# html5

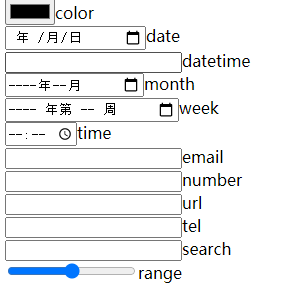
## 语义标签

|  |  |
| --- | --- |
| 标签 | 描述 |
| <header> | 定义了文档的头部区域 |
| <footer> | 定义了文档的尾部区域 |
| <nav> | 定义文档的导航 |
| <section> | 定义文档中的节 |
| <article> | 定义文章 |
| <aside> | 定义页面以外的内容 |
| <details> | 定义用户可以看到或者隐藏的额外细节 |
| <summary> | 标签包含details元素的标题 |
| <dialog> | 定义对话框 |
| <figure> | 定义自包含内容，如图表 |
| <main> | 定义文档主内容 |
| <mark> | 定义文档的主内容 |
| <time> | 定义日期/时间 |

## 增强型表单

### Input 新type

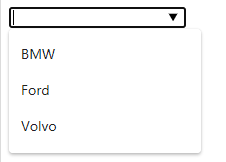
|  |  |
| --- | --- |
| 输入类型 | 描述 |
| color | 主要用于选取颜色 |
| date | 选取日期 |
| datetime | 选取日期(UTC时间) |
| datetime-local | 选取日期（无时区） |
| month | 选择一个月份 |
| week | 选择周和年 |
| time | 选择一个时间 |
| email | 包含e-mail地址的输入域 |
| number | 数值的输入域 |
| url | url地址的输入域 |
| tel | 定义输入电话号码和字段 |
| search | 用于搜索域 |
| range | 一个范围内数字值的输入域 |



### 五个表单元素

|  |  |
| --- | --- |
| <datalist> | 用户会在他们输入数据时看到域定义选项的下拉列表 |
| <progress> | 进度条，展示连接/下载进度 |
| <meter> | 刻度值，用于某些计量，例如温度、重量等 |
| <keygen> | 提供一种验证用户的可靠方法  生成一个公钥和私钥 |
| <output> | 用于不同类型的输出  比如尖酸或脚本输出 |

1. datalist



<input list="cars" />

<datalist id="cars">

<option value="BMW">

<option value="Ford">

<option value="Volvo">

</datalist>

1. Progress



<progress value="30" max="100">

1. meter



<meter value="8" min="0" max="10">2 out of 10</meter><br>

<meter value="0.6">60%</meter>

1. keygen



<form action="/example/html5/demo\_form.asp" method="get">

用户名：<input type="text" name="usr\_name" />

加密：<keygen name="security" />

<input type="submit" />

</form>

1. output



<form oninput="x.value=parseInt(a.value)+parseInt(b.value)">0

<input type="range" id="a" value="50">100

+<input type="number" id="b" value="50">

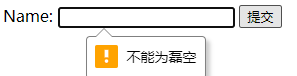
=<output name="x" for="a b"></output>

</form>

### 表单属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| placehoder | 输入框默认提示文字 |
| required | 要求输入的内容是否可为空 |
| pattern | 描述一个正则表达式验证输入的值 |
| min/max | 设置元素最小/最大值 |
| step | 为输入域规定合法的数字间隔 |
| height/wdith | 用于image类型<input>标签图像高度/宽度 |
| autofocus | 规定在页面加载时，域自动获得焦点 |
| multiple | 规定<input>元素中可选择多个值 |

1. Required



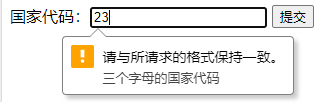
<form action="/example/html5/demo\_form.asp" method="get">

Name: <input type="text" name="usr\_name" required="required" oninvalid="setCustomValidity('提示文字')" />

<input type="submit" value="提交" />

</form>

1. pattern



<form action="/example/html5/demo\_form.asp" method="get">

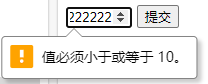
国家代码：<input type="text" name="country\_code" pattern="[A-z]{3}"

title="三个字母的国家代码" />

<input type="submit" />

</form>

1. min/max



<form action="/example/html5/demo\_form.asp">

<input type="number" name="points" min="0" max="10" />

<input type="submit" value="提交" />

</form>

1. step

这个要和type类型联用

<input type="number" name="points" step="3" />

1. Multiple

用于上传文件的多选

<form action="/example/html5/demo\_form.asp" method="get">

选择图片：<input type="file" name="img" multiple="multiple" />

<input type="submit" />

</form>

## 音频和视频

### 音频

音频：<audio src=" "></audio>

<audio controls> //controls属性提供添加播放、暂停和音量控件。

<source src="horse.ogg" type="audio/ogg">

<source src="horse.mp3" type="audio/mpeg">

您的浏览器不支持 audio 元素。 //浏览器不支持时显示文字</audio>

### 视频

视频：<video src=" "></video>

<video width="320" height="240" controls>

<source src="movie.mp4" type="video/mp4">

<source src="movie.ogg" type="video/ogg">

您的浏览器不支持Video标签。</video>

## Canvas绘图

[https://www.runoob.com/w3cnote/html5-canvas-intro.html](https://www.runoob.com/w3cnote/html5-canvas-intro.html" \t "https://www.cnblogs.com/binguo666/p/_blank)

## SVG绘图

## 拖放API

div draggable="true" ondragstart="drag(event)"></div>

<script>

function drap(ev){

console.log(ev);

}

</script>

|  |  |  |
| --- | --- | --- |
| 拖动生命周期 | 属性名 | 描述 |
| 拖动开始 | ondragstart | 在拖动操作开始时执行脚本 |
| 拖动过程中 | ondrag | 只要脚本在被拖动就运行脚本 |
| 拖动过程中 | ondragenter | 当元素被拖动到一个合法的防止目标时，执行脚本 |
| 拖动过程中 | ondragover | 只要元素正在合法的防止目标上拖动时，就执行脚本 |
| 拖动过程中 | ondragleave | 当元素离开合法的防止目标时 |
| 拖动结束 | ondrop | 将被拖动元素放在目标元素内时运行脚本 |
| 拖动结束 | ondragend | 在拖动操作结束时运行脚本 |

## Web Worker // todo

## Web Storage

localStorage.setItem(key,value);//保存数据

let value = localStorage.getItem(key);//读取数据

localStorage.removeItem(key);//删除单个数据

localStorage.clear();//删除所有数据

let key = localStorage.key(index);//得到某个索引的值

## WebSocket

特点：

  （1）握手阶段采用HTTP协议，默认端口是80和443

  （2）建立在TCP协议基础之上，和http协议同属于应用层

  （3）可以发送文本，也可以发送二进制数据。

  （4）没有同源限制，客户端可以与任意服务器通信。

  （5）协议标识符是ws（如果加密，为wss），如ws://localhost:8023

# Css3

https://www.runoob.com/css3/css3-multiple-columns.html

## [选择器](https://www.cnblogs.com/starof/p/4563756.html)

E F 均既可以是DOM 加.是样式 加#是ID

### 层级选择器>

1. F  
   子选择器  
   选择匹配的F元素，且匹配的F元素所匹配的E元素的子元素

子结合符两边可以有空白符，这是可选的。因此，以下写法都没有问题：

h1 > strong

h1> strong

h1 >strong

h1>strong

### 同级元素通用选择器~

E~F

E用来找到目标DOM，F为和E同级的，且在F之后的所有元素

### 相邻兄弟选择器+

h1 + p {margin-top:50px;} 像这种h1和p之间没有关系，只取h1后边第一个p

p + p {margin-top:50px;} 如果都是p，p+p也等于p~p

### 属性选择器

注意：

1. 属性值可以带引号也可以不带input[type\*=text] 等同于 input[type\*=”text”]
2. 可以同时有多个属性a[href][title]

|  |  |
| --- | --- |
| **选择器** | **描述** |
| [[](https://www.w3school.com.cn/cssref/selector_attribute.asp" \o "CSS [attribute] 选择器)*[attribute](https://www.w3school.com.cn/cssref/selector_attribute.asp" \o "CSS [attribute] 选择器)*[]](https://www.w3school.com.cn/cssref/selector_attribute.asp" \o "CSS [attribute] 选择器) | 用于选取带有指定属性的元素。 |
| [[](https://www.w3school.com.cn/cssref/selector_attribute_value.asp" \o "CSS [attribute=value] 选择器)*[attribute](https://www.w3school.com.cn/cssref/selector_attribute_value.asp" \o "CSS [attribute=value] 选择器)*[=](https://www.w3school.com.cn/cssref/selector_attribute_value.asp" \o "CSS [attribute=value] 选择器)*[value](https://www.w3school.com.cn/cssref/selector_attribute_value.asp" \o "CSS [attribute=value] 选择器)*[]](https://www.w3school.com.cn/cssref/selector_attribute_value.asp" \o "CSS [attribute=value] 选择器) | 用于选取带有指定属性和值的元素。 |
| [[](https://www.w3school.com.cn/cssref/selector_attribute_value_contain.asp" \o "CSS [attribute~=value] 选择器)*[attribute](https://www.w3school.com.cn/cssref/selector_attribute_value_contain.asp" \o "CSS [attribute~=value] 选择器)*[~=](https://www.w3school.com.cn/cssref/selector_attribute_value_contain.asp" \o "CSS [attribute~=value] 选择器)*[value](https://www.w3school.com.cn/cssref/selector_attribute_value_contain.asp" \o "CSS [attribute~=value] 选择器)*[]](https://www.w3school.com.cn/cssref/selector_attribute_value_contain.asp" \o "CSS [attribute~=value] 选择器) | 用于选取属性值中包含指定词汇的元素。（必须是空格隔开的独立的一个元素） |
| [[](https://www.w3school.com.cn/cssref/selector_attribute_value_start.asp" \o "CSS [attribute|=value] 选择器)*[attribute](https://www.w3school.com.cn/cssref/selector_attribute_value_start.asp" \o "CSS [attribute|=value] 选择器)*[|=](https://www.w3school.com.cn/cssref/selector_attribute_value_start.asp" \o "CSS [attribute|=value] 选择器)*[value](https://www.w3school.com.cn/cssref/selector_attribute_value_start.asp" \o "CSS [attribute|=value] 选择器)*[]](https://www.w3school.com.cn/cssref/selector_attribute_value_start.asp" \o "CSS [attribute|=value] 选择器) | 用于选取带有以指定值开头的属性值的元素，该值必须是整个单词，后面可以加-。 |
| [[](https://www.w3school.com.cn/cssref/selector_attr_begin.asp" \o "CSS [attribute^=value] 选择器)*[attribute](https://www.w3school.com.cn/cssref/selector_attr_begin.asp" \o "CSS [attribute^=value] 选择器)*[^=](https://www.w3school.com.cn/cssref/selector_attr_begin.asp" \o "CSS [attribute^=value] 选择器)*[value](https://www.w3school.com.cn/cssref/selector_attr_begin.asp" \o "CSS [attribute^=value] 选择器)*[]](https://www.w3school.com.cn/cssref/selector_attr_begin.asp" \o "CSS [attribute^=value] 选择器) | 匹配属性值以指定值开头的每个元素。（不需要完整的空格隔开的一个独立元素，带那几个单词就行） |
| [[](https://www.w3school.com.cn/cssref/selector_attr_end.asp" \o "CSS [attribute$=value] 选择器)*[attribute](https://www.w3school.com.cn/cssref/selector_attr_end.asp" \o "CSS [attribute$=value] 选择器)*[$=](https://www.w3school.com.cn/cssref/selector_attr_end.asp" \o "CSS [attribute$=value] 选择器)*[value](https://www.w3school.com.cn/cssref/selector_attr_end.asp" \o "CSS [attribute$=value] 选择器)*[]](https://www.w3school.com.cn/cssref/selector_attr_end.asp" \o "CSS [attribute$=value] 选择器) | 匹配属性值以指定值结尾的每个元素。（同上） |
| [[](https://www.w3school.com.cn/cssref/selector_attr_contain.asp" \o "CSS [attribute*=value] 选择器)*[attribute](https://www.w3school.com.cn/cssref/selector_attr_contain.asp" \o "CSS [attribute*=value] 选择器)*[\*=](https://www.w3school.com.cn/cssref/selector_attr_contain.asp" \o "CSS [attribute*=value] 选择器)*[value](https://www.w3school.com.cn/cssref/selector_attr_contain.asp" \o "CSS [attribute*=value] 选择器)*[]](https://www.w3school.com.cn/cssref/selector_attr_contain.asp" \o "CSS [attribute*=value] 选择器) | 匹配属性值中包含指定值的每个元素。（同上） |

### ::before和::after

::before和::after这两个主要用来给元素的前面或后面插入内容，这两个常和”content”配合使用，使用的场景最多的就是清除浮动。

例如：div li~li:before{

content: "|";  
 color: red;  
 font-size: 24px;  
 padding-right: 5px;  
}

### 多类选择器

.a.b {background:red;}

<p class="a b">This paragraph is a very important warning.</p>

两个样式之前没有空格，也没有逗号，表示同时有样式a 和样式b的时候，有逗号表格样式a和样式b分别拥有这个样式，两个区别很大

### 伪类

1. 结构性伪类选择器—first-child

:first-child 第一个子元素

p > i:first-child {

background:red;

}

P的子元素里的第一个i

p:first-child i {

color:blue;

}

第一个P里边的i

1. 结构性伪类选择器—last-child

最后一个子元素。

1. 结构性伪类选择器—nth-child(n)

用来定位某个父元素的一个或多个特定的子元素。其中“n”是其参数，而且可以是整数值(1,2,3,4)，也可以是表达式(2n+1、-n+5)和关键词(odd、even)，但参数n的起始值始终是1，而不是0。也就是说，参数n的值为0时，选择器将选择不到任何匹配的元素。

经验与技巧:当“:nth-child(n)”选择器中的n为一个表达式时，其中n是从0开始计算，当表达式的值为0或小于0的时候，不选择任何匹配的元素。如2n+1

注意: 如果只有n，n从1开始，如果有表达式，譬如2n+1 ，n从0开始

1. only-child选择器

是父元素中只有一个子元素，而且只有唯一的一个子元素。也就是说，匹配的元素的父元素中仅有一个子元素，而且是一个唯一的子元素。

1. :enabled和:disabled选择器

在Web的表单中，有些表单元素有可用（“:enabled”）和不可用（“:disabled”）状态，比如输入框，密码框，复选框等。在默认情况之下，这些表单元素都处在可用状态。那么我们可以通过伪选择器:enabled对这些表单元素设置样式。

:disabled选择器刚好与:enabled选择器相反，用来选择不可用表单元素。要正常使用:disabled选择器，需要在表单元素的HTML中设置“disabled”属性。

1. :checked选择器

在表单元素中，单选按钮和复选按钮都具有选中和未选中状态。（大家都知道，要覆写这两个按钮默认样式比较困难）。在CSS3中，我们可以通过状态选择器:checked配合其他标签实现自定义样式。而:checked表示的是选中状态。

1. ::selection选择器

::selection伪元素是用来匹配突出显示的文本(用鼠标选择文本时的文本)。浏览器默认情况下，用鼠标选择网页文本是以“深蓝的背景，白色的字体”显示的。

1. read-only和read-write选择器

:read-only伪类选择器用来指定处于只读状态元素的样式。简单点理解就是，元素中设置了“readonly=’readonly’”

:read-write选择器刚好与:read-only选择器相反，主要用来指定当元素处于非只读状态时的样式。

### :lang 伪类（不常用）

:root{background:orange}

<html>

<head>

<style type="text/css">q:lang(no)

{

quotes: "~" "~"

}

</style>

</head>

<body>

<p>Some text <q lang="no">A quote in a paragraph</q> Some text.</p>

</body>

</html>

## 渐变效果

全兼容  
background-image: -ms-linear-gradient(top, #fff, #ff4f02);  
background-image: -moz-linear-gradient(top, #fff, #ccc);  
background-image: -webkit-gradient(linear, left top, left bottom, from(#fff), to(#ff4f02));  
background-image: -webkit-linear-gradient(top, #fff, #ff4f02);  
background-image: -o-linear-gradient(top, #fff, #ff4f02);  
background-image: linear-gradient(top, #fff, #ff4f02);

基础方法（默认top）

background-image: linear-gradient(#fff, #dededc);

带方向

background-image: linear-gradient(left, #fff, #dededc);

带方向且带值

background-image:-webkit-linear-gradient(left, red 1px, yellow 30%);

带方向且带值且有标准角度  
left, right, top, bottom, left top, left bottom, right top, right， bottom

background-image: -webkit-linear-gradient(right bottom,#fff, #ff4f02);

带方向且带值且有固定角度

background-image:linear-gradient(-45deg, red, yellow)

在Chrome浏览器下的渐变方向居然是相反的！但是45deg是正常的。Firefox浏览器下也是如此，有前缀和没有前缀方向相反！咋回事？

原因很简单，CSS3目前还是草案阶段！

从浏览器去掉前缀前后的变化可以推测，之前，W3C的渐变坐标是与photoshop中一致的，但是，后来，由于某些原因，修改了。

至于什么原因，根据我草草的查找，可能与下面几个关键字之一有联系：animation/transition动画、write-mode书写方向、flex box模型、以及radial-gradient渐变等。在这里就不深入研究了！

径向渐变

background-image: radial-gradient(red, green, blue); /\* 标准的语法（必须放在最后） \*/

## 栅格布局 grid //todo

display: grid; （独占一行，默认）

display: inline-grid; （多个标签同在一行）

grid-template-columns: 150px 1fr 2fr; （fr类似于flex布局中，自元素分别的flex，而且也同时可以搭配具体值一起使用）

grid-template-columns可以设定的值

.container {

display: grid;

grid-template-columns: 100px 100px 100px;

// 有几个值代表几列的宽度 第一列宽度、 第二列宽度、 第三列宽度

grid-template-rows: 100px 100px 100px;

// 有几个值代表几行的宽度 第一行宽度、 第二行宽度、 第三行宽度

}

单行，超过3条出滚动条

<div class="container">  
 <div>1</div>  
 <div>2</div>  
 <div>3</div>  
 <div>4</div>  
 <div>5</div>  
 <div>6</div>  
 <div>6</div>  
 <div>6</div>  
</div>

.container {  
 display: grid;  
 width: 800px;  
 height: 85px;  
 overflow: auto;  
 grid-template-columns: calc(33.3% - 20px) calc(33.3% - 20px) calc(33.3% - 20px);  
 grid-template-rows: 85px 85px 85px 85px 85px 85px 85px;  
 grid-gap: 30px 30px;  
}

## 弹性盒模型 flex

<div class="container2\_total">  
<div class="container2">  
 <div>1</div>  
 <div>2</div>  
 <div>3</div>  
 <div>4</div>  
 <div>5</div>  
 <div>6</div>  
 <div>6</div>  
 <div>6</div>  
</div>  
</div>

实现横向滚动

.container2\_total{  
 width: 200px;  
 height: 100px;  
 overflow: auto;  
}  
.container2{  
 display: flex;  
 -webkit-column-count:3;  
 -moz-column-count:3;  
 column-count:3;  
 justify-content: space-between;  
}  
.container2 div {  
 width: 100px;  
 height: 100px;  
 flex-direction: column;  
 flex-shrink: 0; // 这个是重点  
 background-color: #01B558;  
}  
.container2 div~div{  
 margin-left: 10px;  
}

## 多列布局 column

1. column-count //需要分割的列数。

-webkit-column-count: 3; /\* Chrome, Safari, Opera \*/  
-moz-column-count: 3; /\* Firefox \*/  
 column-count: 3;

1. column-gap // 列与列间的间隙
2. column-rule-style // 指定了列与列间的边框样式

column-rule-style // 指定了列与列间的边框样式

webkit-column-rule-style: solid; /\* Chrome, Safari, Opera \*/  
 -moz-column-rule-style: solid; /\* Firefox \*/  
 column-rule-style: solid;

1. column-rule-width 两列的边框厚度:

-webkit-column-rule-width: 1px; /\* Chrome, Safari, Opera \*/  
-moz-column-rule-width: 1px; /\* Firefox \*/  
column-rule-width: 1px;

1. column-rule-color 列的边框颜色

-webkit-column-rule-color: lightblue; /\* Chrome, Safari, Opera \*/  
-moz-column-rule-color: lightblue; /\* Firefox \*/  
column-rule-color: lightblue;

1. column-rule 属性集合

-webkit-column-rule: 1px solid lightblue; /\* Chrome, Safari, Opera \*/  
-moz-column-rule: 1px solid lightblue; /\* Firefox \*/  
column-rule: 1px solid lightblue;

1. column-span 指定元素跨越多少列  
   -webkit-column-span: all; /\* Chrome, Safari, Opera \*/  
   column-span: all;
2. column-width 指定列的宽度

 -webkit-column-width: 100px; /\* Chrome, Safari, Opera \*/  
column-width: 100px;

## 盒模型定义

box-sizing:border-box的时候，边框和padding包含在元素的宽高之内！

box-sizing:content-box的时候，边框和padding不包含在元素的宽高之内！

## 媒体查询 @media

用来定义不用屏幕大小不同样式

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

body {

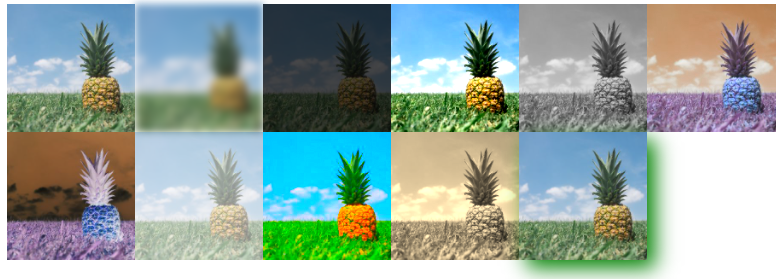
background-color:lightblue;

}

}

## Filter（滤镜）

|  |  |
| --- | --- |
| **Filter** | **描述** |
| none | 默认值，没有效果。 |
| blur(px) | 给图像设置高斯模糊。"radius"一值设定高斯函数的标准差，或者是屏幕上以多少像素融在一起， 所以值越大越模糊；  如果没有设定值，则默认是0；这个参数可设置css长度值，但不接受百分比值。 |
| brightness(%) | 给图片应用一种线性乘法，使其看起来更亮或更暗。如果值是0%，图像会全黑。值是100%，则图像无变化。其他的值对应线性乘数效果。值超过100%也是可以的，图像会比原来更亮。如果没有设定值，默认是1。 |
| contrast(%) | 调整图像的对比度。值是0%的话，图像会全黑。值是100%，图像不变。值可以超过100%，意味着会运用更低的对比。若没有设置值，默认是1。 |
| drop-shadow(h-shadow v-shadow blur spread color) | 给图像设置一个阴影效果。阴影是合成在图像下面，可以有模糊度的，可以以特定颜色画出的遮罩图的偏移版本。 函数接受<shadow>(在CSS3背景中定义)类型的值，除了"inset"关键字是不允许的。该函数与已有的box-shadow box-shadow属性很相似；不同之处在于，通过滤镜，一些浏览器为了更好的性能会提供硬件加速。<shadow>参数如下：  **<offset-x>** **<offset-y>** (必须)  这是设置阴影偏移量的两个 <length>值. **<offset-x>** 设定水平方向距离. 负值会使阴影出现在元素左边. **<offset-y>**设定垂直距离.负值会使阴影出现在元素上方。查看**<length>**可能的单位. **如果两个值都是0**, 则阴影出现在元素正后面 (如果设置了 <blur-radius> and/or <spread-radius>，会有模糊效果).  **<blur-radius>** (可选)  这是第三个code><length>值. 值越大，越模糊，则阴影会变得更大更淡.不允许负值 若未设定，默认是0 (则阴影的边界很锐利).  **<spread-radius>** (可选)  这是第四个 <length>值. 正值会使阴影扩张和变大，负值会是阴影缩小.若未设定，默认是0 (阴影会与元素一样大小).  注意: Webkit, 以及一些其他浏览器 不支持第四个长度，如果加了也不会渲染。    **<color>** (可选)  查看 <color>该值可能的关键字和标记。若未设定，颜色值基于浏览器。在Gecko (Firefox), Presto (Opera)和Trident (Internet Explorer)中， 会应用color**color**属性的值。另外, 如果颜色值省略，WebKit中阴影是透明的。 |
| grayscale(%) | 将图像转换为灰度图像。值定义转换的比例。值为100%则完全转为灰度图像，值为0%图像无变化。值在0%到100%之间，则是效果的线性乘子。若未设置，值默认是0； |
| hue-rotate(deg) | 给图像应用色相旋转。"angle"一值设定图像会被调整的色环角度值。值为0deg，则图像无变化。若值未设置，默认值是0deg。该值虽然没有最大值，超过360deg的值相当于又绕一圈。 |
| invert(%) | 反转输入图像。值定义转换的比例。100%的价值是完全反转。值为0%则图像无变化。值在0%和100%之间，则是效果的线性乘子。 若值未设置，值默认是0。 |
| opacity(%) | 转化图像的透明程度。值定义转换的比例。值为0%则是完全透明，值为100%则图像无变化。值在0%和100%之间，则是效果的线性乘子，也相当于图像样本乘以数量。 若值未设置，值默认是1。该函数与已有的opacity属性很相似，不同之处在于通过filter，一些浏览器为了提升性能会提供硬件加速。 |
| saturate(%) | 转换图像饱和度。值定义转换的比例。值为0%则是完全不饱和，值为100%则图像无变化。其他值，则是效果的线性乘子。超过100%的值是允许的，则有更高的饱和度。 若值未设置，值默认是1。 |
| sepia(%) | 将图像转换为深褐色。值定义转换的比例。值为100%则完全是深褐色的，值为0%图像无变化。值在0%到100%之间，则是效果的线性乘子。若未设置，值默认是0； |
| url() | URL函数接受一个XML文件，该文件设置了 一个SVG滤镜，且可以包含一个锚点来指定一个具体的滤镜元素。  例如：  filter: url(svg-url#element-id) |
| initial | 设置属性为默认值，可参阅： [CSS initial 关键字](https://www.runoob.com/cssref/css-initial.html) |
| inherit | 从父元素继承该属性，可参阅：[CSS inherit 关键字](https://www.runoob.com/cssref/css-inherit.html) |



<!DOCTYPE html>

<html>

<head>

<style>

img {

width: 33%;

height: auto;

float: left;

}

.blur {-webkit-filter: blur(4px);filter: blur(4px);}

.brightness {-webkit-filter: brightness(0.30);filter: brightness(0.30);}

.contrast {-webkit-filter: contrast(180%);filter: contrast(180%);}

.grayscale {-webkit-filter: grayscale(100%);filter: grayscale(100%);}

.huerotate {-webkit-filter: hue-rotate(180deg);filter: hue-rotate(180deg);}

.invert {-webkit-filter: invert(100%);filter: invert(100%);}

.opacity {-webkit-filter: opacity(50%);filter: opacity(50%);}

.saturate {-webkit-filter: saturate(7); filter: saturate(7);}

.sepia {-webkit-filter: sepia(100%);filter: sepia(100%);}

.shadow {-webkit-filter: drop-shadow(8px 8px 10px green);filter: drop-shadow(8px 8px 10px green);}

</style>

</head>

<body>

<p><strong>注意：</strong> Internet Explorer 不支持 filter 属性。</p>

<img src="pineapple.jpg" alt="Pineapple" width="300" height="300">

<img class="blur" src="pineapple.jpg" alt="Pineapple" width="300" height="300">

<img class="brightness" src="pineapple.jpg" alt="Pineapple" width="300" height="300">

<img class="contrast" src="pineapple.jpg" alt="Pineapple" width="300" height="300">

<img class="grayscale" src="pineapple.jpg" alt="Pineapple" width="300" height="300">

<img class="huerotate" src="pineapple.jpg" alt="Pineapple" width="300" height="300">

<img class="invert" src="pineapple.jpg" alt="Pineapple" width="300" height="300">

<img class="opacity" src="pineapple.jpg" alt="Pineapple" width="300" height="300">

<img class="saturate" src="pineapple.jpg" alt="Pineapple" width="300" height="300">

<img class="sepia" src="pineapple.jpg" alt="Pineapple" width="300" height="300">

<img class="shadow" src="pineapple.jpg" alt="Pineapple" width="300" height="300">

</body>

</html>

## 边框属性

### 边框颜色border-color

border-style:solid;

border-color:#ff0000 #0000ff;

注意："border-width" 属性如果单独使用的话是不会起作用的。请首先使用 "border-style" 属性来设置边框。

border-color

### 图片边框border-image

-webkit-border-image:url(border.png) 30 30 round; /\* Safari 5 \*/

-o-border-image:url(border.png) 30 30 round; /\* Opera \*/

border-image:url(border.png) 30 30 round;

### 圆角边框border-radius

border:2px solid;

border-radius:25px;

### 阴影效果box-shadow

box-shadow: 10px 10px 5px #888888;

## 背景属性

### 背景图片尺寸background-size

background-size:80px 60px;

能够以像素或百分比规定尺寸。如果以百分比规定尺寸，那么尺寸相对于父元素的宽度和高度。

### 背景图片从哪里开始显示background-origin

background-origin:content-box;

背景图片可以放置于 content-box、padding-box 或 border-box 区域。

### 背景图片从什么位置开始裁剪background-clip

background-clip:content-box;

## 新增文字效果

### 文本阴影text-shadow

text-shadow: 5px 5px 5px #FF0000;

说明:

(1) <颜色>和<模糊半径>是可选的, 当<颜色>未指定时,将使用文本颜色; 当<模糊半径>未指定时, 半径值为0；

(2) shadow可以是逗号分隔的列表, 如:text-shadow: 2px 2px 2px #ccc, 3px 3px 3px #ddd;

(3) 阴影效果会按照shadow list中指定的顺序应用到元素上；

(4) 这些阴影效果有可能相互重叠, 但不会叠加文本本身；

(5) 阴影可能会跑到容器的边界之外, 但不会影响容器的大小。

### 自动换行word-wrap

word-wrap:break-word;

## 动画效果

### 变换效果transform

css3提供了元素变形效果，也叫做变换。它可以将元素实现旋转、缩放和平移的功能。

属性值：（1）transform ；（2）transform-origin：transform-origin 属性可以设置变换的起点。默认情况下，使用元素的中心作为起点。

transform:rotate(30deg); 旋转30度

### 动画效果animation

CSS3 提供了类似 Flash 关键帧控制的动画效果，通过 animation 属性实现。那么之前的 transition 属性只能通过指定属性的初始状态和结束状态来实现动画效果，有一定的局限性。

animation 实现动画效果主要由两个部分组成：1、通过类似 Flash 动画中的关键帧声明一个动画；2、在 animation 属性中调用关键帧声明的动画。

## 过渡效果

过渡效果一般是通过一些简单的 CSS 动作触发平滑过渡功能，比如：:hover、:focus、:active、:checked 等。CSS3 提供了 transition 属性来实现这个过渡功能。

# Es5新特性总结

# Es6

## 基础变化

### 变量声明let和const

let表示变量（还可以重新赋值）、const表示常量（不可以重新赋值）。

let和const都是块级作用域。说白了 {}大括号内的代码块即为let 和 const的作用域。

var 有个bug就是方法内也会定义全局变量，会造成方法外方法内都不能有同名变量

### 用${}来将字符串格式化。

备注：这里不是单引号，是1左边的那个

const name = 'lux'

console.log(`hello ${name}`)

### 字符串处理

1. includes：判断是否包含然后直接返回布尔值 （类似indexOf）

let str = 'hahay' console.log(str.includes('y')) // true

2.repeat: 获取字符串重复n次 let s = 'he' console.log(s.repeat(3)) // 'hehehe'

3.小数取整, 正数取整数部分，负数取整数部分减1，Math.floor(num)

exp: Math.floor(11.323) = 11 Math.floor(-11.83) = -12

### 方法参数可提供默认值

 function action(num = 200){  
        console.log(num)   
    }

### 可省:function

const people = {   
        name: 'lux', getName: function() {  
            console.log(this.name)   
        }  
    }

可写为

const people = {   
        name: 'lux', getName (){  
            console.log(this.name)   
        }   
    }

### Object.assign()这个方法来实现浅复制

 const obj = Object.assign({}, objA, objB)

{}为对象型

[]为数组型

### 解构

对象:

const people = { name: 'lux', age: 20 }   
const { name, age } = people   
console.log(`${name} --- ${age}`)

数组:

const color = ['red', 'blue']   
const [first, second] = color   
console.log(first) //'red'   
console.log(second) //'blue'

### 用同步的方式去写异步代码

发起异步请求：fetch('/api/todos') .then(res => res.json()).then(data =>({ data })).catch(err => ({ err }));

### 生成器（ generator）是能返回一个迭代器的函数//todo

## 箭头函数Arrow Function

数组常用方法fliter,map,forEach,some

### Fliter

filter() 方法创建一个新的数组，新数组中的元素是通过检查指定数组中符合条件的所有元素。

注意： filter() 不会对空数组进行检测。

注意： filter() 不会改变原始数组。

格式为：数组.filter(条件方法);

let arr = [  
 {'title':'标题标题标题标题','status':0},  
 {'title':'标题标题标题标题','status':1},  
 {'title':'标题标题标题标题','status':2},  
 {'title':'标题标题标题标题','status':3},  
 {'title':'标题标题标题标题','status':4},  
]  
function checkAdult(arr) {  
 return arr.status !=2;  
}  
var b= arr.filter(checkAdult);

var b= arr.filter(checkAdult);

var b= arr.filter(d => d.status === 0);

console.log(b)

简单的几个条件过滤直接这样写

var b= arr.filter(d => d.status === 0);

### forEach

//data中定义

data() {  
 return {  
 arr: ['a1', 'a2', 'a3', 'a4'],  
 }  
},

//方法中遍历 只有一个变量

this.arr.forEach(item => {  
 alert(item)  
})

当有3个变量时，第一个为当前item，第二个为序号，第三个为原变量

变量.map((item ,index,arr) => {  
 console.log(item)  
 console.log(index)  
 console.log(arr)  
}

### Map

map有返回值，可以return 出来。

let arr = [

{'title':'标题标题标题标题','status':0},

{'title':'标题标题标题标题','status':0},

{'title':'标题标题标题标题','status':0},

{'title':'标题标题标题标题','status':0},

{'title':'标题标题标题标题','status':0},

]

let a = arr.map(item =>{

return {

title:item['title']

}

})

console.log(a)}

### some

some() 方法用于检测数组中的元素是否满足指定条件（函数提供）。

some() 方法会依次执行数组的每个元素：

如果有一个元素满足条件，则表达式返回true , 剩余的元素不会再执行检测。

如果没有满足条件的元素，则返回false。

注意： some() 不会对空数组进行检测。

注意： some() 不会改变原始数组。

var ages = [3, 10, 18, 20];

function checkAdult(age) {

return age > 19;

}

var b = ages.some(checkAdult);

## 展开语法Spread Operator

### 组装对象或者数组

数组:

const color = ['red', 'yellow']

const colorful = [...color, 'green', 'pink']

console.log(colorful) //[red, yellow, green, pink]

对象:

const alp = { fist: 'a', second: 'b'}

const alphabets = { ...alp, third: 'c' }

console.log(alphabets) //{ "fist": "a", "second": "b", "third": "c" }

### 获取数组或者对象除了前几项或者除了某几项的其他项

数组：

number = [1,2,3,4,5]

const [first, ...rest] = number

console.log(rest) //2,3,4,5

对象：

const user = { username: 'lux', gender: 'female', age: 19, address: 'peking' }

const { username, ...rest } = user

console.log(rest) //{"address": "peking", "age": 19, "gender": "female" }

### 组合成新的Object

备注：如果有重复的属性名，右边覆盖左边

 const first = { a: 1, b: 2, c: 6, }

const second = { c: 3, d: 4 }

const total = { ...first, ...second }

console.log(total) // { a: 1, b: 2, c: 3, d: 4 }

## import导入模块、export导出模块//todo

require: node 和 es6 都支持的引入  
export / import : 只有es6 支持的导出引入  
module.exports / exports: 只有 node 支持的导出

下边的示例为export / import : 只有es6 支持的导出引入

必须本地构建一个服务器，且标签上加入 type="module"

可以通过 import 任意名 from './a.js' 引入

as 可以重命名

还可以用（\*）指定一个对象，所有的变量都会加载在这个对象上。

import \* as circle from './circle';

<script type="module">  
 import Programmer from './a.js'  
 import {a,b,hello,haha as c} from './b.js'  
 console.log('Programmer', Programmer )  
 console.log('a', a )  
 console.log('b', b )  
 hello()

c()  
</script>

a.js

export default {name: 'k',age:25}

b.js

export const a = {name: 'a',age:11}  
export let b = {name: 'b',age:22}

export function hello () {  
 console.log( '你好')  
}

export const haha = (() => {  
 console.log( '你不好')  
})

# ES7-ES10

## ES7新特性（2016）

1. 重新优化includes => 可用于数组
2. 指数运算符（\*\*） (求幂运算)

a\*\*b 与 Math.pow(a,b)相同

## ES8（2017）

### 获取对象key值Object.values/Object.entries

Es7写法

const obj = {  
 name :'张三',  
 age: 18  
}  
const vals = Object.keys(obj).map((key)=>{  
 console.log('key', key)  
 return obj[key]  
})  
console.log('vals', vals)

Es8写法

for(let [key,value] of Object.entries(obj)){  
 console.log('key', key)  
 console.log('value', value)  
}

### String padding(字符串填充)

### Object.getOwnPropertyDescriptors

### 函数参数列表和调用中的尾逗号（Trailing commas）

### 异步函数（Async Functions）

## ES9（2018）//todo

Promise的实际运用

const resCount = await DECLAREINFOListCountApi(count)  
const res = await DECLAREINFOListApi({ params })  
Promise.all([resCount, res]).then(result => {  
 const { count } = result[0]  
 this.tableData = result[1]})  
 .catch(() => {  
 this.Pagination.loading = false  
 })

## ES10（2019）//todo