

（4）创建图像映射

<map name="planetmap">

<area shape="rect" coords="0,0,82,126" alt="Sun" href="sun.htm">

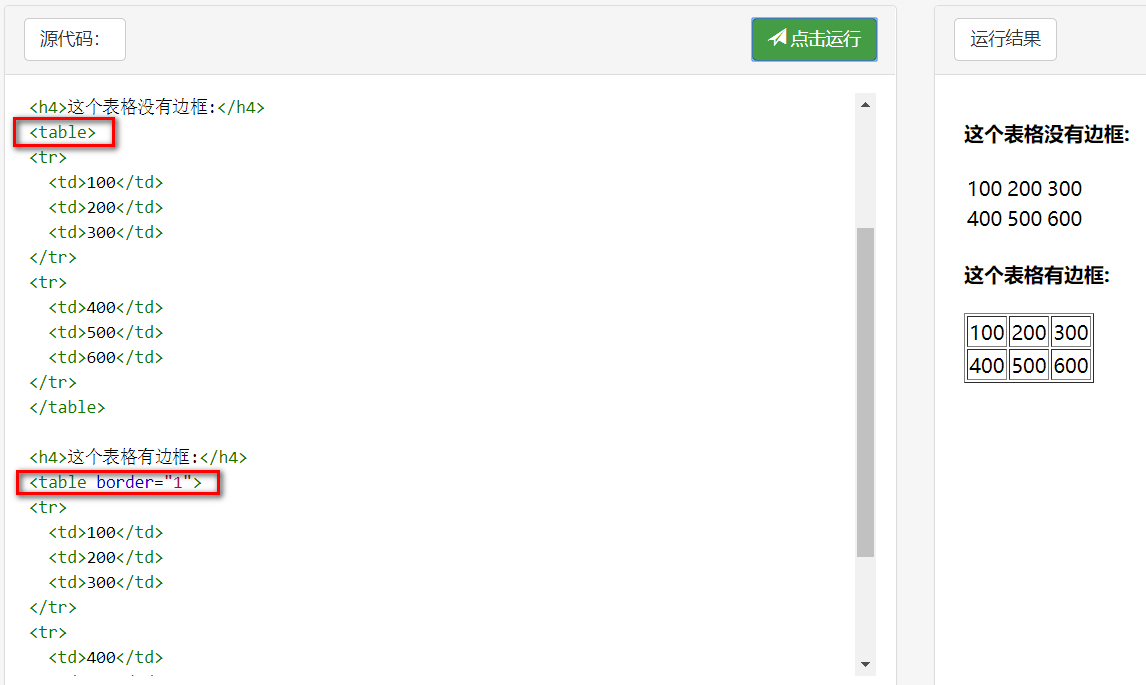
<area shape="circle" coords="90,58,3" alt="Mercury" href="mercur.htm">

<area shape="circle" coords="124,58,8" alt="Venus" href="venus.htm">

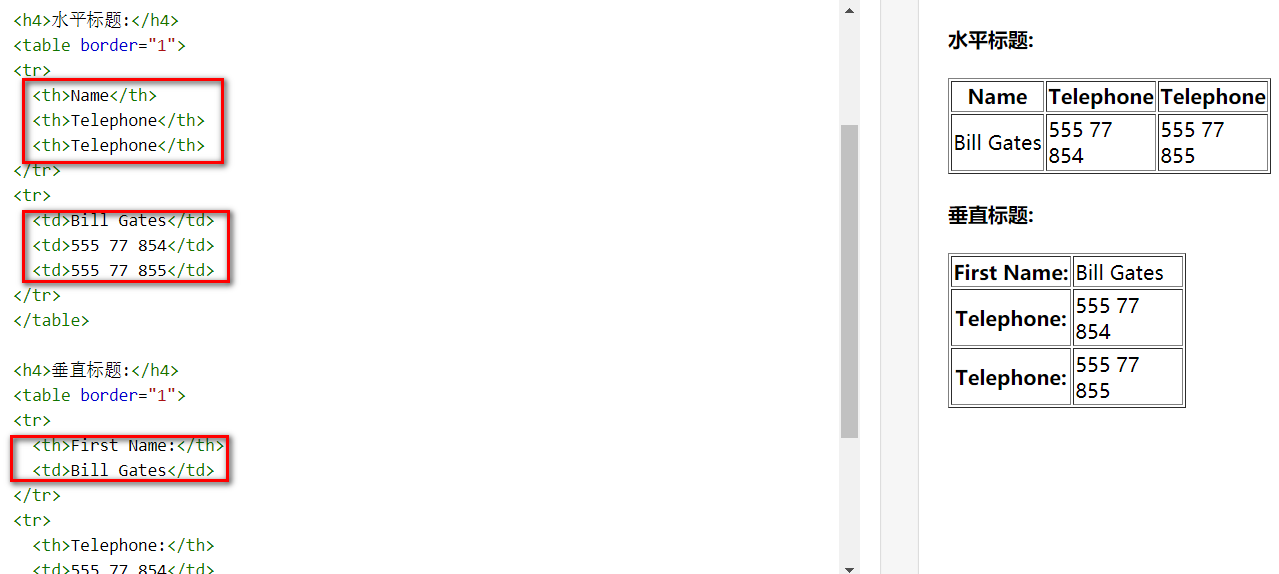
</map>

1.11表格

（1）没有边框的表格



（2）表格中的表头



（3）带标题的表格

<caption></caption>



（4）跨行或跨列的表格单元格

colspan = “2” 单元格跨两格

rowspan = “2” 单元格跨两列

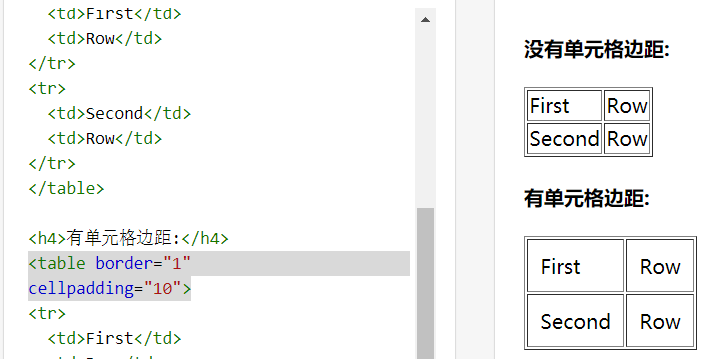


（5）表格内的标签



（6）单元格边距

cellpadding



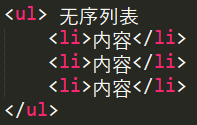
（7）单元格间距

Cellspaceing

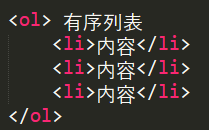


1.12列表

（1）无序列表



（2）有序列表



1.13区块

（1）块级元素：在浏览器显示时，通常会以新行来开始（和结束）

Eg：<div>、<h1>、<p>、<ul>、<table>

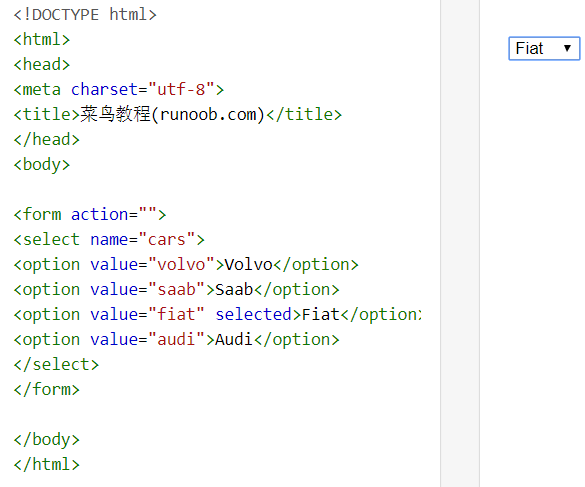
（2）内联元素：不会以新行来开始

Eg：<b>、<td>、<a>、<im>

1.14表单

<form>  
*input 元素*  
</form>

1. 文本域<input type="text">
2. 密码<input type="password">
3. 单选按钮<input type="radio">
4. 复选框<input type="checkbox">
5. 提交按钮<input type="submit">
6. 下拉框



|  |  |
| --- | --- |
| **标签** | **描述** |
| [<form>](http://www.runoob.com/tags/tag-form.html) | 定义供用户输入的表单 |
| [<input>](http://www.runoob.com/tags/tag-input.html) | 定义输入域 |
| [<textarea>](http://www.runoob.com/tags/tag-textarea.html) | 定义文本域 (一个多行的输入控件) |
| [<label>](http://www.runoob.com/tags/tag-label.html) | 定义了 <input> 元素的标签，一般为输入标题 |
| [<fieldset>](http://www.runoob.com/tags/tag-fieldset.html) | 定义了一组相关的表单元素，并使用外框包含起来 |
| [<legend>](http://www.runoob.com/tags/tag-legend.html) | 定义了 <fieldset> 元素的标题 |
| [<select>](http://www.runoob.com/tags/tag-select.html) | 定义了下拉选项列表 |
| [<optgroup>](http://www.runoob.com/tags/tag-optgroup.html) | 定义选项组 |
| [<option>](http://www.runoob.com/tags/tag-option.html) | 定义下拉列表中的选项 |
| [<button>](http://www.runoob.com/tags/tag-button.html) | 定义一个点击按钮 |
| [<datalist>](http://www.runoob.com/tags/tag-datalist.html)**New** | 指定一个预先定义的输入控件选项列表 |
| [<keygen>](http://www.runoob.com/tags/tag-keygen.html)**New** | 定义了表单的密钥对生成器字段 |
| [<output>](http://www.runoob.com/tags/tag-output.html)**New** | 定义一个计算结果 |

1.15框架

<iframe src="https://www.baidu.com"></iframe>

1. heiht和width属性用来定义iframe的高度与宽度
2. frameborder用于定义iframe表示是否显示边框
3. 使用iframe来显示目标链接页面

1.16脚本

（1）. <script> 定义了客户端脚本。

（2）. <noscript> 应对不支持脚本或禁用脚本的浏览器，只有在浏览器不支持脚本或者禁用脚本时，才会显示<noscript>元素中的内容。

1.17字符实体

实体名称对大小写敏感

|  |  |  |  |
| --- | --- | --- | --- |
| **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; | &#162; |
| £ | 镑 | &pound; | &#163; |
| ¥ | 人民币/日元 | &yen; | &#165; |
| € | 欧元 | &euro; | &#8364; |
| § | 小节 | &sect; | &#167; |
| © | 版权 | &copy; | &#169; |
| ® | 注册商标 | &reg; | &#174; |
| ™ | 商标 | &trade; | &#8482; |
| × | 乘号 | &times; | &#215; |
| ÷ | 除号 | &divide; | &#247; |

1.18 URL

统一资源定位器

**scheme://host.domain:port/path/filename**

协议：//主机.域名:端口号/路径/资源名称

2. CSS

2.1 id与class选择器

（1）id 选择器可以为标有特定 id 的 HTML 元素指定特定的样式

（2）class 选择器用于描述一组元素的样式，class 选择器有别于id选择器，class可以在多个元素中使用

2.2 CSS创建

（1）插入样式表的三种方法

* 外部样式表(External style sheet)

<head> <link rel="stylesheet" type="text/css" href="mystyle.css"> </head>

* 内部样式表(Internal style sheet)

<head> <style> hr {color:sienna;} p {margin-left:20px;} body {background-image:url("images/back40.gif");} </style> </head>

* 内联样式(Inline style)

<p style="color:sienna;margin-left:20px">这是一个段落。</p>

（2）多重样式优先级

内联样式）Inline style > （内部样式）Internal style sheet >（外部样式）External style sheet > 浏览器默认样式

Take care: 如果外部样式放在内部样式的后面，则外部样式将覆盖内部样式

（3）！important规则

当 !important 规则被应用在一个样式声明中时,该样式声明会覆盖CSS中任何其他的声明, 无论它处在声明列表中的哪里. 尽管如此, !important规则还是与优先级毫无关系.使用 !important 不是一个好习惯，因为它改变了你样式表本来的级联规则，从而使其难以调试

2.3 background

Background-color

Background-image

Background-repeat

Background-attachment

Background-position定位

2.4文本格式

（1）文本颜色

十六进制值

一个RGB值

颜色的名称

（2）文本的对齐方式

text-align: center | right | justfy

（3）文本修饰

text-decoration: none | overline | line-through | underline

（4）文本转换：用来指定在一个文本中的大写和小写字母

Text-transform: uppercase | lowercase | capitalize

（5）文本缩进

text-indent

2.5字体

（1）CSS字型

通用字体系列——拥有相似外观的字体系统组合（Serif或Monospace）

特定字体系列——一个特定的字体系列（Times或Courier）

|  |  |  |
| --- | --- | --- |
| **Generic family** | **字体系列** | **说明** |
| Serif | Times New Roman Georgia | Serif字体中字符在行的末端拥有额外的装饰 |
| Sans-serif | Arial Verdana | "Sans"是指无 - 这些字体在末端没有额外的装饰 |
| Monospace | Courier New Lucida Console | 所有的等宽字符具有相同的宽度 |

（2）字体系列

font-family设置文本的字体系列

p{font-family:"Times New Roman", Times, serif;}

（3）字体样式

font-style：normal | italic | oblique

（4）字体大小

font-size设置文本的大小

（5）设置字体大小像素

为了避免Internet Explorer中无法调整文本的问题，许多开发者使用em单位代替像素

1em和当前字体大小相等

1em的默认值是16px。

（6）所有CSS字体属性

|  |  |
| --- | --- |
| **Property** | **描述** |
| [font](http://www.runoob.com/cssref/pr-font-font.html) | 在一个声明中设置所有的字体属性 |
| [font-family](http://www.runoob.com/cssref/pr-font-font-family.html) | 指定文本的字体系列 |
| [font-size](http://www.runoob.com/cssref/pr-font-font-size.html) | 指定文本的字体大小 |
| [font-style](http://www.runoob.com/cssref/pr-font-font-style.html) | 指定文本的字体样式 |
| [font-variant](http://www.runoob.com/cssref/pr-font-font-variant.html) | 以小型大写字体或者正常字体显示文本。 |
| [font-weight](http://www.runoob.com/cssref/pr-font-weight.html) | 指定字体的粗细。 |

2.6 CSS链接

链接的样式

a:link - 正常，未访问过的链接

a:visited - 用户已访问过的链接

a:hover - 当用户鼠标放在链接上时

a:active - 链接被点击的那一刻

2.7样式列表

（1）list-style-type属性指定列表标记的类型

（2）list-style-image作为列表标记的图像

（3）浏览器兼容性解决方案

ul { list-style-type: none; padding: 0px; margin: 0px; }

ul li { background-image: url(sqpurple.gif); background-repeat: no-repeat; background-position: 0px 5px; padding-left: 14px; }

（4）列表简写属性

list-style: list-style-type list-style-position list-style-image

2.8表格

（1）表格边框border

（2）折叠边框border-collapse: collapse

（3）表格宽度和高度

（4）表格宽度和高度width height

（5）表格文字对齐text-align vertical-align

2.9盒子模型

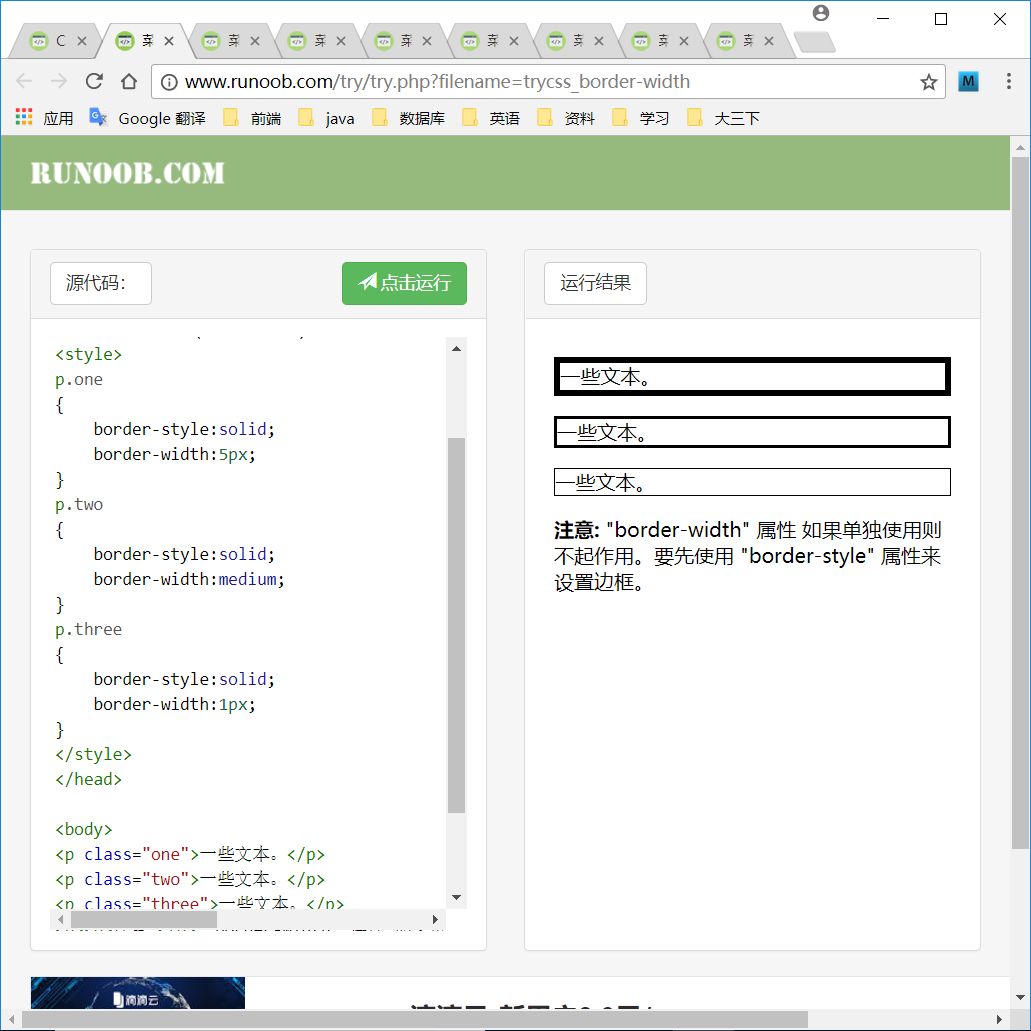
box model



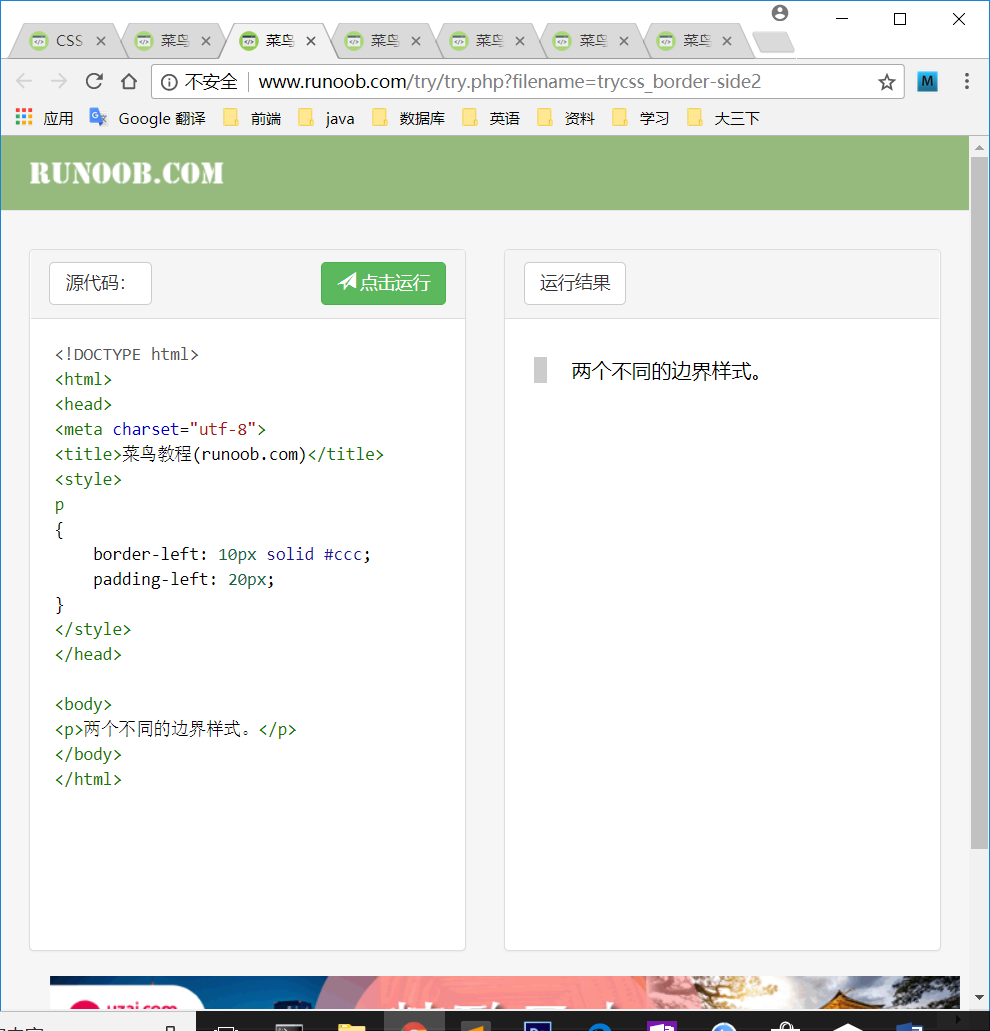
2.10边框

（1）边框样式：border-style

（2）边框宽度：border-width



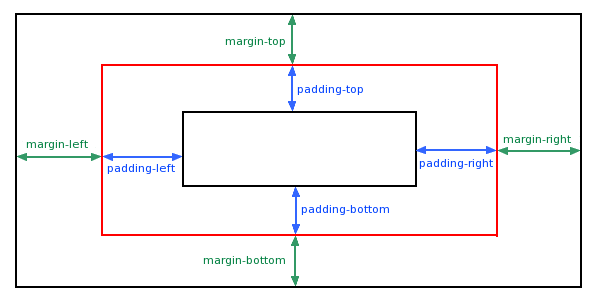
（3）边框颜色：border-color



2.11 CSS轮廓（outline）：是绘制于元素周围的一条线，位于边框边缘的外围，可起到突出元素的作用



2.12 margin（外边距）



值：auto（设置浏览器边距） length（固定的margin） %（定义一个使用百分比的边距）

2.13padding（内边距）

值：length（定义一个固定的填充）%（使用百分比定义一个填充）

2.14尺寸

(1) height

(2) line-height设置行高

(3) max-height设置元素的最大高度

(4) max-width设置元素的最大宽度

(5) min-height设置元素的最小高度

(6) min-width设置元素的最小宽度

(7) width

2.15 display（显示）与visibility（可见性）

（1）区别：display属性设置一个元素应如何显示，visibility属性指定一个元素应可见还是隐藏

（2）隐藏元素

display：none 可以隐藏某个元素，且隐藏的元素不会占用任何空间

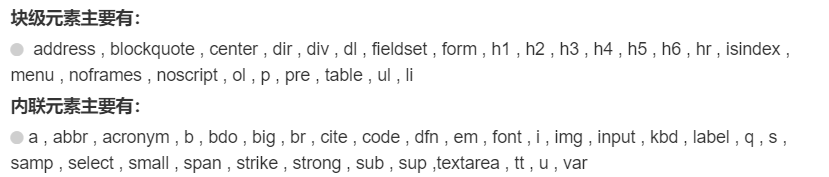
visibility：hidden隐藏的元素仍需占用与未隐藏之前一样的空间，该元素虽然被隐藏了，但仍会影响布局

（3）改变元素显示

将块级元素显示为内联元素display：inline

将行内元素显示为块级元素display: block

Display: inline-block行内块级元素



2.16 position定位

（1）position: static | relative | fixed | absolute

（2）static定位：HTML元素的默认值，没有定位，元素出现在正常流中

（3）fixed定位：元素的位置相对于浏览器窗口的固定位置，即使窗口滚动也不会移动（不占据空间，和其他元素重叠）

（4）relative定位：可以移动相对元素的内容和相互重叠的元素，他原本所占空间不会改变

2.17 float浮动

（1）flaot: left | right

（2）清除浮动 Clear: both

（3）所有的浮动属性

|  |  |  |  |
| --- | --- | --- | --- |
| **属性** | **描述** | **值** | **CSS** |
| [clear](http://www.runoob.com/cssref/pr-class-clear.html) | 指定不允许元素周围有浮动元素。 | left right both none inherit | 1 |
| [float](http://www.runoob.com/cssref/pr-class-float.html) | 指定一个盒子（元素）是否可以浮动。 | left right none inherit | 1 |

2.18布局

（1）元素居中对齐margin: auto

Take care: 如果没有设置width，居中对齐将不起作用

（2）文本居中对齐 text-align: center

（3）图片居中对齐margin: auto（将它放到块元素中）

（4）左右对齐-使用定位方式

左对齐： position: absolute; left: 0;

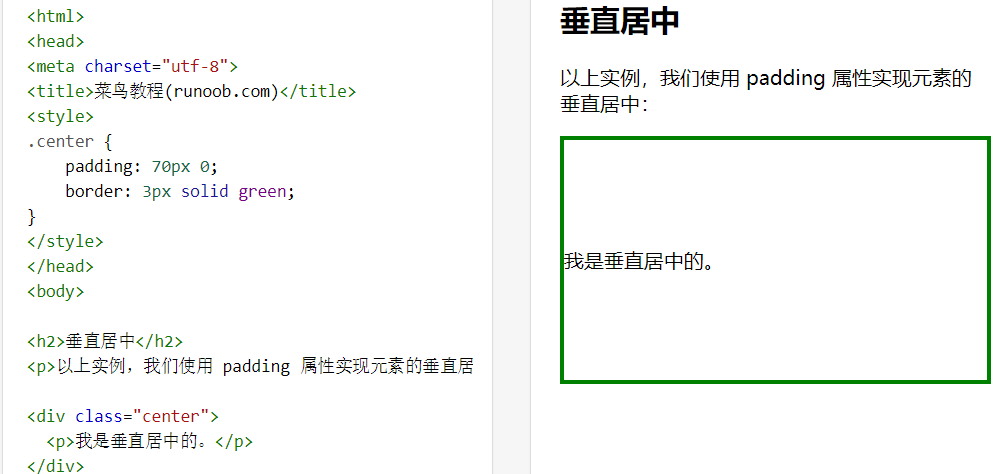
右对齐： position: absolute; right: 0;

（5）左右对齐-使用浮动方式

左对齐：float: left;

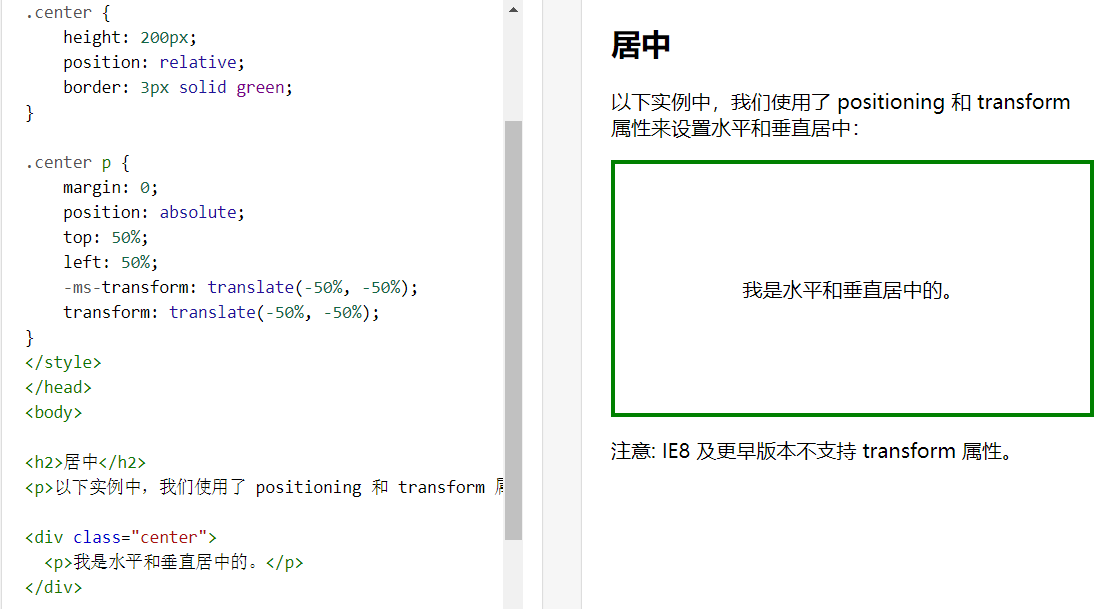
右对齐：float: right

（6）垂直居中对齐-使用padding



（7）垂直居中对齐-使用line-height

（8）垂直居中对齐-使用position和transform

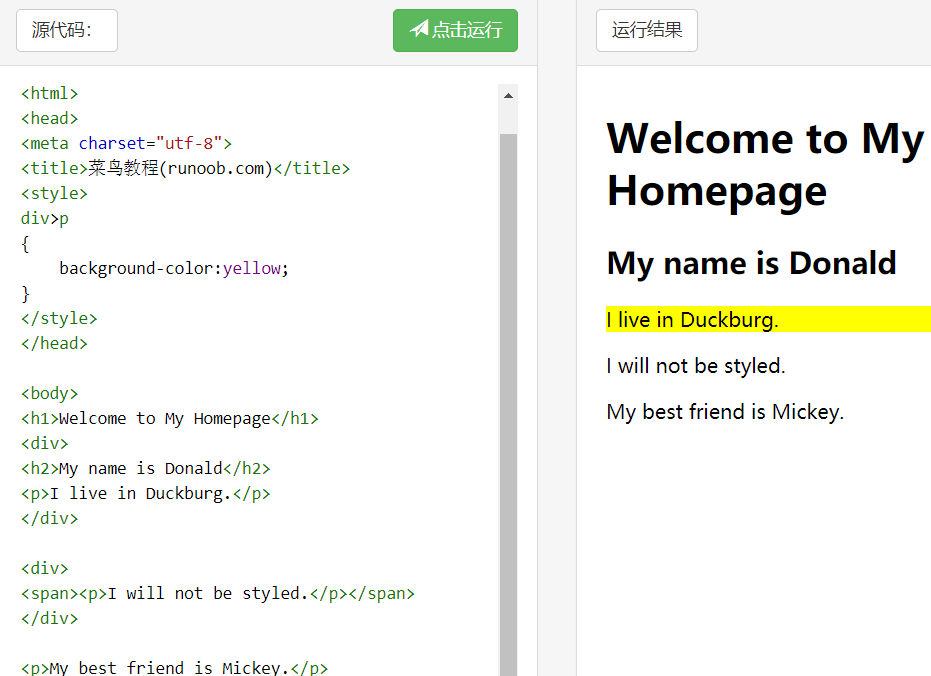


2.19组合选择符

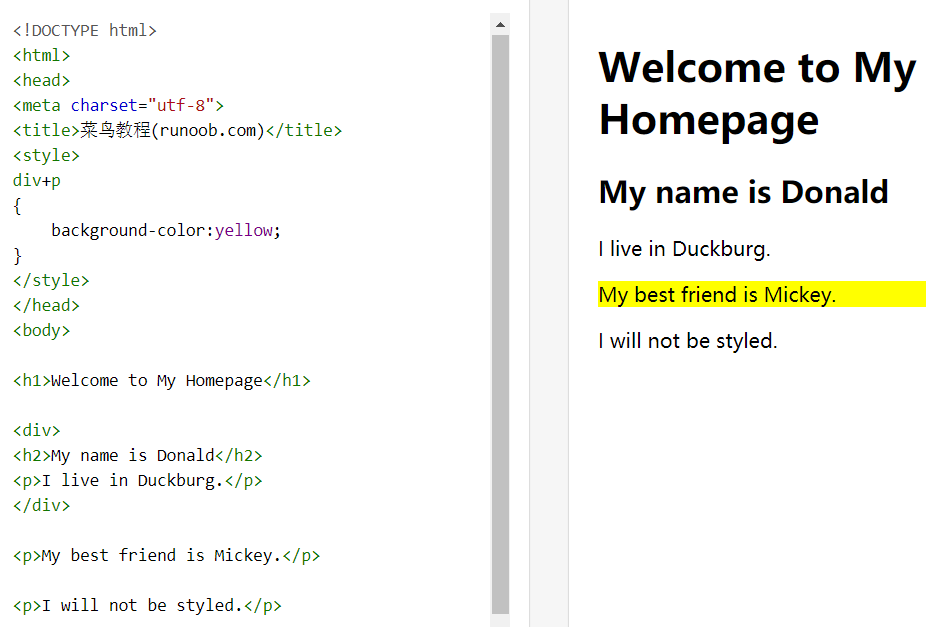
CSS3包含了四种组合方式：

后代选择器（以空格分隔）div p

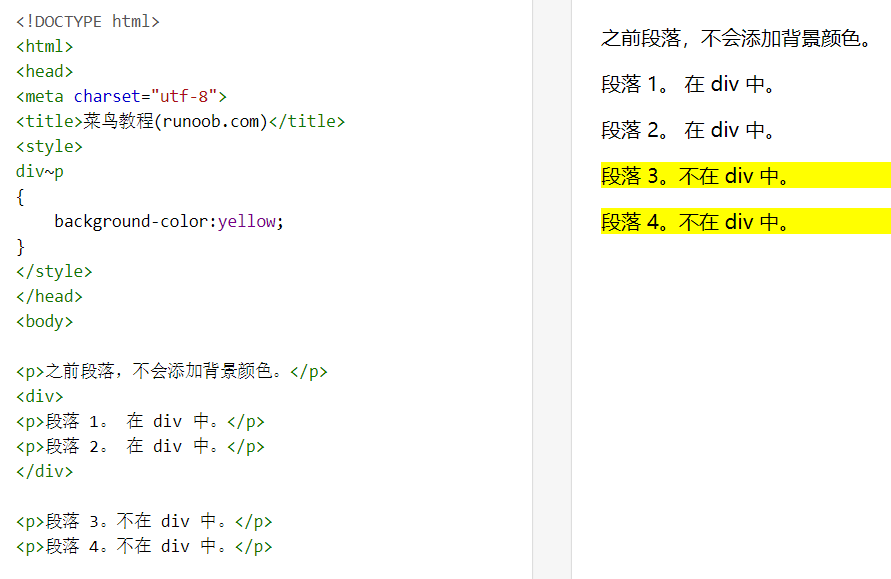
子元素选择器（以大于后分隔）div > p



相邻兄弟选择器（以加号分隔）div+p



普通兄弟选择器（以破折号分隔）div~p



2.20伪类

（1）anchor伪类

a:link {color:#FF0000;} /\* 未访问的链接 \*/

a:visited {color:#00FF00;} /\* 已访问的链接 \*/

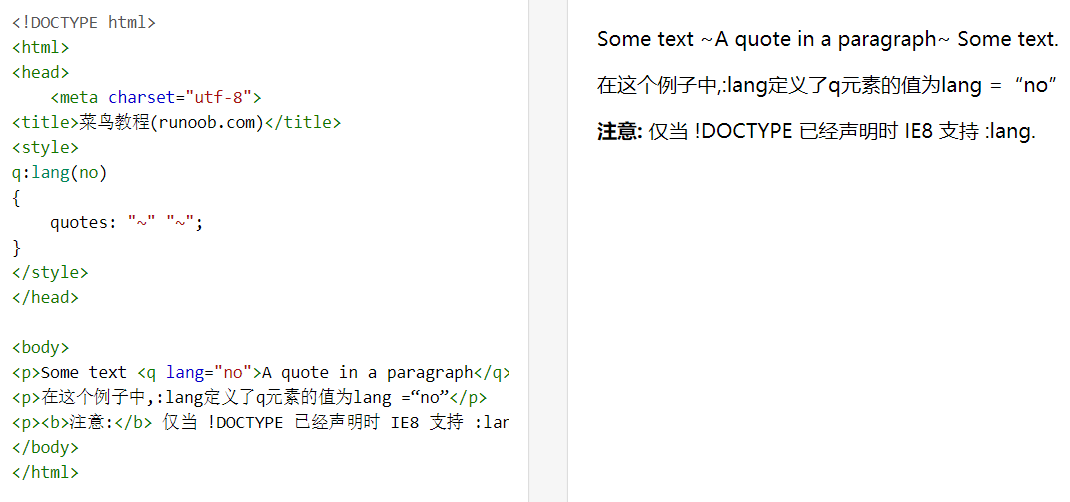
a:hover {color:#FF00FF;} /\* 鼠标划过链接 \*/

a:active {color:#0000FF;} /\* 已选中的链接 \*/

注意：a:hover必须置于a:link和a:visited之后才是有效的，a:active必须置于a:hover之后才是有效的

（2）css-:first-child伪类

（3）css-:lang伪类



（4）所有css伪类

|  |  |  |
| --- | --- | --- |
| **选择器** | **示例** | **示例说明** |
| [:checked](http://www.runoob.com/cssref/sel-checked.html) | input:checked | 选择所有选中的表单元素 |
| [:disabled](http://www.runoob.com/css/cssref/sel-disabled.html) | input:disabled | 选择所有禁用的表单元素 |
| [:empty](http://www.runoob.com/cssref/sel-empty.html) | p:empty | 选择所有没有子元素的p元素 |
| [:enabled](http://www.runoob.com/cssref/sel-enable.html) | input:enabled | 选择所有启用的表单元素 |
| [:first-of-type](http://www.runoob.com/cssref/sel-first-of-type.html) | p:first-of-type | 选择每个父元素是p元素的第一个p子元素 |
| [:in-range](http://www.runoob.com/cssref/sel-in-range.html) | input:in-range | 选择元素指定范围内的值 |
| [:invalid](http://www.runoob.com/cssref/sel-invalid.html) | input:invalid | 选择所有无效的元素 |
| [:last-child](http://www.runoob.com/cssref/sel-last-child.html) | p:last-child | 选择所有p元素的最后一个子元素 |
| [:last-of-type](http://www.runoob.com/cssref/sel-last-of-type.html) | p:last-of-type | 选择每个p元素是其母元素的最后一个p元素 |
| [:not(selector)](http://www.runoob.com/cssref/sel-not.html) | :not(p) | 选择所有p以外的元素 |
| [:nth-child(n)](http://www.runoob.com/cssref/sel-nth-child.html) | p:nth-child(2) | 选择所有p元素的第二个子元素 |
| [:nth-last-child(n)](http://www.runoob.com/cssref/sel-nth-last-child.html) | p:nth-last-child(2) | 选择所有p元素倒数的第二个子元素 |
| [:nth-last-of-type(n)](http://www.runoob.com/cssref/sel-nth-last-of-type.html) | p:nth-last-of-type(2) | 选择所有p元素倒数的第二个为p的子元素 |
| [:nth-of-type(n)](http://www.runoob.com/cssref/sel-nth-of-type.html) | p:nth-of-type(2) | 选择所有p元素第二个为p的子元素 |
| [:only-of-type](http://www.runoob.com/cssref/sel-only-of-type.html) | p:only-of-type | 选择所有仅有一个子元素为p的元素 |
| [:only-child](http://www.runoob.com/cssref/sel-only-child.html) | p:only-child | 选择所有仅有一个子元素的p元素 |
| [:optional](http://www.runoob.com/cssref/sel-optional.html) | input:optional | 选择没有"required"的元素属性 |
| [:out-of-range](http://www.runoob.com/cssref/sel-out-of-range.html) | input:out-of-range | 选择指定范围以外的值的元素属性 |
| [:read-only](http://www.runoob.com/cssref/sel-read-only.html) | input:read-only | 选择只读属性的元素属性 |
| [:read-write](http://www.runoob.com/cssref/sel-read-write.html) | input:read-write | 选择没有只读属性的元素属性 |
| [:required](http://www.runoob.com/cssref/sel-required.html) | input:required | 选择有"required"属性指定的元素属性 |
| [:root](http://www.runoob.com/cssref/sel-root.html) | root | 选择文档的根元素 |
| [:target](http://www.runoob.com/cssref/sel-target.html) | #news:target | 选择当前活动#news元素(点击URL包含锚的名字) |
| [:valid](http://www.runoob.com/cssref/sel-valid.html) | input:valid | 选择所有有效值的属性 |
| [:link](http://www.runoob.com/cssref/sel-link.html) | a:link | 选择所有未访问链接 |
| [:visited](http://www.runoob.com/cssref/sel-visited.html) | a:visited | 选择所有访问过的链接 |
| [:active](http://www.runoob.com/cssref/sel-active.html) | a:active | 选择正在活动链接 |
| [:hover](http://www.runoob.com/cssref/sel-hover.html) | a:hover | 把鼠标放在链接上的状态 |
| [:focus](http://www.runoob.com/cssref/sel-focus.html) | input:focus | 选择元素输入后具有焦点 |
| [:first-letter](http://www.runoob.com/cssref/sel-firstletter.html) | p:first-letter | 选择每个<p> 元素的第一个字母 |
| [:first-line](http://www.runoob.com/cssref/sel-firstline.html) | p:first-line | 选择每个<p> 元素的第一行 |
| [:first-child](http://www.runoob.com/cssref/sel-firstchild.html) | p:first-child | 选择器匹配属于任意元素的第一个子元素的 <p> 元素 |
| [:before](http://www.runoob.com/cssref/sel-before.html) | p:before | 在每个<p>元素之前插入内容 |
| [:after](http://www.runoob.com/cssref/sel-after.html) | p:after | 在每个<p>元素之后插入内容 |
| [:lang(*language*)](http://www.runoob.com/cssref/sel-lang.html) | p:lang(it) | 为<p>元素的lang属性选择一个开始值 |

2.21伪元素

（1）:first-line用于向文本的首行设置特殊样式 只能用于块级元素

（2）:first-letter用于向文本的首字母设置特殊样式 只能用于块级元素

（3）:before 可以在元素的内容前面插入新内容

（4）:after 可以在元素的内容之后插入新内容

3. JavaScript



3.1输出

显示数据

Window.alert()——弹出警告框

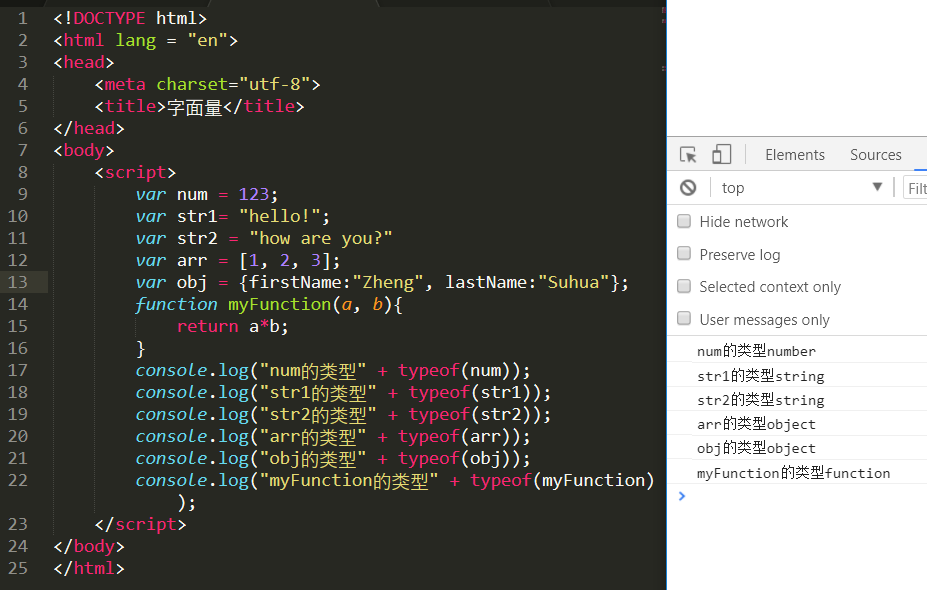
Document.write()——将内容写到HTML文档中

innerHTML——写入到HTML元素

console.log()——写入到浏览器的控制台

3.2语法

（1）字面量Number、String、Array、Object、Function



（2）关键词

**var** 关键字告诉浏览器创建一个新的变量

Javascript重要的保留字（按字母顺序）

|  |  |  |  |
| --- | --- | --- | --- |
| abstract | else | instanceof | super |
|  |  |  |  |
| boolean | enum | int | switch |
|  |  |  |  |
| break | export | interface | synchronized |
|  |  |  |  |
| byte | extends | let | this |
|  |  |  |  |
| case | false | long | throw |
|  |  |  |  |
| catch | final | native | throws |
|  |  |  |  |
| char | finally | new | transient |
|  |  |  |  |
| class | float | null | true |
|  |  |  |  |
| const | for | package | try |
|  |  |  |  |
| continue | function | private | typeof |
|  |  |  |  |
| debugger | goto | protected | var |
|  |  |  |  |
| default | if | public | void |
|  |  |  |  |
| delete | implements | return | volatile |
|  |  |  |  |
| do | import | short | while |
|  |  |  |  |
| double | in | static | with |

（3）js对大小写敏感

（4）js使用Unicode字符集

3.3语句

（1）语句标识符

|  |  |
| --- | --- |
| **语句** | **描述** |
| break | 用于跳出循环。 |
| catch | 语句块，在 try 语句块执行出错时执行 catch 语句块。 |
| continue | 跳过循环中的一个迭代。 |
| do ... while | 执行一个语句块，在条件语句为 true 时继续执行该语句块。 |
| for | 在条件语句为 true 时，可以将代码块执行指定的次数。 |
| for ... in | 用于遍历数组或者对象的属性（对数组或者对象的属性进行循环操作）。 |
| function | 定义一个函数 |
| if ... else | 用于基于不同的条件来执行不同的动作。 |
| return | 退出函数 |
| switch | 用于基于不同的条件来执行不同的动作。 |
| throw | 抛出（生成）错误 。 |
| try | 实现错误处理，与 catch 一同使用。 |
| var | 声明一个变量。 |
| while | 当条件语句为 true 时，执行语句块。 |

（2）空格js会忽略多余的空格

（3）对代码进行浙行



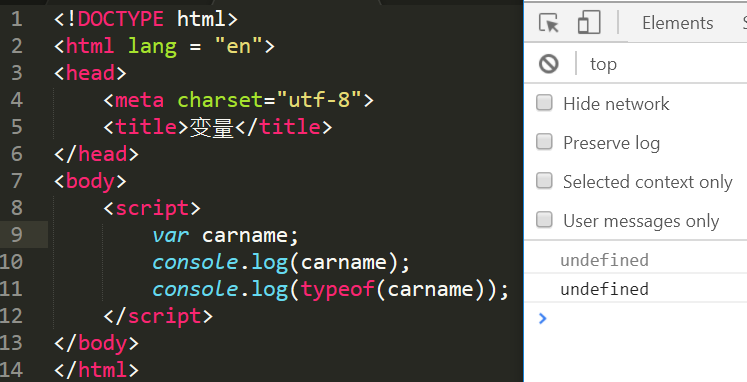
（4）JS是脚本语言，浏览器会在读取代码时，逐行执行脚本代码。对于传统编程来说，会在执行前对所有代码进行编译

3.4变量

（1）声明js变量，使用var关键词，变量声明之后，该变量是空的（没有值），赋值使用等号

（2）一个好的编程习惯是，在代码开始处，统一对需要的变量进行声明

（3）value = undefined

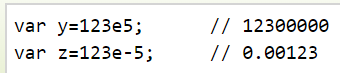


（4）重新声明js变量，该变量的值不会丢失

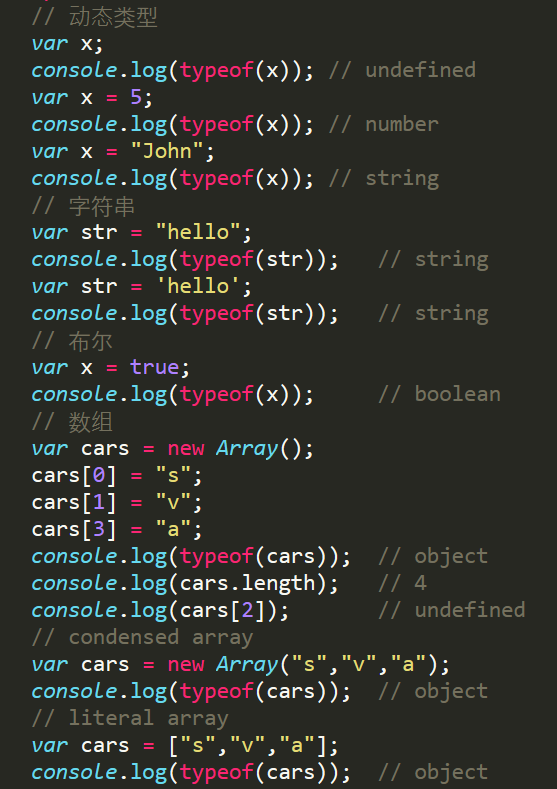
3.5数据类型

String、Number、Boolean、Array、Object、Null、Undefined

1. js拥有动态类型，相同的变量可用作不同的类型
2. 字符串：可以是引号中的任意文本，可以使用单引号或者双引号，可以在字符串中使用引号，只要不匹配包围字符串的引号即可。
3. 数字：js中只有一种数字类型，可以通过科学（指数）计数法来书写



1. 布尔：true、false
2. 数组：数组下标是基于0的



1. 对象：由花括号分隔，对象的属性以名称和值对的形式来定义



1. undefined和null

undefined表示变量不含值

可以通过将变量的值设置为null来清空变量

1. 声明变量类型：可以使用new来声明其变量，js变量均为对象，当声明一饿变量时，就创建了一个新的对象

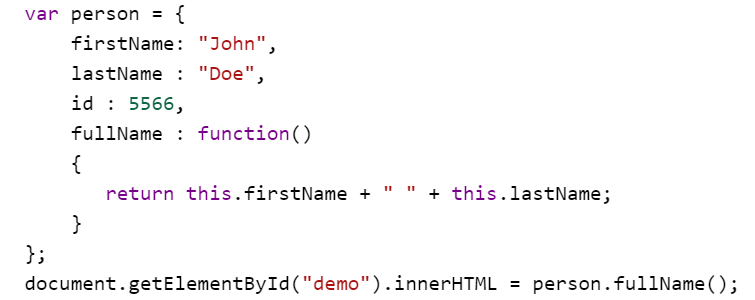
3.6对象

（1）js中，几乎所有事物都是对象，对象也是一个变量，但对象可以包含多个值

（2）定义对象可以跨越多行，空格和换行不是必须的

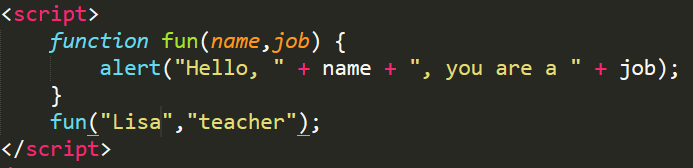
（3）对象属性：js对象是属性变量的容器

（4）对象方法



3.7函数：是由事件驱动的或者当它被调用时执行的可重复使用的代码块

（1）调用带参数的函数



（2）带有返回值的函数

通过return语句就可以实现，在使用return语句时，函数会停止执行，并返回指定的值；如果仅仅是希望退出函数时，也可以使用return语句。

（3）在函数内部声明的变量是局部变量，在函数外声明的变量是全局变量；局部变量在函数运行以后被删除，全局变量会在页面关闭后被删除。如果把值赋给尚未声明的变量，该变量将自动作为全局变量声明。如：carname = “vovo”

3.8作用域

（1）变量在函数内声明，变量为局部作用域，局部变量只能在函数内部访问

（2）变量在函数外定义，即为全局变量，全局变量有全局作用域，网页中所有脚本和函数均可使用

（3）在HTML中，全局变量是window对象，所有数据变量都属于window对象

3.9事件：是发生在HTML元素的事情

常见的HTML事件：onchange()、onclick()、onmouseover()、onmouseout()、onkeydown()、onload()

3.10字符串：用于存储和处理文本



Js中==与===区别

1）对于 string、number 等基础类型，== 和 === 是有区别的

A．不同类型间比较，== 之比较 "转化成同一类型后的值" 看 "值" 是否相等，=== 如果类型不同，其结果就是不等。

 B．同类型比较，直接进行 "值" 比较，两者结果一样。

2）对于 Array,Object 等高级类型，== 和 === 是没有区别的

进行 "指针地址" 比较

3）基础类型与高级类型，== 和 === 是有区别的

A．对于 ==，将高级转化为基础类型，进行 "值" 比较

 B．因为类型不同，=== 结果为 false

4）!= 为 == 的非运算，!== 为 === 的非运算

3.11运算符

3.12比较

3.13条件语句

3.14 switch语句：基于不同的条件来执行不同的动作

3.15 for循环：可以将代码块执行指定的次数

For – 循环代码块一定的次数

For/in – 循环遍历对象的属性

While – 当指定的条件为true时循环指定的代码块

Do/while – 同样当指定的条件为true时循环指定的指定的代码块

3.16 while：当指定条件为true，循环就可以一直执行代码块

3.17 break和continue

Break：用于跳出循环

Continue：用于跳过循环中的一个迭代

3.18 typeof：来检测变量的数据类型

（1）Null：表示什么都没有，是一个只有一个值得特殊类型，表示一个空对象引用

Typeof(null) ----- object

（2）undefined：是一个没有设置值得变量

Typeof(undefined)-----undefined

（3）undefined与null的区别：值相等，但类型不同

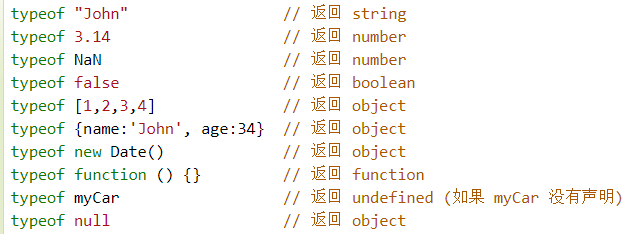
3.19类型转换

（1）js数据类型

A）5种不同的数据类型：string number Boolean object function

B）3种对象类型：Object Date Array

C）2个不包含值得数据类型：null undefined



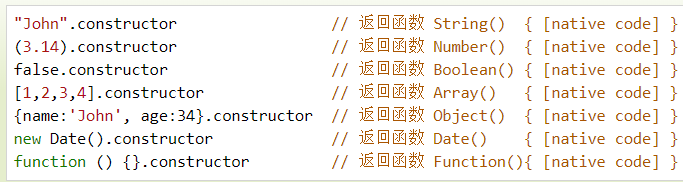
（2）take care

A）NaN的数据类型是number

B）Array、Date、null的数据类型是object

C）未定义变量的数据类型为undefined

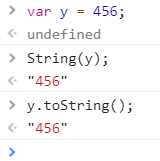
（3）constructor：返回所有js变量的构造函数



1. 将数字转换为字符串

全局方法String()

Number方法toString()



1. 将布尔值转换为字符串

全局方法String()

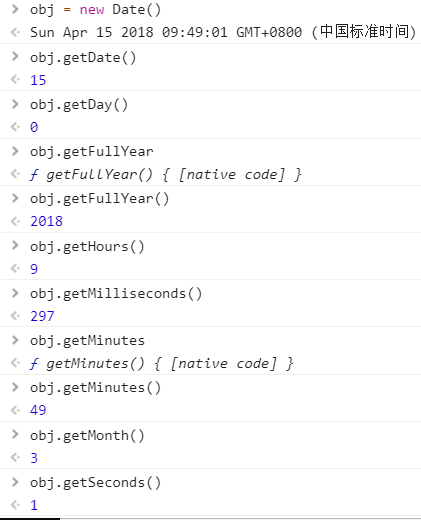
Boolean方法toString()

1. 将日期转换为字符串

Date()

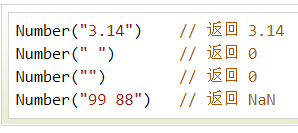
全局方法String()

Date方法toString()



1. 将字符串转换为数字

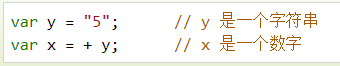
A）全局方法Number()



B）parseFloat() 解析一个字符串，并返回一个浮点数

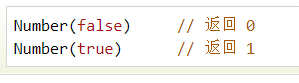
parseInt() 解析一个字符串，并返回一个整数

C）一元运算符+



1. 将布尔值转换为数字

全局方法Number()



1. 将日期转换为数字

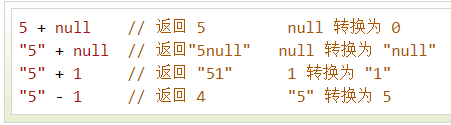
全局方法Number()



日期方法getTime()



1. 自动转换类型



3.20正则表达式（Regular Expression）：是由一个字符序列形成的搜索模式

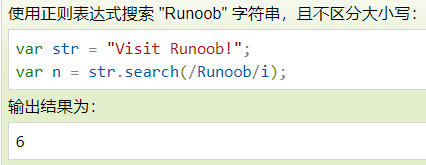
**语法**：/正则表达式主体/修饰符（可选）

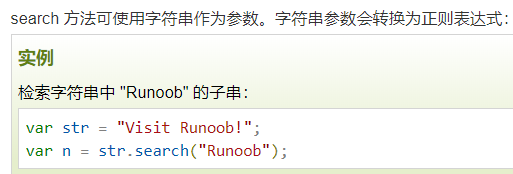
（1）正则表达式通常用于两个字符串方法：search()和replace()

**search() 方法** 用于检索字符串中指定的子字符串，或检索与正则表达式相匹配的子字符串，并返回子串的起始位置。

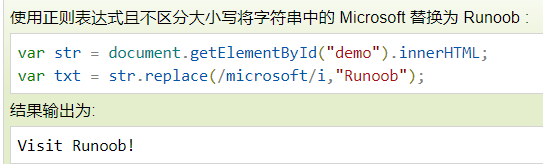
**replace() 方法** 用于在字符串中用一些字符替换另一些字符，或替换一个与正则表达式匹配的子串。

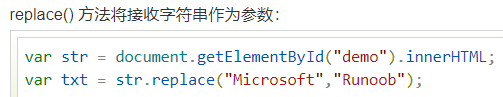
（2）search()使用正则表达式



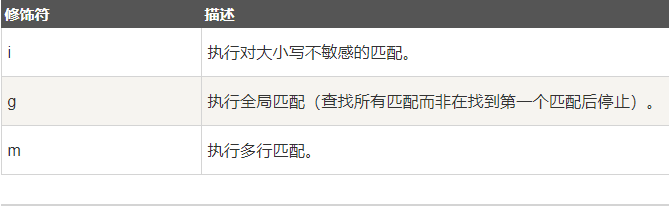


（3）replace()使用正则表达式





（4）正则表达式修饰符



（5）正则表达式模式



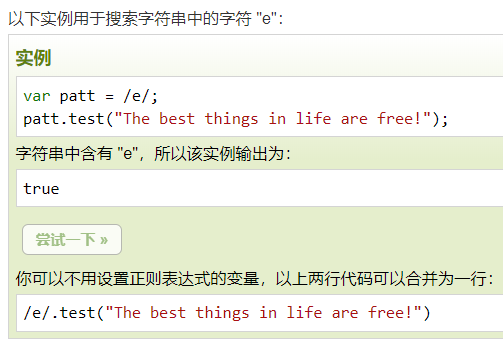




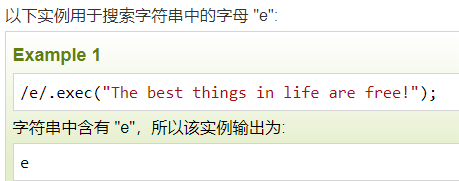
（6）使用RegExp对象：RegExp对象是一个预定义了属性和方法的正则表达式对象



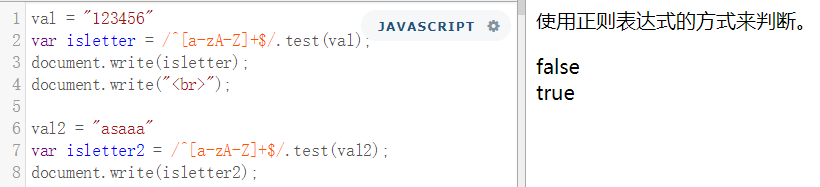
（7）使用test()：是一个正则表达式方法，用于检测一个字符串是否匹配某个模式，如果匹配，则返回true，不匹配，则返回false



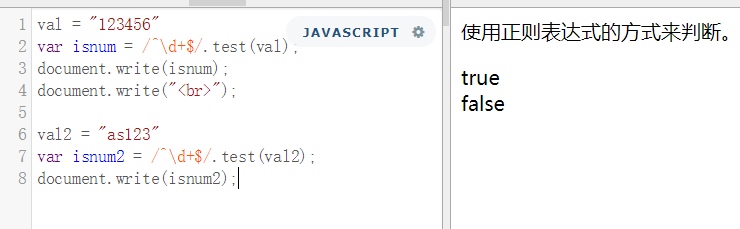
（8）使用exec()：是一个正则表达式方法，用于检索字符串中的正则表达式的匹配。该函数返回一个数组，其中存放匹配的结果，如果未找到匹配，则返回null。



1. 判断是否为字母



1. 判断是否为数字



3.21错误

Try测试代码块的错误

Catch处理错误

Throw创建自定义错误



3.22变量提升

（1）在js中，函数及变量的声明都将被提升到函数的最顶部

（2）js初始化不会提升

（3）js严格模式不允许使用未声明的变量

3.23严格模式

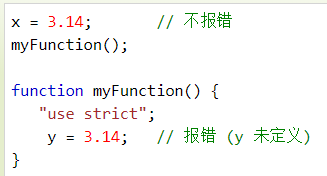
（1）使用“use strict”指令

是ES5中新增的，目的是指定代码在严格条件下执行

严格模式下你不能使用未声明的变量



（2）函数内部声明是局部作用域，只在函数内使用严格模式



（3）为什么使用严格模式

A消除js语法的一些不合理、不严谨之处、减少一些怪异行为

B "严格模式"体现了Javascript更合理、更安全、更严谨的发展方向

（4）严格模式的限制

（5）严格模式的限制

A不允许使用未声明的变量

B不允许删除变量或对象

C不允许删除函数

D不允许变量重名

E不允许使用转义字符

F不允许对只读属性赋值

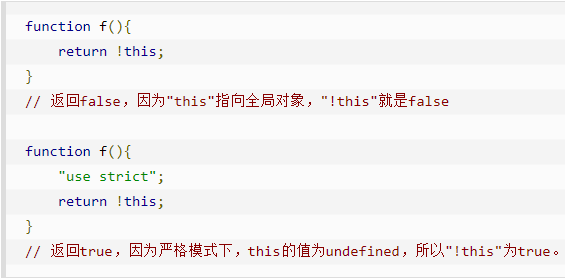
G不允许对一个使用getter()方法读取的属性进行赋值

H不允许删除一个不允许删除的属性

I变量名不能使用“eval”字符串

J变量名不能使用“arguments”字符串

K禁止this关键字指向全局对象



（6）保留关键字



3.24使用误区

（1）== 与 ===



（2）字符串断行需要使用反斜杠\



(3) return

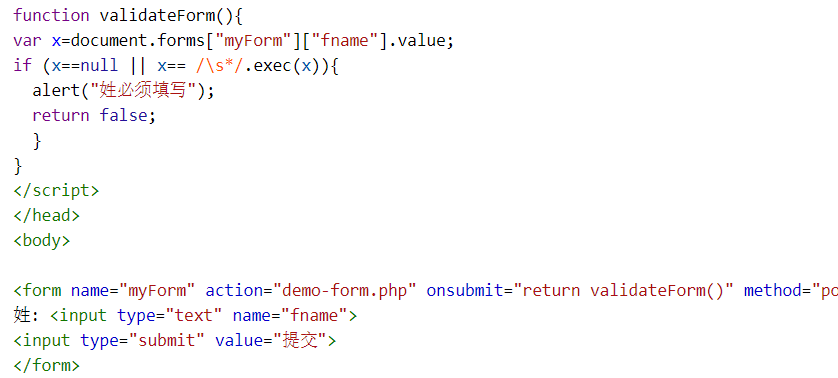


（4）在js中，分号是可选的

（5）js不支持使用名字来索引数组，只允许使用数字索引，对象使用名字作为索引

（6）null用于对象，undefined用于变量、属性和方法，对象只有被定义才有可能为null，否则为undefined

3.25表单验证



邮箱验证：



3.26验证API

（1）约束验证DOM方法

|  |  |
| --- | --- |
| **Property** | **Description** |
| checkValidity() | 如果 input 元素中的数据是合法的返回 true，否则返回 false。 |
| setCustomValidity() | 设置 input 元素的 validationMessage 属性，用于自定义错误提示信息的方法。  使用 setCustomValidity 设置了自定义提示后，validity.customError 就会变成true，则 checkValidity 总是会返回false。如果要重新判断需要取消自定义提示，方式如下：  setCustomValidity('')  setCustomValidity(null)  setCustomValidity(undefined) |

（2）约束验证DOM属性

|  |  |
| --- | --- |
| **属性** | **描述** |
| validity | 布尔属性值，返回 input 输入值是否合法 |
| validationMessage | 浏览器错误提示信息 |
| willValidate | 指定 input 是否需要验证 |

（3）validity属性

|  |  |
| --- | --- |
| **属性** | **描述** |
| customError | 设置为 true, 如果设置了自定义的 validity 信息。 |
| patternMismatch | 设置为 true, 如果元素的值不匹配它的模式属性。 |
| rangeOverflow | 设置为 true, 如果元素的值大于设置的最大值。 |
| rangeUnderflow | 设置为 true, 如果元素的值小于它的最小值。 |
| stepMismatch | 设置为 true, 如果元素的值不是按照规定的 step 属性设置。 |
| tooLong | 设置为 true, 如果元素的值超过了 maxLength 属性设置的长度。 |
| typeMismatch | 设置为 true, 如果元素的值不是预期相匹配的类型。 |
| valueMissing | 设置为 true，如果元素 (required 属性) 没有值。 |
| valid | 设置为 true，如果元素的值是合法的。 |

3.27 JSON

（1）json是用于存储和传输数据的格式，通常用于服务端向网页传递数据

（2）JSON(JavaScript Object Notation) 是一种轻量级的数据格式交换格式，是独立的语言

（3）JSON格式化后为JavaScript对象

（4）语法规则

数据为键值对

数据由逗号分隔

大括号保存对象

方括号保存数组

（5）JSON对象保存在大括号内

{"name":"Runoob", "url":"www.runoob.com"}

（6）JSON数组保存在中括号内

"sites":[

{"name":"Runoob", "url":"www.runoob.com"},

{"name":"Google", "url":"www.google.com"},

{"name":"Taobao", "url":"www.taobao.com"}

]

（7）JSON字符串转换为JavaScript对象

使用JavaScript内置函数JSON.parse()将字符串转换为JavaScript对象

Var obj = JSON.parse(text)



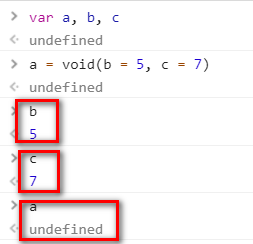
（8）Javascript值转换为JSON字符串

JSON.stringify()

3.27 JavaScript void

（1）void是JavaScript中非常重要的关键字，该操作符指定要计算一个表达式但是不返回值

（2）<a href = “javascript:void(0)”>单击此处什么也不会发生</a>



（3）href = “#”与href = “javascript:void(0)”的区别

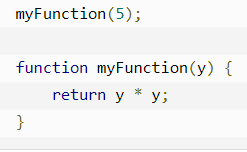
#来定位页面的具体位置，JavaScript:void(0)定义一个死链接

3.28函数

3.28.1函数定义

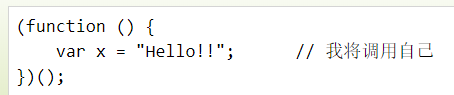
1）函数提升（Hoisting）

函数可以在声明之前调用，但是使用表达式定义函数时无法提升。

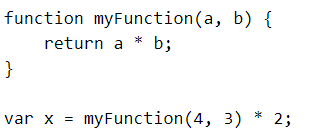


2）自调用函数

如果表达式后面紧跟()，则会自动调用；不嫩自调用声明的函数；通过添加括号，来说明它是一个函数表达式；



3）函数可作为一个值或者表达式使用



4）函数是对象

函数有属性和方法，arguments.length属性返回函数调用过程接收到的参数个数

toString()方法将函数作为一个字符串返回

3.28.2函数参数

1）函数显示参数（Parameters）与隐式参数（Arguments）

函数显式参数在函数定义时列出。

函数隐式参数在函数调用时传递给函数真正的值。

2）参数规则

函数定义显式参数没有指定数据类型；对隐式参数没有进行类型检测，对隐式参数的个数没有进行检测

3）默认参数

如果函数在调用时未提供隐式参数，参数会默认设置为：undefined

4）Arguments对象

JavaScript 函数有个内置的对象 arguments 对象。

argument 对象包含了函数调用的参数数组

5）通过值传递参数

在函数中调用的参数是函数的隐式参数。隐式参数通过值来传递，函数仅仅是获取。如果函数修改参数的值，不会修改显式参数的初始值，隐式参数的改变在函数外是不可见的。

6）通过对象传递参数

在JavaScript中，可以引用对象的值。因此我们在函数内部修改对象的属性就会修改其初始的值。修改对象属性可作用于函数外部（全局变量）。修改对象属性在函数外是可见的。

3.28.3函数调用

1）有四种调用方式，每种方式的不同在于this的初始化

2）this

This指向函数执行时的当前对象

3）作为一个函数调用



4）全局对象

当函数没有被自身的对象调用时this的值就会变成全局对象。

5）函数作为方法调用

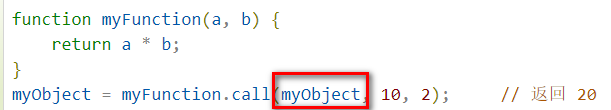
6）使用构造函数调用

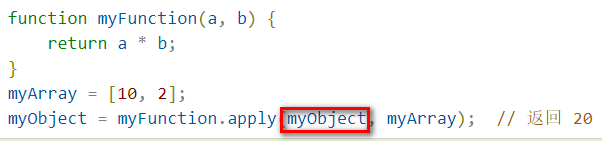
7）作为函数方法调用函数

Call()和apply()是预定义的函数方法。两个方法可用于调用函数，两个方法的第一个参数必须是对象本身。

在 JavaScript 严格模式(strict mode)下, 在调用函数时第一个参数会成为 **this** 的值， 即使该参数不是一个对象。

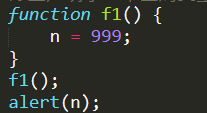
在 JavaScript 非严格模式(non-strict mode)下, 如果第一个参数的值是 null 或 undefined, 它将使用全局对象替代。



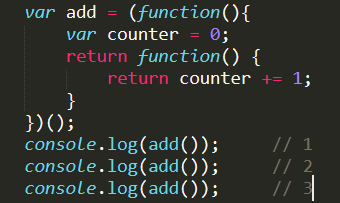


3.28.4闭包（closure）

1）变量声明时如果不使用var关键字，那么它就是一个全局变量，即便它在函数内定义



2）闭包，使得函数拥有私有变量成为可能

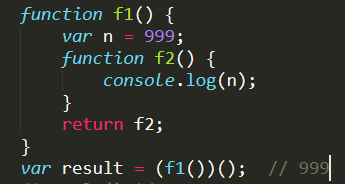


变量add指定了函数自我调用的返回值

3）链式作用域（chain scope）：子对象会一级一级地向上寻找所有父对象地变量，父对象的所有变量，对子对象都是可见的，反之则不成立。

4）理解闭包：就是能够读取其他函数内部变量的函数，定义在一个函数内部的函数。在本质上，闭包就是将函数内部和函数外部连接起来的一座桥梁。

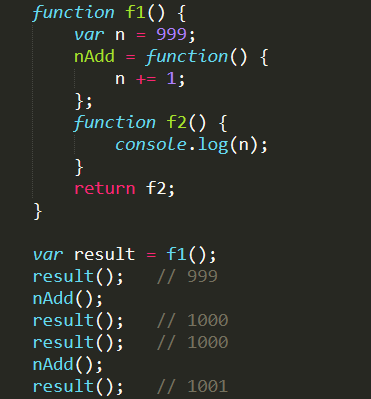
下面的f2()就是闭包



5）用途

A 读取函数内部的变量

B 让变量的值始终保持在内存中

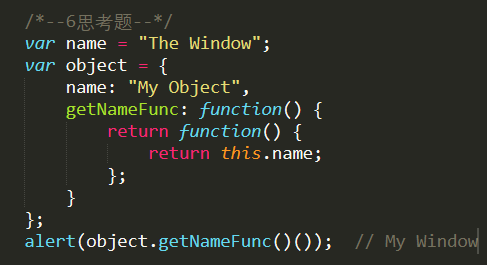


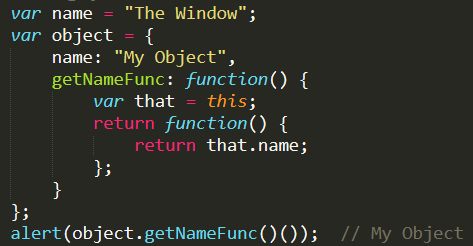
6）使用闭包的注意点

A闭包会使得函数中的变量被保存在内存中，内存消耗很大。所以，在退出内存时，需将不使用的局部变量删除。

B闭包会在父函数外部，改变父函数内部变量的值。如果把父函数当成Object使用，把闭包当作它的Public method，把内部变量当作它的Private value，这时一定要小心，不要随便改变父函数内部变量的值。

7）思考





4. JS HTML DOM

4.1 DOM

1）HTML DOM树



2）查找HTML

A通过id找到HTML元素：getElementById

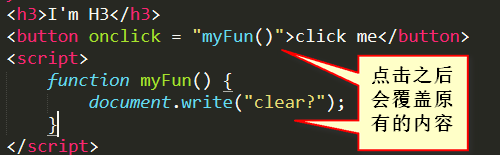
B通过标签名找到HTML元素：getElement**s**ByTagName

C通过类名找到HTML元素：getElement**s**ByClassName

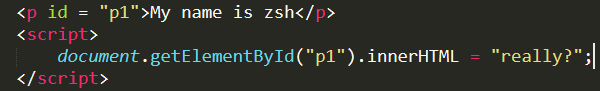
4.2 HTML DOM – 改变html

1）改变HTML输出流

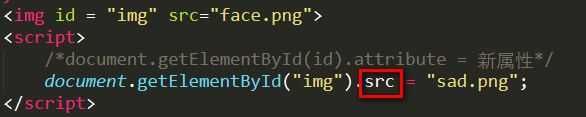
不要再文档加载完成之后使用document.write()，因为这会覆盖该文档



2）改变HTML内容



3）改变HTML属性

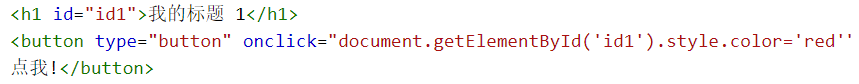


4.3 HTML DOM – 改变css

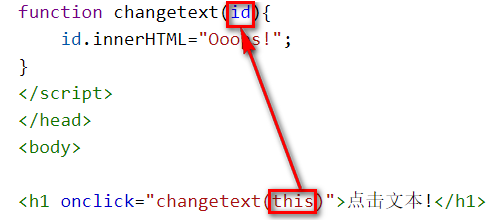
1）改变HTML样式



2）使用事件



4.4 HTML DOM事件





4.5 EventListener

1）addEventListener()

A 向指定元素添加事件句柄，不会覆盖已存在的事件句柄。

B 可以向一个元素添加多个事件句柄，也可以向同个元素添加多个同类型的事件句柄。

C可以更简单的控制事件（冒泡与捕获）

D在没有控制HTML标记时也可以添加事件监听

2）语法

Element.addEventListener(event, function, useCapture)

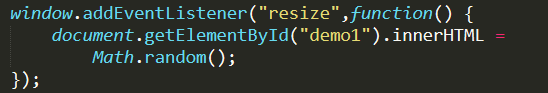
3）添加句柄

A向原元素添加事件句柄

B向同一个元素中添加多个事件句柄



C向Window对象添加事件句柄



4）事件冒泡事件捕获

A事件传递两种方式：冒泡与捕获

B在冒泡中，内部元素的事件会先被触发，然后再触发外部元素

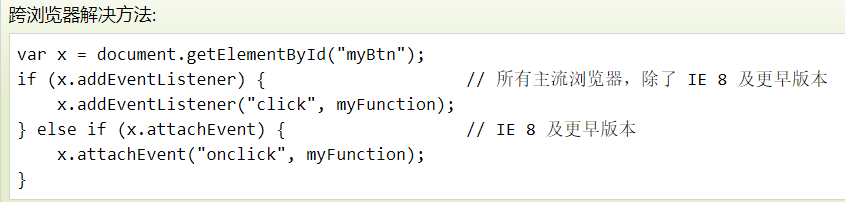
C在捕获中，外部元素的事件会先被触发，然后再触发内部元素

D指定“useCapture”参数来设置传递类型，默认值为false(冒泡传递)，true为捕获传递

5）removeEventListener() 移除由addEventListener()方法添加的事件句柄

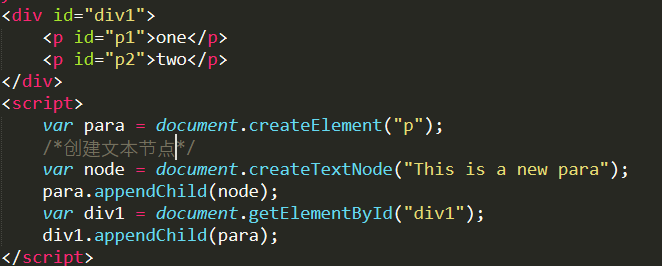


6）跨浏览器解决方法

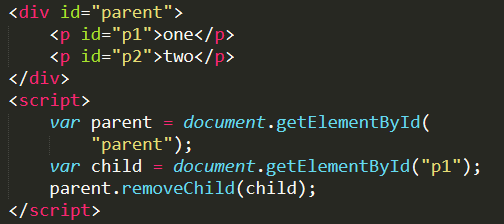


4.6元素节点

1）创建新的html元素



2）删除已有html元素



找到希望删除的元素，然后使用parentNode属性来找到父元素：

