前端

html标签:

结构:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>

</body>
</html>
```

格式:

```
HTML标签
    head  一个的思维
        meta  元信息
            charset 编码
            <meta http-equiv="refresh" content="2;URL=http://www.luffycity.com">

            description  描述
            Keywords  关键字

        title  标题


        <style></style>  <!-- css -->（注释）
        <link rel="stylesheet" href=""> <!-- 连接 -->
        <script></script>    <!-- js -->


    body  一个人的身体
```

基于browser，主要是为了内容展示

请求和响应

1. html：显示的内容
2. css：样式，美化
3. js：动态效果

编写规范:

1. 所有标记元素都要正确的嵌套，不能交叉嵌套。正确写法举例：`<h1><font></font></h1>`
2. 所有的标记都必须小写。
3. 所有的标记都必须关闭。

- 双边标记：`<span></span>`
- 单边标记：`<br>` 转成 `<br />` `<hr>` 转成 `<hr />`，还有`<img src="URL" />`

4. 所有的属性值必须加引号。

```

5. 所有的属性必须有值。```<input type="radio" checked="checked" />`

6. HTML的结构：
   - 声明部分：主要作用是用来告诉浏览器这个页面使用的是哪个标准。是HTML5标准。
   - head部分：将页面的一些额外信息告诉服务器。不会显示在页面上。
   - body部分：我们所写的需要显示出来的代码必须放在此标签内。

## 标记的分类：

html标签又叫做html元素，它分为**块级元素**和**内联元素**（也可以叫做行内元素），都是html规范中的概念。

| 标题 | h1 | h2 | h3 | h4 | h5 | h6 |
|---|---|---|---|---|---|---|
| 列表 | ol | ul | li | dl | dt | dd |
| 排版标签 | p | div | hr | center | pre | |
| 表格 | table | | | | | |
| 表单 | form | | | | | |

1. 双边标记：双封闭标签
2. 单边标记：单封闭标签
3. 块级标签：h1--h6
   1. 特点：

      独占一行，每一个块级元素都会从新的一行重新开始，从上到下排布
      可以直接控制宽度、高度以及盒子模型的相关css属性
      在不设置宽度的情况下，块级元素的宽度是它父级元素内容的宽度
      在不设置高度的情况下，块级元素的高度是它本身内容的高度

4. 行内（内联）标签：span
```

字体	粗体	斜体	上下标	划线
	b	em	sup	del/s
	strong	i	sub	u
排版	span	br		
超链接	a			
图片	img			

1. br换行
2. b，strong：字体加粗
3. i，em：斜体
4. s，del：删除线
5. u：下划线
6. img：代表的就是一张图片，是单边标记，能够插入类型为jpg(jpeg)、gif、png、bmp，不能插入psd，ai格式的图片，引入地址
    1. src：图片的相对路径和绝对路径（src 是英语source“资源”的缩写）
        1. 相对路径：相对当前页面所在的路径。两个标记 `.` 和 `..` 分表代表当前目录和父路径。

```
<!-- 当前目录中的图片 -->
<img src="2.jpg">
<img src=".\2.jpg">
<!-- 上一级目录中的图片 -->
<img src="..\2.jpg">
```

        2. 绝对路径：

（1）以盘符开始的绝对路径。举例：

```
<img src="C:\Users\aaa\Desktop\html-01\images\1.jpg">
```

（2）网络路径。举例：

```
<img src="http://www.baidu.com/2016040102.jpg">
```

        3. 总结：推荐使用相对路径：站点不管拷贝到那里，文件和图片的相对路径关系都是不变的，有一个前提就是网页文件和你的图片，必须在一个服务器上
    2. title：提示性文本，主要用来告诉用户和搜索引擎这个网页的主要内容是什么，搜索引擎可以通过网页标题，迅速的判断出当前网页的主题。

```
格式：<title>你起的标题名</title>
```

    3. alt：图片的提示

4. width：宽度
5. height：高度

7. a（anchor）标签：锚，超链接
    1. herf：
        1. 给定一个网址链接：跳转到当前网址
        2. #回到顶部
        3. Meta标签页面锚点：
            1. http-equiv属性

```
<!--重定向 2秒后跳转到对应的网址，注意分号-->
<meta http-equiv="refresh"
content="2;URL=http://www.luffycity.com">
<!--指定文档的内容类型和编码类型 -->
<meta http-equiv="Content-Type"
content="text/html;charset=utf-8" />
<!--告诉IE浏览器以最高级模式渲染当前网页-->
<meta http-equiv="x-ua-compatible" content="IE=edge">
```

            2. 标签+ id ： 跳转
            3. a标签+ name ： 跳转到name所在。主要用于页面的关键字和描述，是写给搜索引擎看的，关键字可以有多个用‘,’号隔开，与之对应的属性值为content，
                - content中的内容主要是便于搜索引擎机器人查找信息和分类信息用的。

```
 <meta name="Keywords" content="网易,邮箱,游戏,新闻,体育,娱乐,女性,亚运,论坛,短信" />

            - 这些关键词，就是告诉搜索引擎，这个网页是干嘛的，能够提高搜索命中率。让别人能够找到你，搜索到。

            ```Python
            <meta name="Description" content="网易是中国领先的互联网技术公司，为用户提供免费邮箱、游戏、搜索引擎服务，开设新闻、娱乐、体育等30多个内容频道，及博客、视频、论坛等互动交流，网聚人的力量。" />
```

```
        - 只要设置Description页面描述，那么百度搜索结果，就能够显示这些语句，这个技术叫做**SEO**（search engine optimization，搜索引擎优化）。
```

        2. target：
            1. _ self(默认) :当前页面刷新
            2. _ blank ： 新建一个页面窗口

HTML中的特殊字符：

```
 ：空格 （non-breaking spacing，不断打空格）
<：小于号（less than）
```
```

```
&gt;：大于号（greater than）
&amp;：符号&
&quot;：双引号
&apos;：单引号
&copy;：版权©
&trade;：商标™

要求大家背过的特殊字符： 、&lt;、&gt;、&copy

比如说，你想把<p>作为一个文本在页面上显示，直接写<p>是肯定不行的，因为这代表的是一个段落标签，所以这里需要用到转义字符。应该这么写：

这是一个HTML语言的&lt;p&gt;标签

特殊字符参照表：http://tool.chinaz.com/Tools/HtmlChar.aspx
```

6. html颜色
    1. 颜色表示：

```
纯单词表示：red、green、blue、orange、gray等
10进制表示：rgb(255,0,0)
16进制表示：#FF0000、#0000FF、#00FF00等
```

    2. RGB色彩模式

```
    自然界中所有的颜色都可以用红、绿、蓝(RGB)这三种颜色波长的不同强度组合而得，这就是人们常说的三原色原理。
    RGB三原色也叫加色模式，这是因为当我们把不同光的波长加到一起的时候，可以得到不同的混合色。例：红+绿=黄色，红+蓝＝紫色，绿+蓝=青
    在数字视频中，对RGB三基色各进行8位编码就构成了大约1678万种颜色，这就是我们常说的真彩色。所有显示设备都采用的是RGB色彩模式。
    RGB各有256级(0-255)亮度，256级的RGB色彩总共能组合出约1678万种色彩，即256×256×256=16777216。

  HTML颜色对照表：https://htmlcolorcodes.com/zh/yanse-ming/
```

块级标签：都是在body标签内实现

排版标签

1. p标签：一个文本级的段落标签，前后有间距，p标签中不能嵌套其他块级标签
2. div标签：没有任何样式的块级标签
3. hr标签：单边标签，起分割线效果
4. center标签：h5中被废弃了，但仍有效果
5. pre：预定义标签，会保留写的原样式

列表标签

1. 无序（序号）列表：

    ，使用type

    - type= "none" 无，type = "disc" 实心圆，type= "circle"空心圆，type= "square"方点，

```
无序标签
<ul>
    <li>你</li>
    <li>我</li>
    <li>他</li>
</ul>

<ul type ="none">
    <li>瓜子</li>
    <li>啤酒</li>
    <li>花生米</li>
</ul>


<ul type="circle">
    <li>饮料</li>
    <li>啤酒</li>
    <li>炸鸡</li>
</ul>

<ul type="disc">
    <li>啤酒</li>
    <li>炸鸡</li>
    <li>小龙虾</li>
</ul>


<ul type="square">
    <li>生蚝</li>
    <li>麻小</li>
    <li>炸鸡</li>
</ul>
```

- 有序列表：
    - type="数字的样式" start ="起始值"（数字）

```
<!--有序标签-->
<!-- 样式 1（数字）、 a（小写字母）、A（大写）、i（小写）、（罗马数字）-->
<ol start="20">
    <li>瓜子</li>
    <li>花生</li></dd>
    <li>瓜子</li>
    <li>花生米</li>
    <li>毛豆</li>
</ol>

<ol type="1">
    <li>花生</li>
    <li>毛豆</li>
</ol>


<ol type="a">
    <li>可乐</li>
    <li>雪碧</li>
    <li>二锅头</li>
```

```
</ol>

<ol type="A">
    <li>炸鸡</li>
    <li>麻小</li>
    <li>生蚝</li>
</ol>

<ol type="i">
    <li>啤酒</li>
    <li>白酒</li>
    <li>红酒</li>
</ol>

<ol type="I">
    <li>电脑</li>
    <li>手机</li>
    <li>ipad</li>
</ol>
```

1. 定义列表：
    - dt：标题
    - dd：内容

```
<dl>
    <dt>小吃一条街</dt>
    <dd>啤酒</dd>
    <dd>炸鸡</dd>
    <dd>小龙虾</dd>
    <dd>小龙虾</dd>
</dl>
```

表格标签：有表头的表格，table标签

1. table标签属性：border：添加边框，cellpadding：内容与单元格之间的距离，cellsapcing：单元格与边框之间的距离
2. table标签嵌套thead（表头），tbody（表内容）
3. thead，tbody嵌套tr标签（tr标签里面写的是表的格式）
4. tr标签嵌套th（表的第一列数据），td（表的内容）
    1. tr标签属性：
        1. align：内容水平排列 left，center，right
        2. valign：内容垂直排列 top，middle，bottom
    2. td标签属性：
        1. rowspan：占几行
        2. colspan：占几列

```
<table border="1" cellpadding="20px" cellspacing="20px">
    <thead>
    <tr align="left">
        <th> 序号</th>
        <th> 姓名</th>
```

```
        <th> 年龄</th>
    </tr>
    </thead>
    <tbody>
    <tr align="center" valign="bottom">
        <td>1</td>
        <td >alex</td>
        <td >84</td>


    </tr>
    <tr align="center" valign="top">
        <td>2</td>
        <td >alex</td>


    </tr>
    <tr>
        <td>2</td>
        <td>wusir</td>
        <td rowspan="2">2208</td>

    </tr>
    </tbody>

</table>

<!-- 无表头的表格-->
<table border="1" cellpadding="20px" cellspacing="20px">

    <tbody>
    <tr align="center" valign="bottom">
        <td>1</td>
        <td >alex</td>
        <td >84</td>


    </tr>
    <tr align="center" valign="top">
        <td>2</td>
        <td >alex</td>


    </tr>
    <tr>
        <td>2</td>
        <td>wusir</td>
        <td rowspan="2">2208</td>

    </tr>
    </tbody>

</table>
```

表单标签：form标签

1. form表单中的action：提交的地址（后台链接）

2. input标签中
    1. type：类型
        1. text：普通文本
            1. placeholder：提示
            2. readonly：只读，不能改
            3. disabled：禁用，不能改值，不可以提交
        2. password：密码，密文
        3. radio：单选框
        4. checkbox：复选框
        5. submit：提交按钮
    2. name属性：标识标签名称
    3. value：可以设置默认值
    4. hidden：隐藏
    5. reset：创建重置按钮，会清空之前选择的东西
    6. button：创建一个普通按钮，后跟一个value属性
    7. date：日期格式
3. form表单中button标签：创建提交按钮类似于type类型中的submit
4. select：单选框
    1. name中嵌套size=“数字”,multiple
    2. option中嵌套value（默认值）selected（选中）
5. label：用户名：<input placeholder="请输入用户名" />
6. textarea标签：文本框
    1. name：名字
    2. cols：长度
    3. rows：宽度
7. 注意：
    1. 要提交数据，必须有一个input的类型的submit或者button
    2. 要上传文件file 的时候，必须在form表头处改编码enctype="multipart/form-data"

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>

<form action="http://127.0.0.1:9999" >  #连接后台的地址
<!--    <input type="text" name="user" placeholder="请输入用户名">
&lt;!&ndash;name:提交数据的key   value:值   placeholder：占位的内容
&ndash;&gt;-->
<!--    <input type="password" name="pwd">-->
<!--    <input type="radio" name="sex" value="1" checked > 男--> #
radio 单选按钮，需要用name与下一个建立连接，value给后台返回值
<!--    <input type="radio" name="sex" value="2"> 女-->
```

```
<!--    <input type="checkbox" name="hobby" value="1"
checked="checked"> 跳-->    checkbox：多选框
<!--    <input type="checkbox" name="hobby" value="2"> 唱-->
<!--    <input type="checkbox" name="hobby" value="3"> rap-->
<!--    <input type="checkbox" name="hobby" value="4"> 篮球-->
    <!--    <input type="submit">-->    #submit 在input内创建提交按钮

    <!--    <p>-->
    <!--        <label for="i1">用户名：</label><input id="i1"
type="text" name="user" placeholder="请输入用户名">-->  # 使用label标签
与后面的内容建立连接

    <!--    </p>-->
    <!--     <p>-->
    <!--        密码：<input type="password" name="pwd" >-->

    <!--    </p>-->

    <!--    <input type="hidden" name="alex" value="alexdsb">--
>#hidden 隐藏内容
    <!--    <input type="reset">-->  #reset：创建重置按钮，全局选择重置
    <!--    <input type="button" value="提交"> &lt;!&ndash;  普通的按
钮  &ndash;&gt;-->  #input内button创建一个普通按钮

<!--    <select name="city" id="" size="4" multiple>--> #multiple可
以一次选择多个内容，select设置下拉框，size显示下拉框大小

<!--        <option value="1" selected="selected">北京</option>-->
# 下拉框
<!--        <option value="2">上海</option>-->
<!--        <option value="3">深圳</option>-->

<!--    </select>-->

<!--    <input type="file" name="f1">-->  # 上传文件按钮
    <input type="text">

    <button>提交</button>

</form>

</body>
</html>
```

css

1. 引入方式：
    1. 内联引入：style标签，内使用div标签

```
<!--内联引入-->
    <style>
        div {
            color: #ffef6b
        }

    </style>
```

2. 外联引入：link标签,首先要在外部建立一个css样式的文件，推荐使用这个，在head标签内设置

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

3. 外联导入：

```
<style>
        @import url('index.css');
    </style>
```

4. 行内导入：直接在div标签内设置格式

```
<div style="color: red">黄焖鸡米饭</div>
<div>黄焖鸡排骨</div>
```

css简单样式：

1. color：字体颜色
2. width：宽度
3. height：高度
4. background：背景色

选择器

基本选择器：标签\id\类\通用选择器

```
<style>
    #标签选择器
        div {
            color: #ffef6b;
        }

        a {
            color: green;
        }


        span {
            color: #42ff68;
        }
    #id选择器
        #id {
            color: chartreuse;

        }
    #类选择器
        .类名 {
             color: #192aff;
        }

        .x1 {
             background: #3bff00;
        }
```

```
    #通配选择器
        * {
            background: #3bff00;
        }

    </style>
```

```
<body>

<!--<div >黄焖鸡米饭</div>-->
<!--<div>黄焖鸡排骨</div>-->

<div>黄焖鸡米饭
    <span class="cai">鸡</span>
    <span>米饭</span>
    <a href="xxxx">外卖连接</a>
</div>
<div>黄焖鸡排骨
    <span class="cai x1 x2">排骨</span>#类后面可以写多个样式选择器
    <span>米饭</span>
</div>
<span id="sp1">米饭</span>


</body>
```

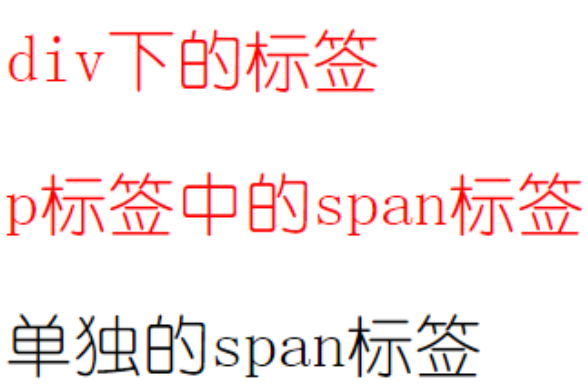
高级选择器

1. 后代\子代

```
后代:
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        div span {
            color: red;
        }
    </style>
</head>
<body>
<div>
    <span>
        div下的标签
    </span>
    <p>
        <span>p标签中的span标签</span>
    </p>
</div>
<span>单独的span标签</span>
</body>
</html>
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        div span {
            color: red;
        }
    </style>
</head>
<body>
<div>
    <span>
        div下的标签
    </span>
    <p>
        <span>p标签中的span标签</span>
    </p>
</div>
<span>单独的span标签</span>
</body>
</html>
```

网页上显示的效果如下：



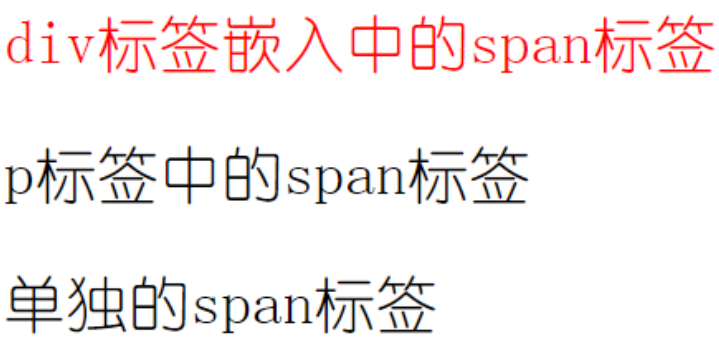
子代：

```
    <style>
        div > span {
            color: red;
        }
    </style>
</head>
<body>
<div>
    <span>div标签嵌套中的span标签</span>
    <p>
        <span>p标签中的span标签</span>
    </p>
</div>
```

```
<span>单独的span标签</span>

</body>
</html>
```

```
    <style>
        div > span {
            color: red;
        }
    </style>
</head>
<body>
<div>
    <span>div标签嵌套中的span标签</span>
    <p>
        <span>p标签中的span标签</span>
    </p>
</div>
<span>单独的span标签</span>

</body>
</html>
```

网页中的效果



2. 毗邻\弟弟

```
毗邻：找div标签下一个标签
    <style>
        p + a {
            color: red;
        }
    </style>
</head>
<body>
<div>
    <a > 这是一个div标签嵌套的a标签</a>
    <p> 这是一个div标签嵌套的p标签</p>
    <a> 这是一个div标签嵌套的a12标签</a>
```

```
</div>
</body>
</html>
```

```
    <style>
        p + a {
            color: red;
        }
    </style>
</head>
<body>
<div>
    <a > 这是一个div标签嵌套的a标签</a>
    <p> 这是一个div标签嵌套的p标签</p>
    <a> 这是一个div标签嵌套的a12标签</a>
</div>
</body>
</html>
```

这是一个div标签嵌入的a标签

这是一个div标签嵌入的p标签

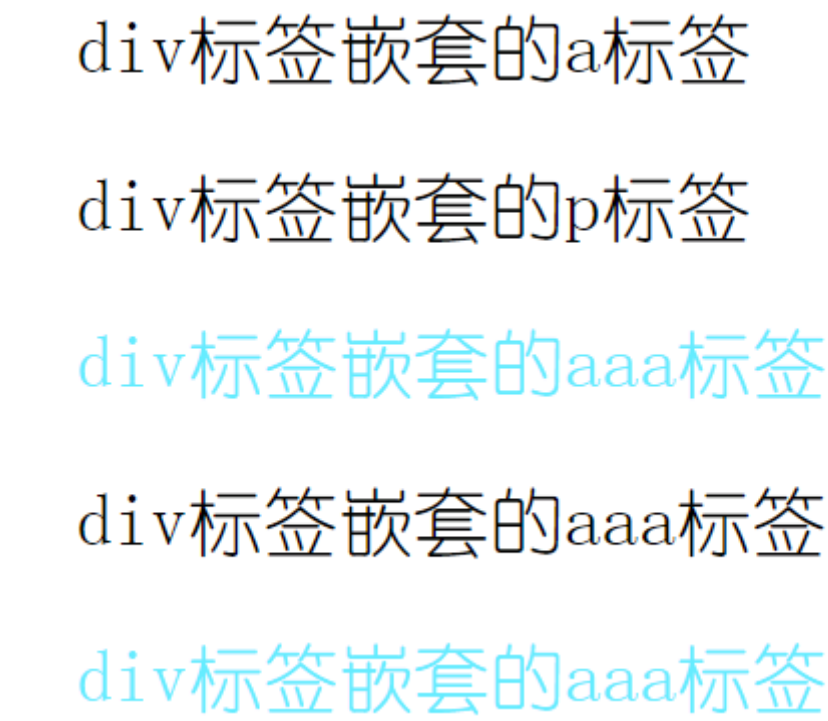
这是一个div标签嵌入的a12标签

弟弟：找div标签后面所有的p标签

```
 <style>
        p ~ a {
            color: #65ecff;
        }
    </style>
</head>
<body>
<div>
    <a>div标签嵌套的a标签</a>
    <p>div标签嵌套的p标签</p>
    <a>div标签嵌套的aaa标签</a>
    <p>div标签嵌套的aaa标签</p>
    <a>div标签嵌套的aaa标签</a>
</div>

</body>
</html>
```

```
    <style>
        p ~ a {
            color: #65ecff;
        }
    </style>
</head>
<body>
<div>
    <a>div标签嵌套的a标签</a>
    <p>div标签嵌套的p标签</p>
    <a>div标签嵌套的aaa标签</a>
    <p>div标签嵌套的aaa标签</p>
    <a>div标签嵌套的aaa标签</a>
</div>

</body>
</html>
```

页面展示效果：



3. 属性

```
属性：指定值的标签
    <style>
        a[href] {     #找属性有href的标签
            color: #65ecff;
        }

        a[href="http://www.mi.com"] {    #找属性有href=xxxx的标签
            color: #2bff30;
        }
```

```
    </style>

</head>
<body>
<div>
    <a href="https://www.mi.com/" target="_blank">小米官网</a>
    <a href="">xxxxxx</a>
    <a>xxxxxx</a>
</div>
</body>
</html>
```

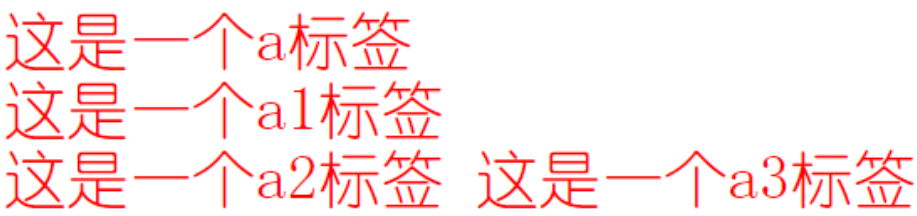
```
    <style>
        a[href] {
            color: #65ecff;
        }

        a[href="http://www.mi.com"] {
            color: #2b1f36;
        }
    </style>

</head>
<body>
<div>
    <a href="https://www.mi.com/" target="_blank">小米官网</a>
    <a href="">xxxxxx</a>
    <a>xxxxxx</a>
</div>
</body>
</html>
```

小米官网 xxxxxx XXXXXX

4. 并集、交集

```
  并集：找所有的标签
    <style>
        a,p {      # a，p所有的标签
            color:red;
        }
    </style>
</head>
<body>
<a>这是一个a标签</a>
<br>
<a>这是一个a1标签</a>
<br>
<a>这是一个a2标签</a>
<a>这是一个a3标签</a>
</body>
</html>
```

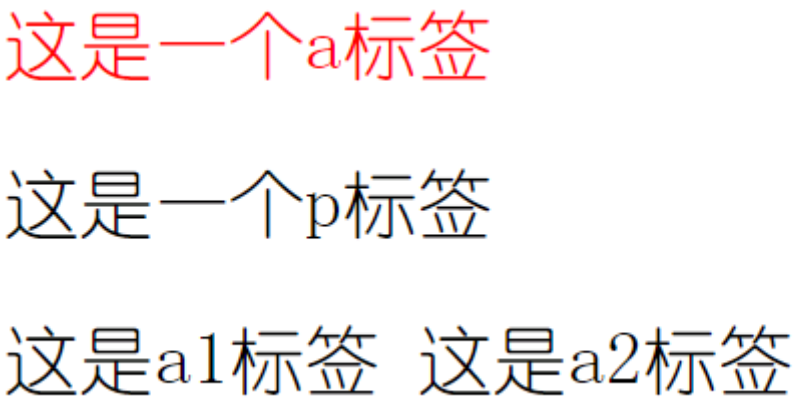
```
    <style>
        a,p {
            color:red;
        }
    </style>
</head>
<body>
<a>这是一个a标签</a>
<br>
<a>这是一个a1标签</a>
<br>
<a>这是一个a2标签</a>
<a>这是一个a3标签</a>
</body>
</html>
```



交集：找a标签里面类为x1的标签

```
<head>
    <style>
        a.x1 {
            color:red;
        }
    </style>
</head>
<body>
<a class="x1">这是一个a标签</a>
<p class="x1">这是一个p标签</p>
<a >这是a1标签</a>
<a >这是a2标签</a>

</body>
</html>
```

```
    <style>
        a.x1 {
            color:red;
        }
    </style>
</head>
<body>
<a class="x1">这是一个a标签</a>
<p class="x1">这是一个p标签</p>
<a >这是a1标签</a>
<a >这是a2标签</a>

</body>
</html>
```

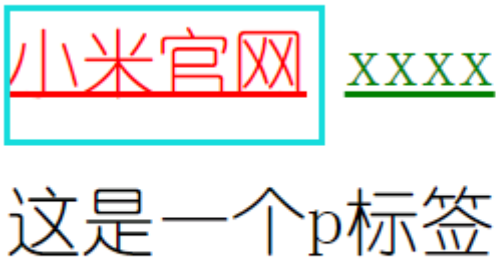


5. 伪类:

```
    <style>
        a:visited {   #访问过后的样式
            color: red;
        }
        a:link {   #没有访问的样式
            color: green;
        }

        a:active {   # 鼠标点击之后的样式
            color: blue ;
        }
        div: hover{   # 鼠标放上去之后会自动改变颜色
            backgroud-color:red
        }
    </style>
</head>
<body>
<a href="http://www.mi.com" target="_blank">小米官网</a>  #  点击完链接之后颜色会变回原来的
<a href="xxx">xxxx</a>
<p>这是一个p标签</p>
```

```
</body>
</html>
```

```
    <style>
        a:visited {
            color: red;
        }
        a:link {
            color: green;
        }

        a:active {
            color: blue ;
        }
    </style>
</head>
<body>
<a href="http://www.mi.com" target="_blank">小米官网</a>
<a href="xxx">xxxx</a>
<p>这是一个p标签</p>
</body>
</html>
```

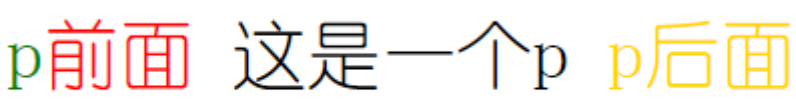


6. 伪元素

```
    <style>
        p:before {
            content: 'p前面';
            color: red;
        }
        p:after {
            content: 'p后面';
            color: gold;
        }
        p:first-letter {
            color: green;
        }
    </style>
</head>
<body>
<p> 这是一个 </p>
</body>
```

```
</html>
```

```
    <style>
        p:before {
            content: 'p前面';
            color: red;
        }
        p:after {
            content: 'p后面';
            color: gold;
        }
        p:first-letter {
            color: green;
        }
    </style>
</head>
<body>
<p> 这是一个 </p>
</body>
</html>
```



css选择器的优先级

1. 行内样式1000> id选择器100> 类选择器10> 标签选择器1> 继承0
2. 优先级可以累加，但是不进位
3. 优先级相同时，后面的样式会应用
4. ！ important 提升选择器的优先级到最高

颜色的表示

1. rgb（三位）：计算机中的三原色是光学三原色，分别是红，绿，蓝
2. 颜色的英文单词表示
3. rgba（四位）：第四位为透明度设置（0-1）0为全透明，

字体

```
font-family: '华文楷体','微软雅黑 Light','宋体';  字体
font-size:16px;     字体大小
font-weight:400;    字体的粗细
```

```
    <style>

        p {
            font-family: '宋体';
        }

        #p1 {
            font-size:100px;
```

```
            font-weight:900;
            font-family: '华文楷体','微软雅黑 Light','宋体';
        }

    </style>
</head>
<body>


<p>这是一个p标签</p>
<p id="p1">这是一个p标签</p>


</body>
</html>
```

1. 网页中不是所有字体都能用哦，因为这个字体要看用户的电脑里面装没装，比如你设置： font-family: "华文彩云"; 如果用户电脑里面没有这个字体，那么就会变成宋体,页面中，中文我们只使用： 微软雅黑、宋体、黑体。 如果页面中，需要其他的字体，那么需要切图。 英语：Arial 、 Times New Roman
2. 为了防止用户电脑里面，没有微软雅黑这个字体。就要用英语的逗号，隔开备选字体，就是说如果用户电脑里面，没有安装微软雅黑字体，那么就是宋体：font-family: "微软雅黑","宋体"; 备选字体可以有无数个，用逗号隔开。
3. 我们要将英语字体，放在最前面，这样所有的中文，就不能匹配英语字体，就自动的变为后面的中文字体： font-family: "Times New Roman","微软雅黑","宋体";
4. 所有的中文字体，都有英语别名，我们也要知道： 微软雅黑的英语别名：font-family: "Microsoft YaHei";宋体的英语别名： font-family: "SimSun";font属性能够将font-size、line-height、font-family合三为一： font:12px/30px "Times New Roman","Microsoft YaHei","SimSun";
5. 行高可以用百分比，表示字号的百分之多少。一般来说，都是大于100%的，因为行高一定要大于字号。 font:12px/200% “宋体” 等价于 font:12px/24px “宋体”; 反过来，比如： font:16px/48px “宋体”;
   等价于 font:16px/300% “宋体”

属性	描述	属性值	说明
font-size	字体大小		
font-weight	字体粗细	none bold border lighter 100~900 inherit	默认值，标准粗细 粗体 更粗 更细 值，400=normal，700=bold 继承父元素字体的粗细值
font-family	字体系列	"Microsoft Yahei","微软雅黑", "Arial", sans-serif	浏览器使用它可识别的第一个值

文本：

```
    <style>
        #p1 {
            background-color:yellow ;
            height: 80px;
            text-align: left;
            /*font-size: 18px;*/
            color: red;
            text-decoration: line-through;
            text-indent:2em;
            line-height: 80px;
        }
        a {
            text-decoration: none;
        }
    </style>
</head>
<body>
<u>这是一个p标签</u>
<s>这是一个p标签</s>
<p id="p1">这是一个p标签</p>
<p style="font-size: 18px">这是一个p标签</p>
<a href="">这是一个a标签</a>
</body>
</html>
```

```
text-align: right;  文本水平排列
text-decoration:   装饰线
underline下划线
line-through 删除线
overline 上划线
text-indent:2em;    文本缩进
em 相对长度单位
1em=当前字体的一个大小
line-height:  行高  调节文字上下居中 行高=块的高度

 文本显示在一行超出部分显示为省略号
/*强制在一行内显示*/
white-space: nowrap;
/*超出部分隐藏*/
overflow: hidden;
/*显示省略符号来代表被修剪的文本*/
text-overflow: ellipsis;
```

文字溢出效果：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>text-overflow</title>
    <style type="text/css">
        .div0 {
            width: 300px;
            border: 1px solid darkblue;
```

```
        }
        .div1 {
            width: 300px;
            border: 1px solid red;

            /*强制在一行内显示*/
            white-space: nowrap;
            /*超出部分隐藏*/
            overflow: hidden;
        }
        .div2 {
            width: 300px;
            border: 1px solid black;

            /*强制在一行内显示*/
            white-space: nowrap;
            /*超出部分隐藏*/
            overflow: hidden;
            /*修剪超出的文本*/
            text-overflow: clip;
        }
        .div3 {
            width: 300px;
            border: 1px solid chocolate;

            /*强制在一行内显示*/
            white-space: nowrap;
            /*超出部分隐藏*/
            overflow: hidden;
            /*显示省略符号来代表被修剪的文本*/
            text-overflow: ellipsis;
        }
    </style>
</head>
<body>
<div class="div0">各国领导人感谢中方作为东道主对各国参展给予的大力支持</div><br>
<div class="div1">各国领导人感谢中方作为东道主对各国参展给予的大力支持</div><br>
<div class="div2">各国领导人感谢中方作为东道主对各国参展给予的大力支持</div><br>
<div class="div3">各国领导人感谢中方作为东道主对各国参展给予的大力支持</div><br>
</body>
</html>
```

文本设置链接：https://www.cnblogs.com/maple-shaw/articles/7146821.html

背景图片：

```
    div {
            /*background-color: red;*/      背景颜色
            /*background-image: url("1.png");*/     背景图片
            /*background-repeat: no-repeat;*/
                图片的重复方式  no-repeat 不重复 repeat 重复  repeat-x  x轴方向重复 repeat-y  y轴方向重复
            /*background-position:right bottom;*/  图片位置
            width: 900px;
            height: 900px;
            background: red url("1.png") no-repeat left center;   综合的写法
        }
```

边框的设置

```
border: 2px;
border-color: black;
border-style: solid(实线);   dashed（虚线）   dotted（实心圆） double（双）
/*  4个 方向  上 右 下 左
            2个   上下  左右
            1个  所有方法
 */
4个方向单独设置
border-left:2px ;
border-left-color:yellow ;
border-left-style:solid ;
统一设置
border: #3bff00 2px solid;
```

块和行内解释和转换

```
行内元素不能设置宽高
块元素可以设置宽高
行内 --》  块   display:block;
块 --》  行内块  display:inline-block;
display: none;  不显示 并且不占位置
```

```
<style>
        span {
            display: block;
            background: #3bff00;
        }
        div {
            width: 100px;
            height: 100px;
            /*display: inline-block;*/
            background: #ff1cf2;
        }

        #div1 {
            display: none;

        }
    </style>
```

```
</head>
<body>

<span>行内</span><span>行内</span><span>行内</span>
<div id="div1">块1</div>
<div>块</div>
```

盒模型

```
<style>

        div {
            background: #3bff00;
            height: 200px;
            width: 200px;
            padding: 20px;
            /*padding-top: 10px;*/
            border: 2px red solid;
            margin: 20px;
        }

    </style>

</head>
<body>

<div>块1</div>
<div style="margin-top: -30px" >块1</div>

</body>
</html>
```

```
padding    内边距
border   边框
margin   外边距

宽度 = width+2*padding + 2*border + 2*margin
塌陷的现象：上下的盒子外边距取最大外边距
```

overflow

```
overflow: visible;   可见   默认
overflow: hidden;   隐藏
overflow: auto;     超出时出现滚动条
overflow: scroll;   显示滚动条
```

浮动：display

```
float: right; left ：  脱离文档流  先浮动的先占位，后面的紧贴着
清除浮动：
.clear{
            clear: both;
        }
```

```
<div id="father" style="">
    <div id="d1" class="box"></div>
    <div class="box"></div>
    <div class="clear"></div>
</div>
伪元素清除法：
.clearfix:after{
      content: '';
      display: block;
      clear: both;
   }
```

定位：position

```
相对定位 relative  相对于原位置进行定位，还占原来的位置
绝对定位 absolute  相对于已经有相对定位的父标签的定位\相对于body的定位 不占原来的位置
固定定位：fixed  相对于窗口的位置
top: ;
left: ;
right: ;
bottom: ;
```

Z-index

```
- z-index 值表示谁压着谁，数值大的压盖住数值小的，
- 只有定位了的元素，才能有z-index,也就是说，不管相对定位，绝对定位，固定定位，都可以使用z-index，而浮动元素不能使用z-index
- z-index值没有单位，就是一个正整数，默认的z-index值为0如果大家都没有z-index值，或者z-index值一样，那么谁写在HTML后面，谁在上面压着别人，定位了元素，永远压住没有定位的元素。
- 从父现象：父亲怂了，儿子再牛逼也没用
```

javascript:是一门编程语言

ECMAScript 标准化的规范

JS引入

```
写在html script标签内部
<script>
    alert('hello,world!')
</script>
```

变量

var是英语“variant”变量的缩写。后面要加一个空格，空格后面的东西就是“变量名”，

定义变量：var就是一个关键字，用来定义变量。所谓关键字，就是有特殊功能的小词语。关键字后面一定要有空格隔开。

变量的赋值：等号表示赋值，将等号右边的值，赋给左边的变量。

变量名：我们可以给变量任意的取名字。

PS：在JavaScript中，永远都是用var来定义变量，这和C、Java等语言不同

```
变量名 数字  字母  下划线 $
使用var 定义变量 var a = 1
var a ；表示声明一个变量 undefined
```

注释

```
当行注释  //
多行注释  /*
*/
```

输出和输入

```
alert()  弹窗
console.log()  控制台输出

prompt('请输入')
"1111111"
var a = prompt('请输入')
```

基本的数据类型

number 数字

表示整数，浮点数 NaN not a number

a.toFixed(2) 保留小数位数 四舍五入

string：字符串

var a = ‘单引号’

var a = ‘双引号’

```
属性：a.length
索引：a[]
按照索引去字符串的值:a.charAt
按照字符串的值找索引：a.indexof
方法:.trim() 左右去空白
    .concat()拼接
    .slice(0，3) 切片
    .split(‘’，3) 分割，数字代表取几个元素
    .toUpperCase():大写
    .toLowerCase():小写
```

布尔值 boolean

true：[] {}

false：‘’ null undefined NaN

空元素null 未定义元素 undefined

var a = null

var a； undefined

数组

var a = 【1，2，3】

var a = new Array（）

```
方法：a.push() 从后边插入数据
      a.unshift() 从前边插入数据
      a.pop() 删除后边你的数据，也可以按照索引删除
      a.shift()从前边删除数据
      a.concat() 拼接
      a.slice() 切片
      a.join()
      a.sort() 排序，默认按照第一个进行排序
      a.reverse() 反转
```

对象

var a = {name:'alex',age:87}
var a = {'name':'alex','age':87}
取值 a['name']
赋值 a['name'] = 'alexdsb'

数据类型的转换

字符串转数字

```
parseInt 字符串转整形
parseFloat 字符串转浮点数
Number（null） 输出的是0
Number（undefined） 输出NaN
```

数字转字符串

```
String （111.111） 输出的是字符串形式的“111.111”
var a = 11
var b = a.toString()
```

```
> parseInt(null)
<- NaN
> parseInt(undefined)
<- NaN
> Number(null)
<- 0
> Number(undefined)
<- NaN
> String(111)
<- "111"
> String(111.1111)
<- "111.1111"
> var a =  11
<- undefined
> var b = a.toString()
<- undefined
> b
<- "11"
>
```

转布尔值Boolean('')

```
true :[] {}

false :0,'',null,undefined
```

```
> Boolean('')
<- false
> Boolean(null)
<- false
> Boolean(0)
<- false
> Boolean(1)
<- true
> Boolean(undefined)
<- false
> Boolean([])
<- true
>
```

运算符

字符串+数字 输出 字符串

字符串-数字 =数字

赋值运算符

```
= += -= *= /=
```

算数运算符

```
+ - * /  %
a++  ++a  自加1
```

比较运算符

```
>  <  >=  <=
==  !=
===  !==
// != NaN 值为true
```

逻辑运算符

```
与 或  非
与：1 && 2
或：false || true
非：  ! true
```

流程控制

格式：条件

　　代码块

if语句

```
if (2>1){
        console.log(1)    输出1
    }

    if (2 < 1) {       输出：1
        console.log(1)
    } else {
        console.log(2)
    }


    if (2 < 1) {      输出：1，3
        console.log(1)
    } else if (2 > 1) {
        console.log(3)
    } else {
        console.log(2)
    }
```

case swith

```
var error_num = 1;

    switch (error_num) {

        case 1:
            console.log(1);
            break;
        case 2:
            console.log(2);
            break;
        default:
            console.log('xxxx');
    }
```

while

```
while(i<=9){ //判断循环条件
    console.log(i);
    i++; //更新循环条件
}
```

do - while 至少执行一次

```
do{

    console.log(i)
    i++;//更新循环条件

}while (i<10)
```

for

```
var a = [1,2]

for (i in a ){
    //  i 索引
    //  a[i]
}

for (var i=0;i<a.length;i++){
   //  a[i]
}
```

三元运算

```
var c = a>b ? a:b  //如果a>b成立返回a，否则返回b
```

函数:

```
python

def 函数名():
```

```
    函数体
    return 1
函数名()

js
function  函数名(参数){
    函数体
    return 返回值

}

arguments 伪数组    接受所有的参数

匿名函数

function (参数){
    函数体
    return 返回值
}

自执行函数
(函数)(参数，参数)

函数
函数（参数）
var c = (function (a,b){
    console.log(arguments)
    console.log(a,b)
    return 1
})(1,2)


    function add(a, b) {
        console.log(arguments);
        console.log(a, b);
        return 1
    }
    add(1.2)

    (function (a, b) {
        console.log(arguments);
        console.log(a, b);
        return 1
    })(1, 2)
```

正则表达式

```
var reg = RegExp('\\d')
var reg = /\d/
reg.test('sss1')   能匹配成功返回true 不能返回false


字符串中使用正则
var a = 'alex is a dsb'
var reg = /a/

a.match(reg)  从左到右匹配一次
```

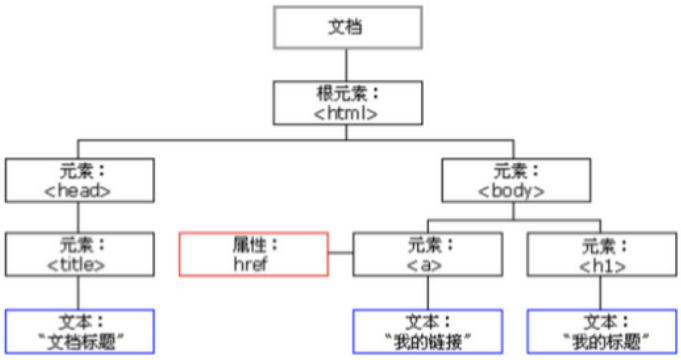
```
a.match(/a/g) g 代表获取所有
a.match(/a/ig)  i 忽略大小写
a.search(/a/)   获取索引值 只找一个  匹配到就是索引 匹配不到就是-1
a.replace(/a/ig,'123')   替换  ig替换所有并且忽略大小写


问题1:
var reg =  /\d/g
reg.test('a1b2c3')
true
reg.test('a1b2c3')
true
reg.test('a1b2c3')
true
reg.test('a1b2c3')
false

问题2
var reg =  /\d/
reg.test()   #不写内容相当于写的undefined
```

DOM document object model 文档对象模型

DOM概念:

文档对象模型。DOM 为文档提供了结构化表示，并定义了如何通过脚本来访问文档结构。目的其实就是为了能让js操作html元素而制定的一个规范。

DOM就是由节点组成的：HTML加载完毕，渲染引擎会在内存中把HTML文档，生成一个DOM树。

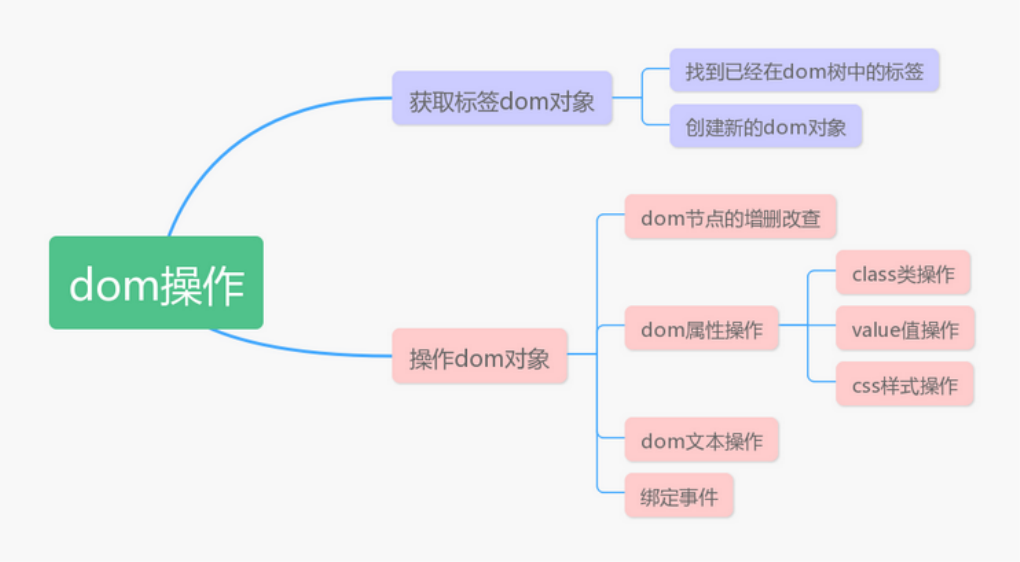


上图可知，**在HTML当中，一切都是节点**：（非常重要）

- **元素节点**：HTML标签。
- **文本节点**：标签中的文字（比如标签之间的空格、换行）
- **属性节点**：：标签的属性。

整个html文档就是一个文档节点。所有的节点都是Object。

dom操作的内容



查找元素

直接查找

```
var div1 = document.getElementById('div1') // 通过ID查找，返回一个节点
var div2 = document.getElementsByClassName('类名') //通过类名查找，返回一个对象。对象中包含多个节点
var div3 = document.getElementsByTagName('标签名') //通过标签名获得标签数组

//其中通过类名和标签名获取的是标签数组，那么习惯性是先遍历之后再使用。
```

```
特殊情况：

即便如此，这一个值也是包在数组里的。这个值的获取方式如下：
document.getElementsByTagName("div1")[0];     //取数组中的第一个元素
document.getElementsByClassName("hehe")[0];   //取数组中的第一个元素
```

间接查找

js中**父子兄**之间的关系如下图：

JS中的**父子兄**访问关系：

父节点	兄弟节点	子节点	所有子节点
parentNode	nextSibling	firstChild	childNodes
	nextElementSibling	firstElementChild	children
	previousSibling	lastChild	
	previousElementSibling	lastElementChild	

```
节点.parentNode  //  找到父节点
节点.children    //  查找子节点，多个节点
节点.firstElementChild  // 找第一个子节点
节点.lastElementChild   //找最后一个子节点
节点.nextElementSibling  // 找下一个兄弟节点
节点.previousElementSibling  // 找上一个兄弟节点
子节点数组 = 父节点.childNodes;   //获取所有节点。
子节点数组 = 父节点.children;   //获取所有节点。用的最多。


节点自己.parentNode.children[index];  //随意得到兄弟节点
```

节点的属性操作

```
节点.getAttribute('属性名') //   获取属性
节点.setAttribute('属性名','值')   // 修改点中属性对应的值
节点.removeAttribute('属性名')     // 删除属性名
```

节点的文本操作

```
节点.innerText  // 标签内的文本内容
节点.innerHTML  // 标签内部的html文本


节点.innerText='设置的文本'     // 标签的文本内容
节点.innerHTML='设置HTML的文本'  // 标签内部的html的文本
```

节点的值

```
//  input  select textarea
节点.value  // 获取节点的值
节点.value = '值'  // 设置节点的值
```

节点样式的操作

```
节点.style // 所有的样式    只有行内样式才能拿多
节点.style.样式名称
节点.style.样式名称    = '值'  // 设置样式
```

节点类的操作

```
节点.classList                    // 节点所有的class
节点.classList.add（'类名'）       // 给节点添加一个类
节点.classList.remove（'类名'）    // 给节点删除一个类
```

节点的操作

```
document.createElement('标签的名字')   //  创建一个标签节点    div  a  span

添加节点到文档中
父节点.appendChild(子节点)   // 添加一个子节点到父节点的后面
父节点.insertBefore(新节点,父节点的子节点)   // 添加一个子节点到父节中的指定子节点前

删除节点
父节点.removeChild(子节点) // 通过父节点删除子节点


替换节点
节点.replaceChild(新节点,节点的子节点)

复制节点
节点.cloneNode()   //  不写数字或者0 只拷贝节点
节点.cloneNode(1)   // 拷贝节点和它的子孙节点
```

事件

事件的绑定

```
// 方式一
<div onclick="alert('你点我了')">
    事件示例
</div>

// 方式二
<div onclick="func()">
    事件示例2
</div>


<script>
    function func() {
        alert('你点我了')
        alert('你还点')
    }

</script>

// 方式三
<div id="div1">
    事件示例3
```

```
</div>

<script>

    var div1 = document.getElementById('div1')
    div1.onclick = function () {
        alert('你点我了');
        alert('你还点');
        alert('你还点!!!');
    }

</script>
```

事件的种类

```
属性    当以下情况发生时，出现此事件
onblur    元素失去焦点
onchange    用户改变域的内容
onclick 鼠标点击某个对象
ondblclick    鼠标双击某个对象
onerror 当加载文档或图像时发生某个错误
onfocus 元素获得焦点
onkeydown    某个键盘的键被按下
onkeyup 某个键盘的键被松开
onload    某个页面或图像被完成加载
onmousemove 鼠标被移动
onmouseout    鼠标从某元素移开
onmouseover 鼠标被移到某元素之上
onmouseup    某个鼠标按键被松开
onreset 重置按钮被点击
onresize    窗口或框架被调整尺寸
onselect    文本被选定
onsubmit    提交按钮被点击
```

事件名	说明
onclick	鼠标单击
ondblclick	鼠标双击
onkeyup	按下并释放键盘上的一个键时触发
onchange	文本内容或下拉菜单中的选项发生改变
onfocus	获得焦点，表示文本框等获得鼠标光标
onblur	失去焦点，表示文本框等失去鼠标光标
onmouseover	鼠标悬停，即鼠标停留在图片等的上方
onmouseout	鼠标移出，即离开图片等所在的区域
onload	网页文档加载事件
onunload	关闭网页时
onsubmit	表单提交事件
onreset	重置表单时

滚动事件

```
<script>

    //实施监听滚动事件
    window.onscroll = function () {
        console.log(1111)
        console.log('上' + document.documentElement.scrollTop)
        console.log('左' + document.documentElement.scrollLeft)
        console.log('宽' + document.documentElement.scrollWidth)
        console.log('高' + document.documentElement.scrollHeight)
    }
</script>
```

BOM bowser object model 浏览器对象模型

窗口

```
打开新窗口
window.open(url,target)
关闭窗口
window.close()  // 只能关闭用open打开的窗口

窗口的宽高
window.innerHeight  高度
window.innerWidth   宽度
```

定时器（重点）

```
定时器 **
// setTimeout  某段时间结束后触发执行
 function func(){
        alert('是否已满18岁')
    }

 setTimeout(func,2000)

// setInterval  设置时间间隔的定时器 每隔一段时间要触发执行函数
var ret= setInterval(func, 500);  // 开启定时器
clearInterval(ret)  // 关闭定时器
```

location（重点）

```
location.href // 当前的地址
location.href = '地址'  // 当前页面会跳转到新页面
```

history

```
history.back()   //后退
history.go()    // 括号里面是0表示当前页面刷新，1表示前进一个网页，-1表示后退一个网页
history.forward()  //表示前进一个网页
```

JQuery

引入方式

```
<script src="jquery.3.4.1.js"> </script>
<script src="https://code.jquery.com/jquery-3.4.1.min.js"> </script>
```

为什么要用JQuery（优点）：

1. 浏览器的兼容性好
2. 隐式循环
3. 链式编程

JQuery对象和DOM对象的关系，装换：

```
JQuery对象内封装了DOM对象，方法

DOM对象 --->（转换）  JQuery对象
JQuery（DOM对象）  /   $(DOM对象)

JQuery方法 --->(转换)  DOM对象
JQuery对象[索引]   /   $对象[]

备注：JQuery和$ 是同一个东西
```

JQuery的选择器

基本选择器

```
$('#id')  id选择器
$('.类名')  类选择器
$('标签名')  标签选择器
$('*')   通用选择器


$('标签名.属性#标签名')   交集选择器
$('标签名1，标签名2，标签名3')   并集选择器
```

层次选择器

```
$('父标签  子标签')   // 选择父标签下的所有子标签，   后代选择器 空格表示
$('ul>a')    // 子代选择器（>），选择ul标签下的所有a标签
$('#l1+li')  // 毗邻选择器（+），选择id为l1标签的下一个li标签
$('#l1~li')  // 弟弟选择器（~），选择id为l1标签后的所有li标签
```

属性选择器

```
$('[属性]') // 查看标签属性
$("标签[属性]")   // 选择某个包含某个属性的标签
$('[属性="值"]')  //
$('[属性$="值"]') //属性结尾为某个值的标签
$('[属性^=“值”]')  // 属性开头为某个值的标签
$('[属性*=“值”]')  // 属性包含某个值的标签
```

JQuery的筛选器

基本筛选器

```
$('选择器:筛选器')
$('选择器:first')
$('选择器:last')
eq(索引)  等于某索引的标签
gt(索引)  大于某索引的标签
lt(索引)  小于某索引的标签
odd     奇数
even    偶数
not(选择器)  // 在当前的标签内排除某些标签
```

type筛选器

```
$(':text')
$(':radio')
$(':checkbox')
$(':password')
$(':submit')
$(':button')
$(':file')

注意： date 不支持
```

状态选择器

```
$(':disabled')   // 禁用的标签
$(':enabled')   //  可使用的标签
$(':checked')    // radio、checkbox、select（option） 选中的标签
$(':selected')   // select 选中的option标签
```

JQuery筛选器方法

```
.children()  // 子代
.parent()   //  父代
.parents()   // 父辈们
.parentsUntil('选择器')   // 找父辈们，直达某个标签，且不包含该标签


.siblings()  // 所有兄弟标签
.prev()   // 上一个
.prevAll()   // 前面所有的兄弟
.prevUntil()   // 前面的兄弟到某个标签位置


.next()   // 下一个
.nextAll()   // 后面所有的兄弟
.nextUntil()   // 后面的兄弟到某个标签位置
```

```
first()
last()
eq()
has()  //  $('li').has('a')     选择一个包含a标签的li标签
find()  //  $('li').find('a')  在li标签下找所有的a标签
not()  //  在所有的标签内排除某些标签
filter()  // 获取一些满足条件的标签  交集选择器
```

补充内容

循环多个元素时

```
var ul_li =$('li')
for (let i=0;i<ur_li.length;i++){   // let声明的变量只在代码块中生效
    console.log('li',ur_li[i])
}
```

局部变量和全局变量

```
function f(){
    var b = 2;
    let c = 3;
    d = 4;
}
```

事件

```
click(function(){...})    // 点击事件

focus(function(){...})   //  获取焦点
blur(function(){...})    // 失去焦点


change(function(){...})  //内容发生变化，input（鼠标移出），select等

keyup(function(){...})    //键盘按下去之后的变化，

            $('#i1').keyup(function (e) {
        console.log(e.keyCode)    //要是想看key就使用keyCode
    })

mouseover/mouseout    鼠标的变化
mouseenter/mouseleave  =   hover(function(){...})  //鼠标进入和离开
```

```
click(function(){...})    // 点击事件

focus(function(){...})   //  获取焦点
blur(function(){...})    // 失去焦点


change(function(){...})  //内容发生变化，input（鼠标移出），select等

keyup(function(){...})

mouseover/mouseout
mouseenter/mouseleave  =   hover(function(){...})
```

事件的绑定

```
<button>按钮1</button>
<button>按钮2</button>

<script>
```

```
    $('button').click(function () {
        alert('你点我干啥')
        alert('点你咋地')
        alert('再点一个试试')
    })

</script>


// bind
 $('button').bind('click',{'a':'bb'},fn);  //   事件类型 参数 函数

    function fn(e) {    //   e 事件的对象
        console.log(e);
        console.log(e.data);   //  传的参数
        // alert(123)
  }

 $('button').bind('click',fn);

    function fn(e) {
        console.log(e);
    }

// 事件

$('button').click({'a': 'b'}, fn)  //  参数  函数

    function fn(e) {
        console.log(e.data);
    }

$('button').click( fn)

    function fn(e) {
        console.log(e.data);
    }
```

事件的解除

```
$('button').unbind('click')
```

JQuery对象的操作

文本的操作

```
.text()     // 获取标签的内容
.text(内容)  // 设置标签的内容


.html()        // 获取标签的内容
.html(HTML标签文本)  // 设置标签的内容
.html(DOM对象)  // 设置标签的内容
.html(jq对象)  // 设置标签的内容
```

标签的操作

```
添加   //  js  父节点.appendChild(新的节点)  父节点.insertBefore(新的节点,参考节点)

父子关系：
    添加到后面
    a.append（b）       在a节点孩子中添加一个b
    // a 父节点   b 新添加的子节点
    // a 必须是jq对象 b   （标签字符串、dom对象、jq对象 ）

    a.appendTo(b)     把a节点添加到b的孩子中
    // b 父节点   a 新添加的子节点
    // a 必须是jq对象   b   （选择器、dom对象、jq对象 ）
    添加到前面
    a.prepend(b)   b.prependTo(a)


兄弟关系：
    a.after(b)     在a的后面添加一个b    b.insertAfter(a)    把b添加到a的后面
    a.before(b)    在a的前面添加一个b    b.insertBefore(a)   把b添加到a的前面


// 操作同一个对象时，操作多次，相当于是移动的效果。
```

```
删除 remove detach empty

remove //删除标签并且删除了事件，返回值是标签对象
detach //删除标签，但保留了标签内的事件，返回的是标签对象
empty  //清空标签内容，但是保留标签本身


替换 replaceWith  replaceAll

a.replaceWith(b)  // 用b标签替换a标签
a.replaceAll(b)   // 用a标签替换所有的b标签（选择器，jq对象，dom对象）


拷贝
clone（false）  // 只拷贝标签
clone（true）   // 拷贝标签也拷贝事件
```

JQuery属性的操作

普通属性

```
.attr('属性')   //获取属性的值
.atr('属性','值') //设置单个属性的值
.attr({'属性':'值','属性2':'值'})  // 设置多个属性的值


removeAttr('属性')   //删除单一属性
```

值

```
input select teatarea
使用
val()  //获取值
val('值')  //设置值

radio checkbox select 是否选中使用prop ，不建议使用attr
.pop('属性')  // 获取属性的值
.prop('checked',true) // checked变为选中状态
.prop('checked',false)  // checked 变为未选中状态
```

类

```
addClass()   // 添加类
removeClass()  // 删除类
toggleClass()  // 切换类
```

css

```
.css（'css样式名'）    // 获取对应样式的值
.css（'css样式名','值'）    // 设置对应样式的值
.css（{'css样式名':'值','css样式名2':'值'}）    // 设置对应样式的值
```

盒子模型

```
width()   //  内容的宽
height()  //  内容的高

innerHeight()    // 内容 + 内边距  高
innerWidth()     // 内容 + 内边距  宽

outerWidth()    //  内容 + 内边距  + 边框  宽
outerHeight()   //  内容 + 内边距  + 边框  高

outerHeight(true)    //  内容 + 内边距  + 边框 + 外边距 高
outerWidth(true)   //  内容 + 内边距  + 边框 + 外边距 宽

设置宽高的时候 只是设置内容的宽高
```

动画效果

```
滑动系列
slideDown       // 向下划入
slideUp         // 向上划入
slideToggle    切换
slideDown（毫秒数，回调函数）

显示系列
show   hide  toggle

渐入渐出
fadeIn   渐入
fadeOut  渐出
fadeToggle  切换
```

```
stop()  停止之前的动画
```

模态框示例

```
<script src="jquery.3.4.1.js"></script>

    <style>
        .mask {
            position: absolute;
            top: 0;
            left: 0;
            background-color: rgba(127, 127, 127, 0.5);
            width: 100%;
            height: 100%;
            display: none;
        }
        .model {
            position: absolute;
            top: 50%;
            left: 50%;
            width: 500px;
            height: 400px;
            background-color: white;
            margin-top: -200px;
            margin-left: -250px;
        }
    </style>
</head>
<body>

<h3> 模态框示例 </h3>
<button id="b1">显示</button>     //显示按钮

<div>
    <div class="mask">
        <div class="model">
            <input type="text">
            <input type="password">
        </div>
    </div>
</div>
    <script>
        $('#b1').click(
            function () {
                $('.mask').show()
            }
        );

        $(window).keyup(function (e) {    //esc键退出模态框

            if (e.keyCode === 27) {      //27指的是esc键的ascii码
                $('.mask').hide()
            }

        })
    </script>
```

```
js
window.onload = function () {  //  页面 图片 视频 音频 都加载好执行
            $('#b1').click(
                function () {
                    $('.mask').show()
                }
            );
        }

//  window.onload  只执行一次  多次的划  后面的覆盖前面的

jquery
 $(window).ready(function () {  //  页面 图片 视频 音频 都加载好执行
            $('#b1').click(
                function () {
                    $('.mask').show()
                }
           );

  })
  // $(window).ready()  可执行多次  不会覆盖


  $(document).ready(function () {  //  页面  都加载好执行
            $('#b1').click(
                function () {
                    $('.mask').show()
                }
            );
            })
        })

//  简写
    $(
           function () {  //  页面  都加载好执行
                $('#b1').click(
                    function () {
                        $('.mask').show()
                    }
                );


        )
```

each对比for循环

```
$('li').each(function (i,dom) {
        console.log(i,dom)
        console.log(i,dom.innerText)
})
```

事件冒泡

触发子代的事件，父辈的事件也会依次执行

触发子代的事件，父辈的事件也依次执行。

阻止事件冒泡：

return false

e.stopPropagation()

事件委托

利用事件冒泡的原理，将子代的事件委托给父代执行。

```
 $('tbody').on('click','button',function () {
     console.log(this)
})
```

bootstrap

https://v3.bootcss.com/

http://www.fontawesome.com.cn/