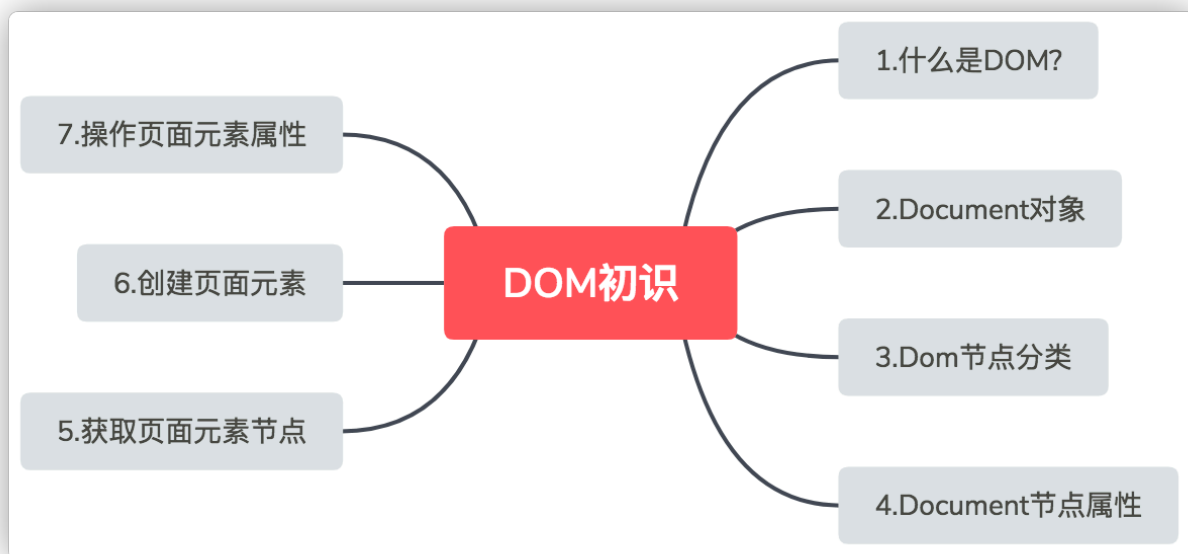


# DOM初识

---

## 0.1. 本节主要内容：

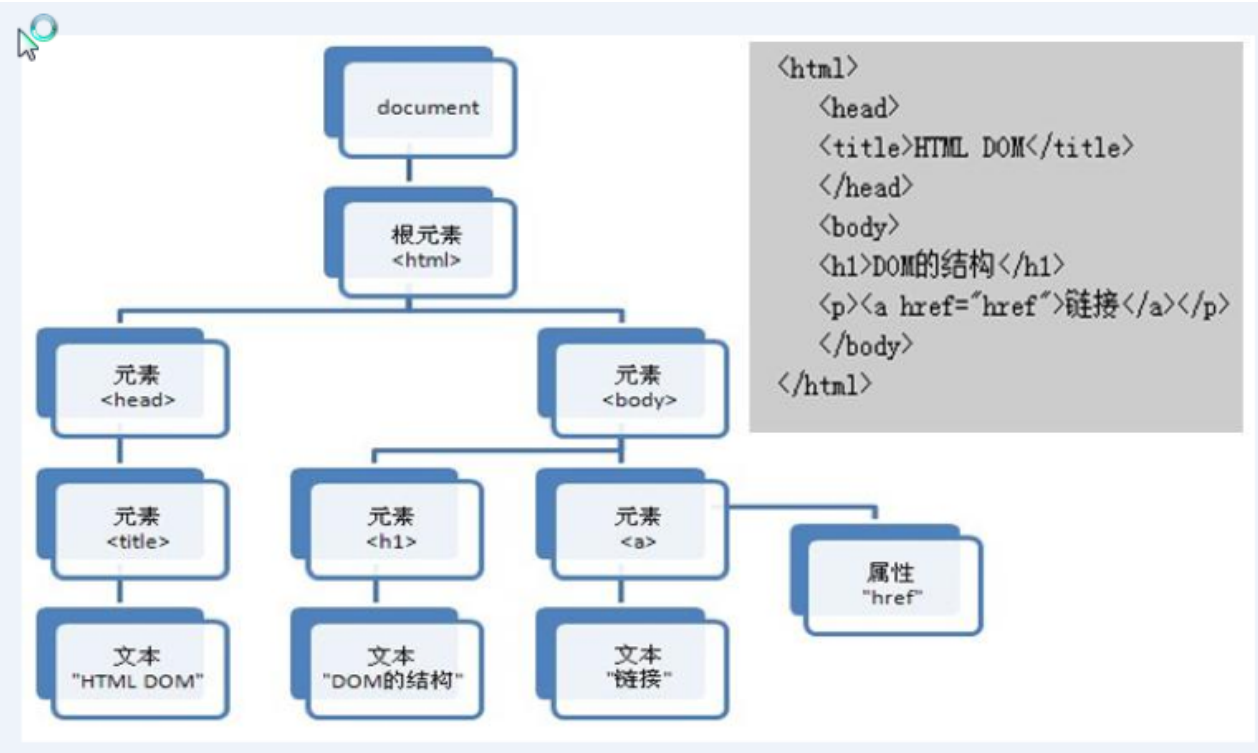


## 0.1. 学习目标：

节数	知识点	要求
第一节 什么是DOM?	DOM的介绍	了解
第二节 Document对象	Document对象介绍	了解
第三节 Dom节点分类	Dom节点分类	掌握
第四节 Document节点属性	Document节点属性	掌握
第五节 获取页面元素节点	获取页面元素节点	掌握
第六节 创建页面元素	创建页面元素	掌握
第七节 操作页面元素属性	操作页面元素属性	掌握

# 1. 什么是DOM?

DOM **D**ocument **O**bject **M**odel，[文档对象模型](#)，DOM可以以一种独立于平台和语言的方式访问和修改html文档的内容和结构。



## 2. Document对象

---

**Document 对象是 HTML 文档的根节点。**

Document 对象使我们可以从脚本中对 HTML 页面中的所有元素进行访问。

Document 对象是 Window 对象的一部分，可通过 window.document 属性对其进行访问

## 3. DOM节点分类

---

在 HTML DOM (Document Object Model) 中，每一个元素都是节点：

文档是一个文档节点。

所有的HTML元素都是元素节点。

所有 HTML 属性都是属性节点。

文本插入到 HTML 元素是文本节点。are text nodes。

注释是注释节点。

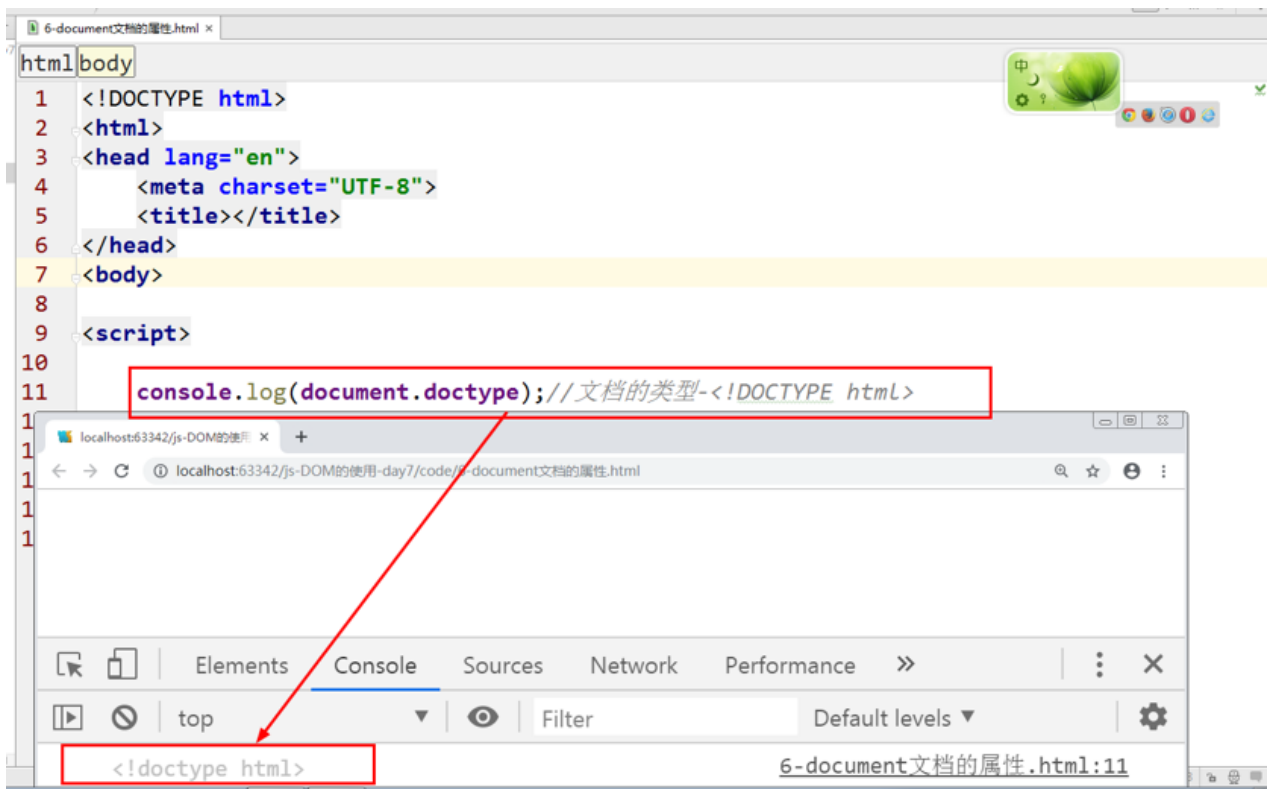
## 4. Document节点属性

---

### 4.1. 返回文档内部的某个节点

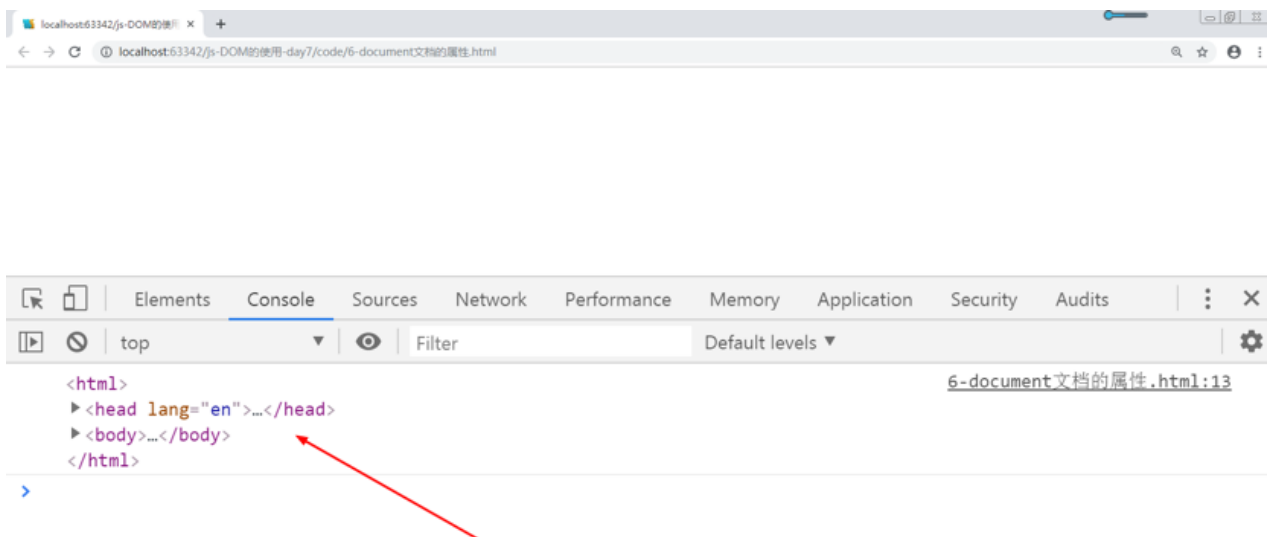
**doctype,documentElement,body,head**

1. doctype



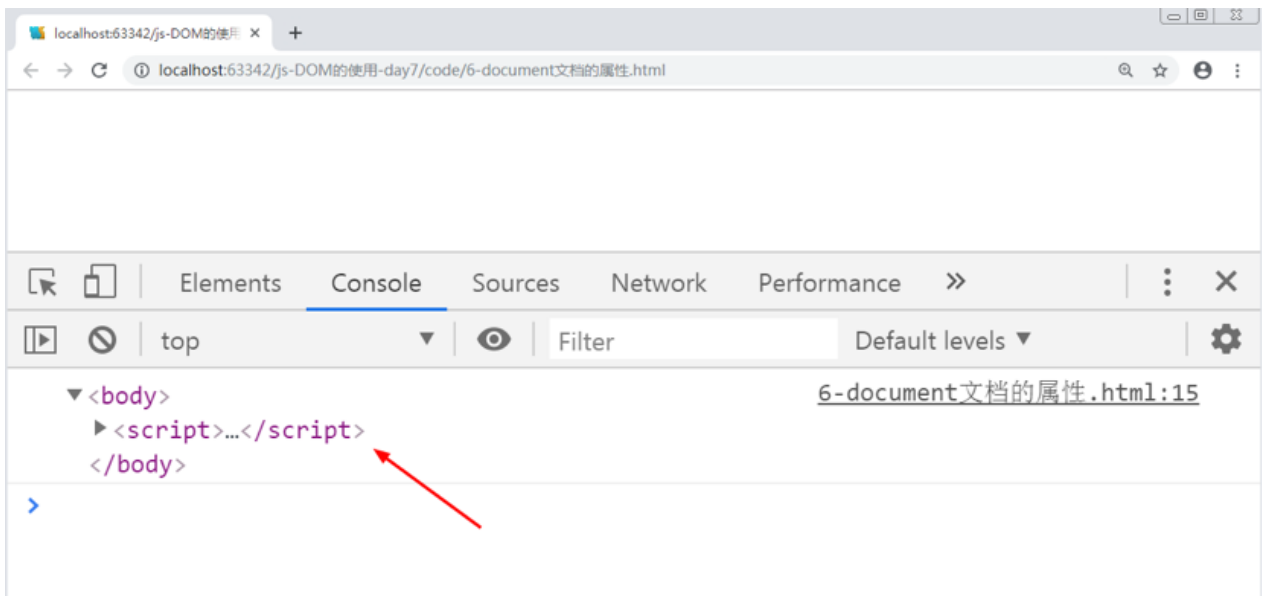
## 2. documentElement

```
12
13 console.log(document.documentElement); // 返回的是整个的html文档结构
14
15
```



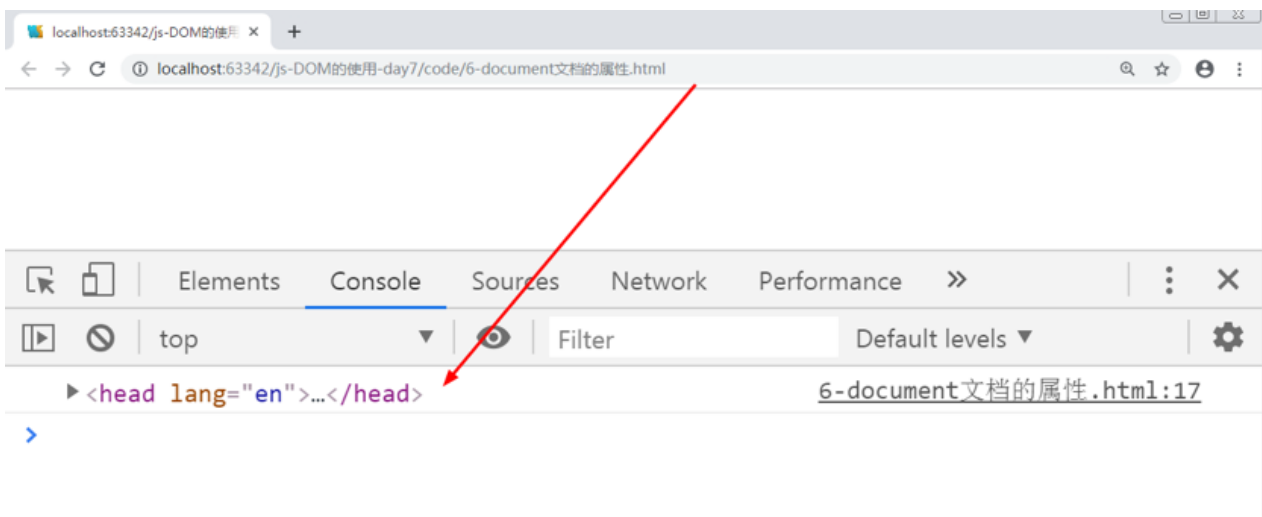
## 3. Body

```
14
15 console.log(document.body); // 返回html文档的body部分
16
```



## 4. Head

```
16  
17 console.log(document.head); // 返回html文档的head部分  
18 </script>
```

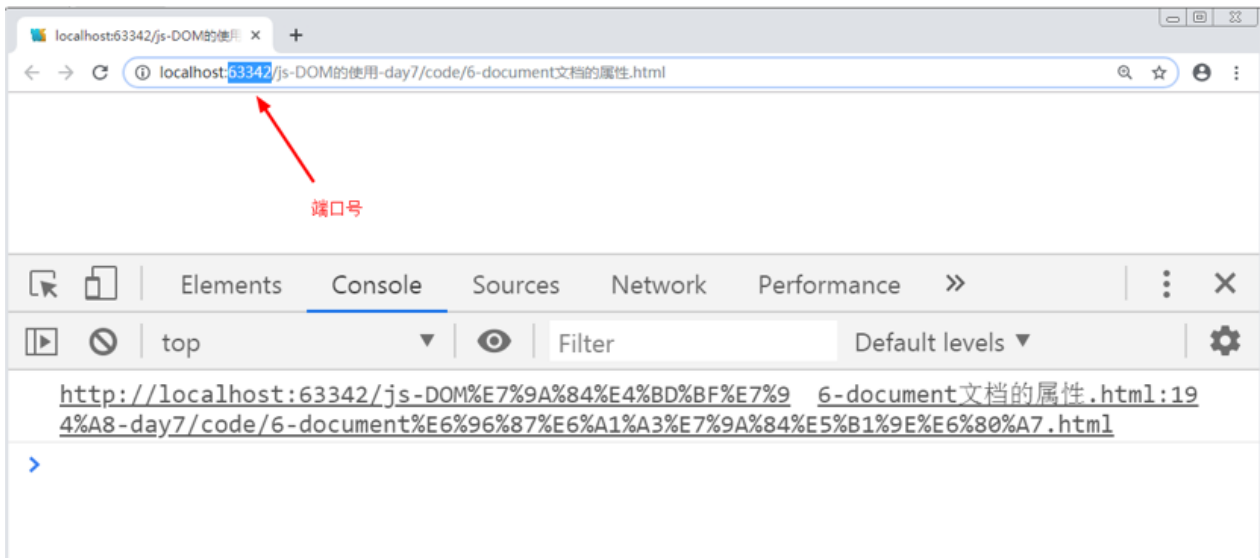


## 4.2. 返回文档指定信息

documentURI, URL, domain, lastModified, location, title, readyState属性

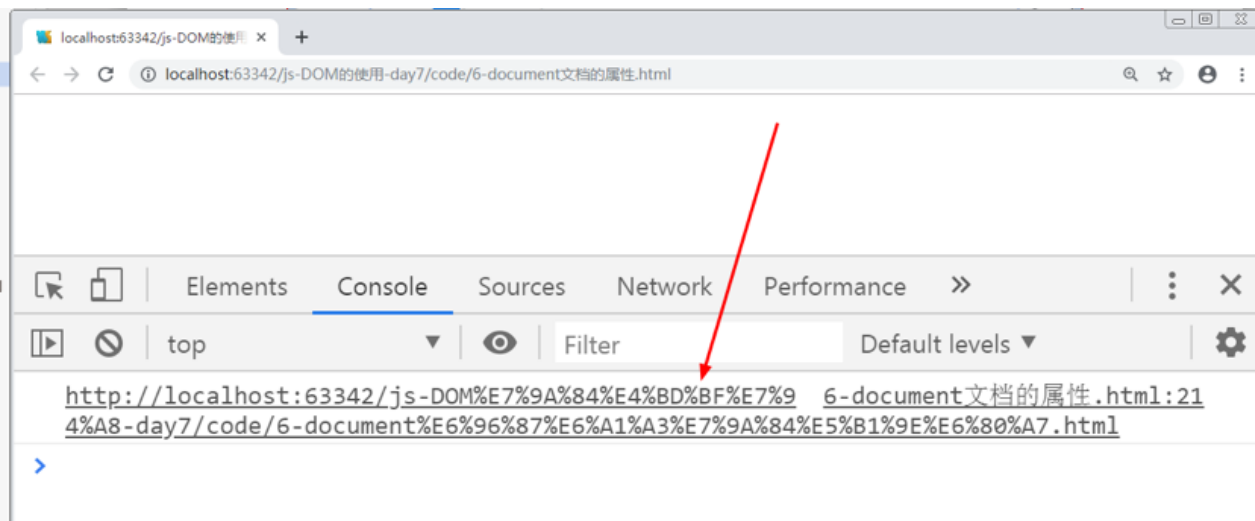
### 1. documentURI:返回当前的网址 (url)

```
18  
19 console.log(document.documentURI); // 返回当前的url  
20
```



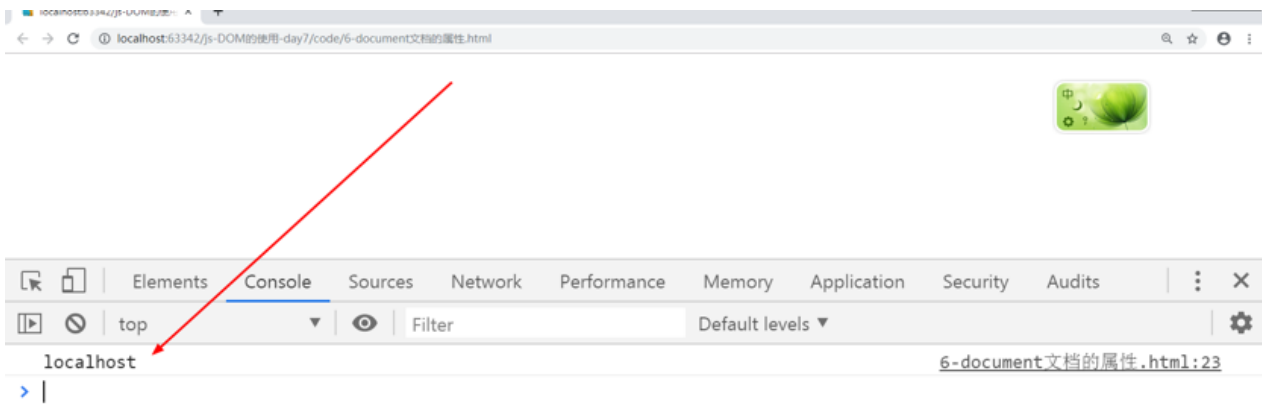
## 2. URL

```
20  
21 console.log(document.URL);//返回当前的url  
22  
23
```



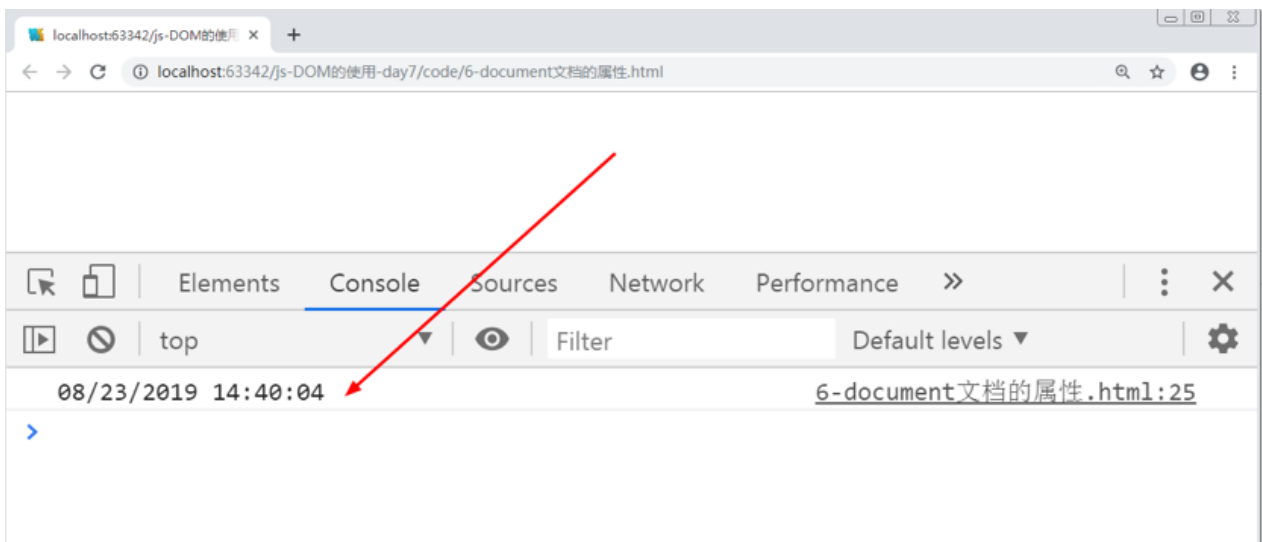
## 3. domain:返回当前的域名

```
22  
23 console.log(document.domain);//返回当前的域名  
24  
25  
26
```



lastModified

```
24  
25 console.log(document.lastModified);//返回当前的最新的修改时间  
26  
27
```



Location:

location.assign('传递一个url');

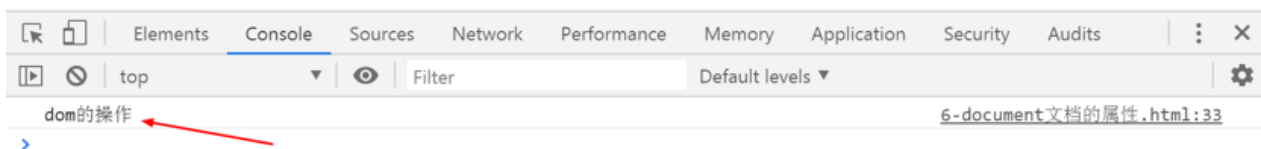
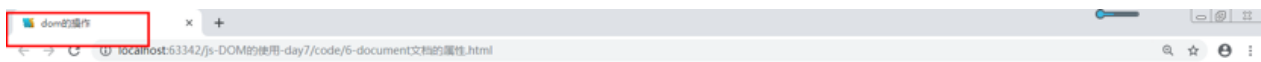
window.location = '传递一个url';

location.href = '传递一个url';

```
26  
27 location.assign("http://www.huawei.com");//方式1-跳转网址  
28  
29 window.location = "http://www.bjsxt.com";//方式2-跳转网址  
30  
31 location.href = "http://www.tmall.com";//方式3-跳转网址  
32  
33
```

Title:

```
32  
33 console.log(document.title); // 返回文档标题  
34 </script>  
35
```



readyState

**readyState**属性返回当前文档的状态。

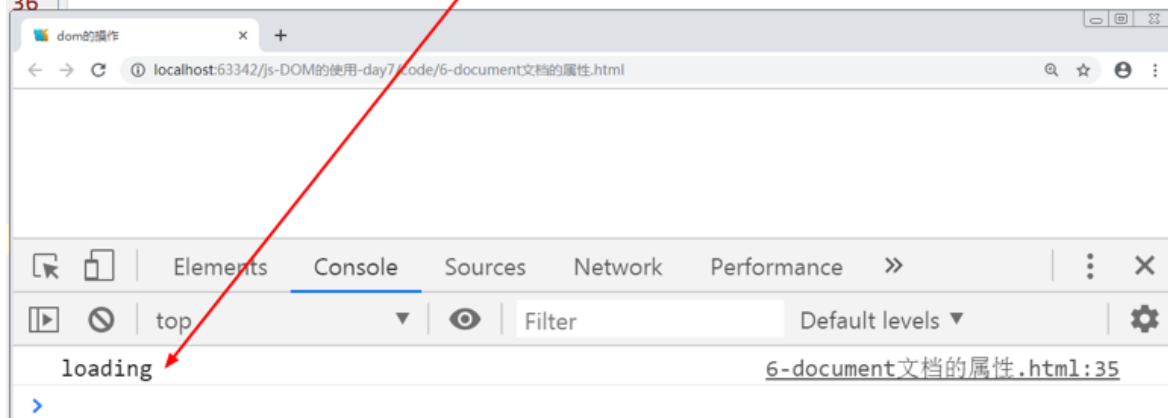
共有三种可能值：

加载HTML代码阶段（尚未完成解析）是“loading”，

加载外部资源阶段是“interactive”，

全部加载完成是“complete”。

```
34  
35 console.log(document.readyState); // Loading  
36
```

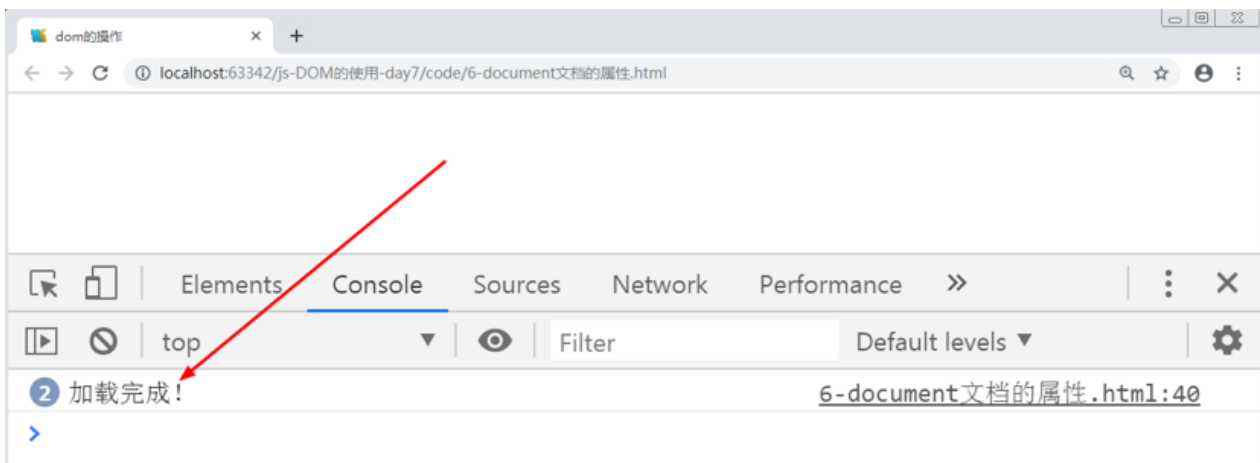




```

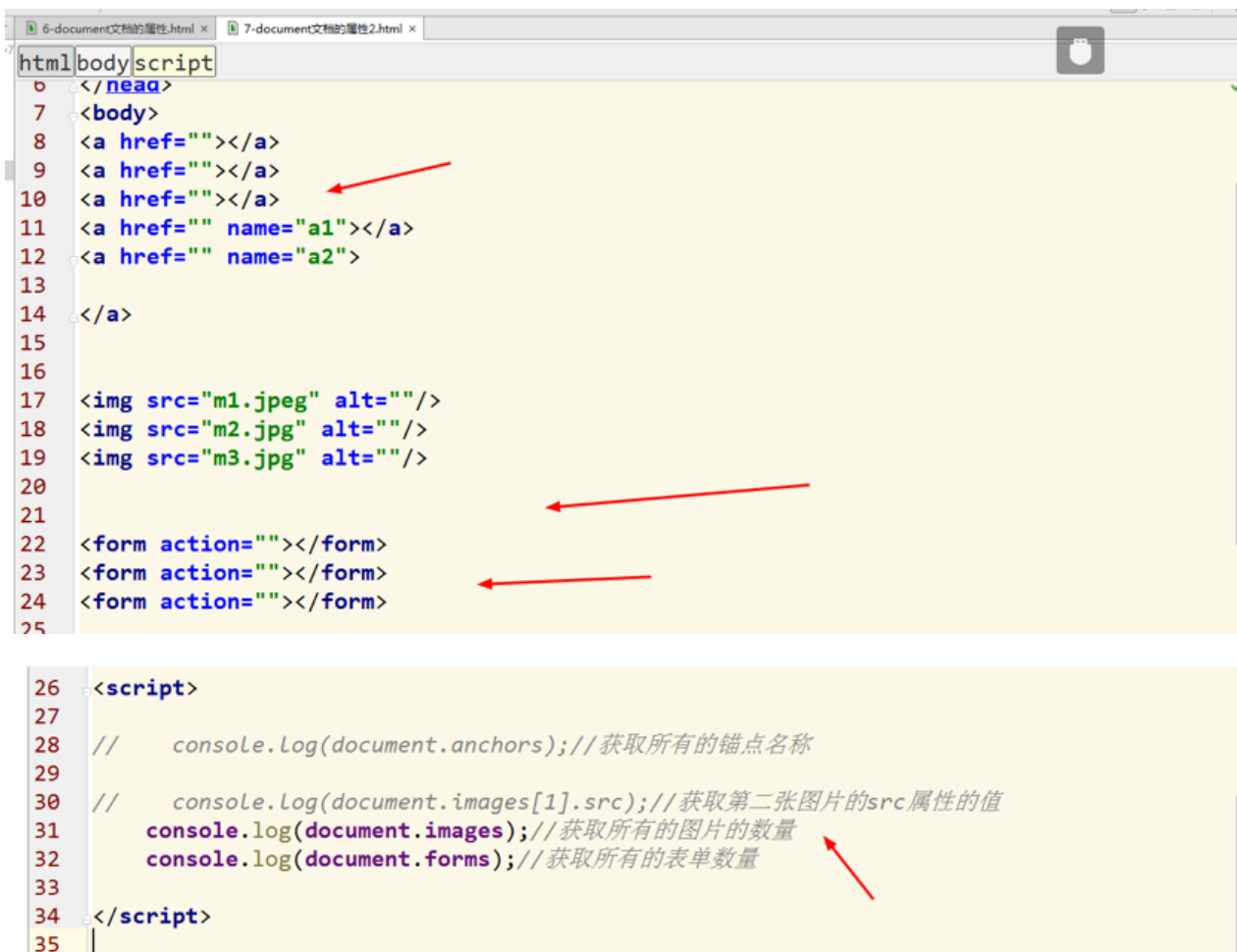
36
37 // 异步加载完成
38 setInterval(function(){
39     if(document.readyState == "complete"){
40         console.log("加载完成!");
41     }
42 },1000)
43
44 </script>

```



## 4.3. 返回文档内部特定节点的集合

anchors,forms,images,links,scripts



## 5. 获取元素节点（重点）

---

### 5.1. getElementById()

getElementById():通过标签的**id**属性获取元素

### 5.2. getElementsByTagName()

getElementsByTagName():通过**标签名**来获取元素。（数组）

### 5.3. getElementsByName()

getElementsByName():通过标签的**name**属性获取元素。（数组），

### 5.4. getElementsByClassName()

getElementsByClassName():通过标签的**class**属性来获取元素。（数组），有浏览器兼容性，主要是ie8以下。

### 5.5. querySelector()

querySelector():通过css选择器来获取元素

### 5.6. querySelectorAll()

querySelectorAll():通过css选择器来获取元素(数组)

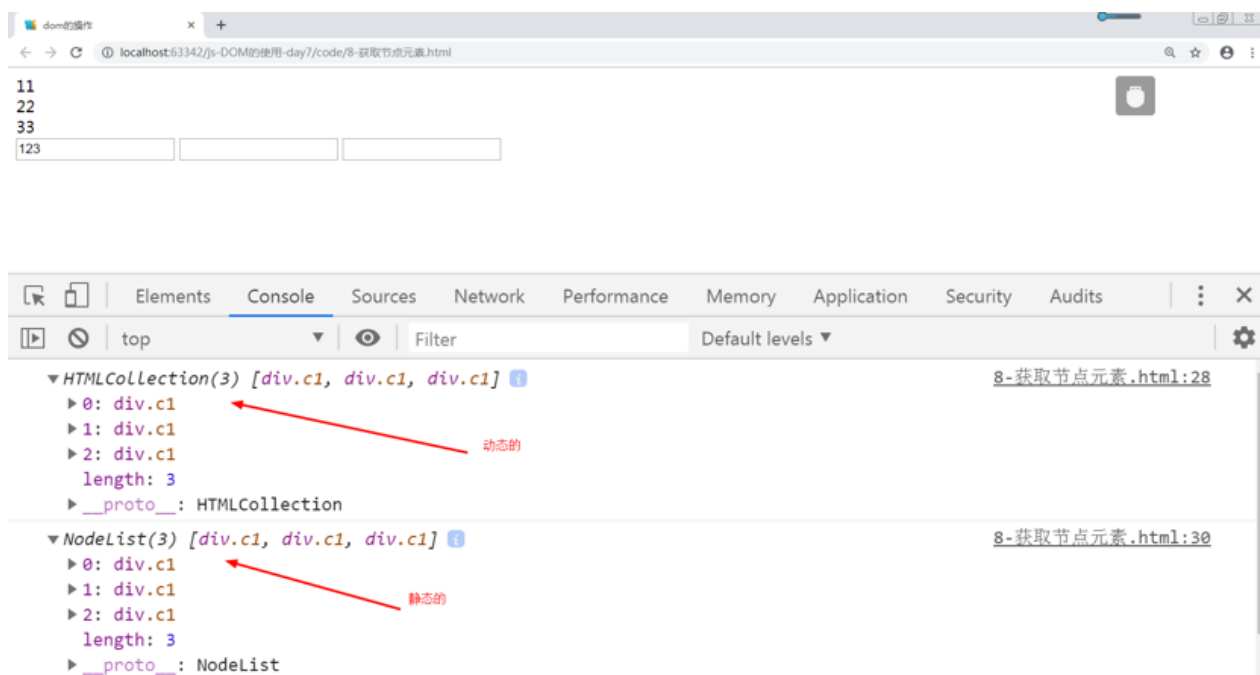
### 5.7. getElement和querySelector的区别

**query**选择符选出来的元素及元素数组是**静态的**，而**getElement**这种方法选出的元素是**动态的**。静态的就是说选出的所有元素的数组，不会随着文档操作而改变。在使用的时候getElement这种方法性能比较好，query选择符则比较方便。

1.得到的元素不是需要很麻烦的多次getElementBy..的话，尽量使用getElementBy..,因为他快些。

2.得到的元素需要很麻烦的多次getElementBy..组合才能得到的话使用querySelector，方便。

3.看实际情况，你决定方便优先还是性能优先。



创建li元素，getElement创建元素，会造成死循环，而使用querySelector创建的li元素是可以的

```
9-getElement和querySelector的区别.html x
html body
9 <ul>
10 <li>11</li>
11 <li>22</li>
12 <li>33</li>
13 </ul>
14
15 <script>
16
17 // var oul = document.getElementsByTagName("ul")[0];
18 //
19 // var oli = document.getElementsByTagName("li");//数组
20
21 var oul = document.querySelectorAll("ul")[0];
22
23 var oli = document.querySelectorAll("li");//数组
24
25 for(var i = 0 ; i<oli.length;i++){
26
27     oul.appendChild(document.createElement("li"));//给ul添加li元素
28
29 }
30
31 </script>
```

## 5.8. 实例：通过单击修改一个div块的内容

## 5.9. 实例：选项卡的制作

```
10-选项卡的制作.html x
html body script
61 //var:它是全局作用域，指向最后i的值10，而let块作用域，当前的块内有效，所有是6
62
63 var a = [];
64
65 for(var i = 0;i<10;i++){
66     a[i] = function(){
67         console.log(i);
68     }
69 }
70
71 a[6]()//10
72
73
74 var a = [];
75
76 for(let i = 0;i<10;i++){
77     a[i] = function(){
78         console.log(i);
79     }
80 }
81
82 a[6]()//6
83 </script>
```

普通实现：

```

6   <style>
7
8   body{
9       font-size: 32px;
10  }
11  button{
12      background: green; /*默认绿色*/
13  }
14  .active{
15      background: red; /*单击变成黄色*/
16  }
17  div{
18      width: 300px;
19      height: 200px;
20      background: gray;
21      display: none;
22  }
23
24  </style>

```

```

26  <body>
27
28      <button class="active">1</button>
29      <button>2</button>
30      <button>3</button>
31
32      <div style="display: block">aaa</div>
33      <div>bbb</div>
34      <div>ccc</div>
35

```

```

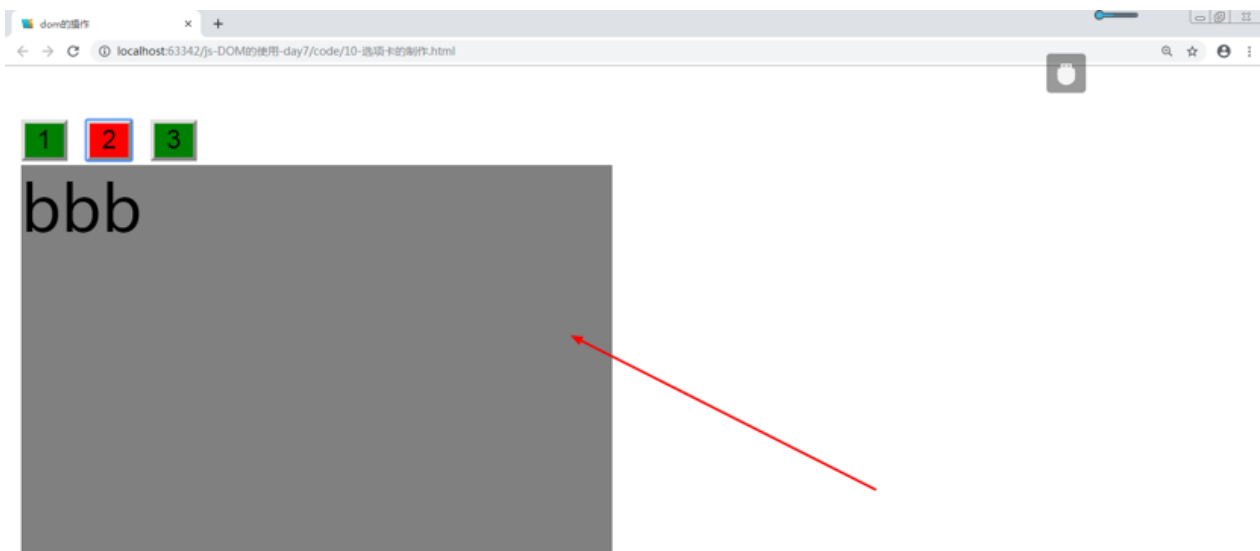
36  <script>
37
38      var obt = document.querySelectorAll("button");//获取所有的按钮元素
39
40      var odiv = document.querySelectorAll("div");//获取所有的div元素
41

```

```

58
59  for(var i = 0;i<obt.length;i++){
60
61      obt[i].index = i; //将当前的i 保存到按钮的index属性中
62
63      obt[i].onclick = function(){
64          //内循环将所有的按钮button和div都不应用样式
65          for(var i = 0;i<obt.length;i++){
66              obt[i].className = "";
67              odiv[i].style.display = "none";
68          }
69          //只应用当前的
70          this.className = "active"; //给当前的按钮添加样式-yellow
71
72          odiv[this.index].style.display = "block"; //当前的div显示
73
74      }
75
76  }
77

```



使用let实现



## 5.10. 作业：闭包实现选项卡

# 6. 创建页面元素（重点）

## 6.1. createElement(): 创建元素节点

## 6.2. createTextNode(): 创建文本节点

```
13
14 <script>
15
16     var op = document.createElement("p");// 创建p元素
17
18     var otxt = document.createTextNode("暖风吹的游人醉");// 创建文本节点
19
20     op.appendChild(otxt);// 给p元素添加文本内容
21
22     document.body.appendChild(op);// 将p元素添加body元素中
23
24
```



## 6.3. createAttribute(): 创建属性节点

```
13
14 <script>
15
16     var op = document.createElement("p");// 创建p元素
17
18     var otxt = document.createTextNode("暖风吹的游人醉");// 创建文本节点
19
20     op.appendChild(otxt);// 给p元素添加文本内容
21
22     var ostyle = document.createAttribute("style");// 创建一个style属性
23
24     ostyle.value = "font-size:60px;color:red;font-weight:bold;font-style:italic;";
25
26     op.setAttributeNode(ostyle);// 给p元素添加style样式属性
27
28     document.body.appendChild(op);// 将p元素添加body元素中
29
30
31
32 </script>
```



## 6.4. className的使用

## 6.5. 实例：创建一个菜单

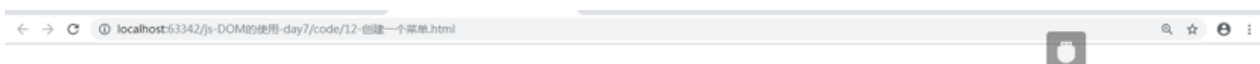
创建哪些元素

UI ,li,

设置属性：样式

```
12-创建一个菜单.html x
html body script
1 <!DOCTYPE html>
2 <html>
3 <head lang="en">
4   <meta charset="UTF-8">
5   <title>dom的操作</title>
6   <style>
7
8     li:hover{
9       background:tomato!important;
10    }
11
12  </style>
13 </head>
14 <body>
15
```

```
12-创建一个菜单.html x
html body script
18
19 var arr = ["首页","百度","华为","京东","淘宝"];
20
21 var oul = document.createElement("ul");//创建ul元素
22
23 var ostyle = document.createAttribute("style");//创建属性
24
25 ostyle.value = "list-style:none";
26
27 oul.setAttributeNode(ostyle);//给ul添加style样式属性
28
29 for(var i=0;i<arr.length;i++){
30
31   var oli = document.createElement("li");//创建li元素
32
33   oli.innerHTML = arr[i];//给li元素添加内容
34
35   oli.style.cssText = "float:left;color:snow;width:100px;height:39px;line-height:
36
37   oul.appendChild(oli);//将li添加到ul中
38 }
39
40 document.body.appendChild(oul);//将ul添加到body中
41
```



首页

百度

华为

京东

淘宝

## 7. 操作页面元素属性

### 7.1. 元素属性节点的操作

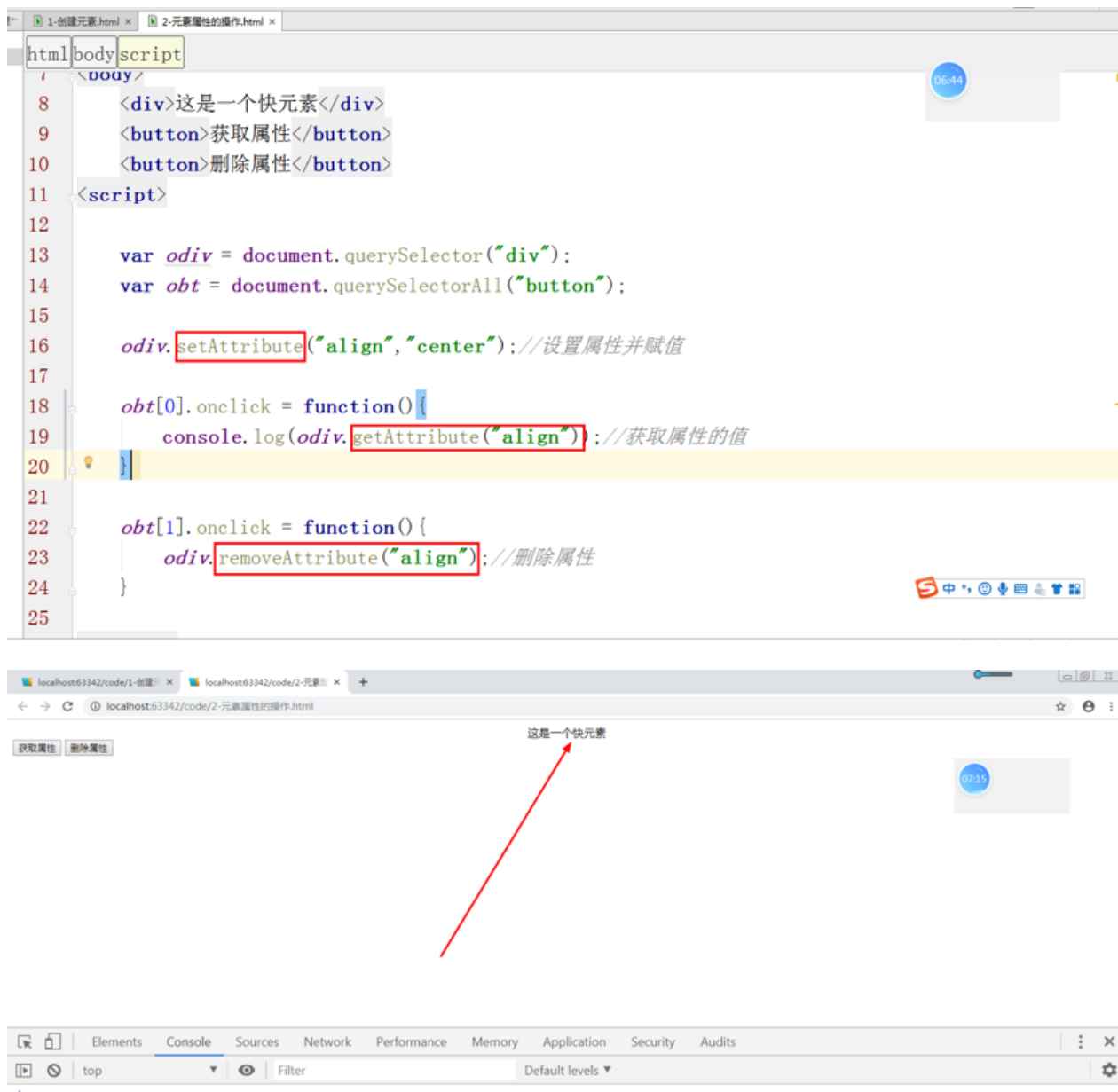
语法:



setAttribute('属性名','属性值')：给节点元素设置属性

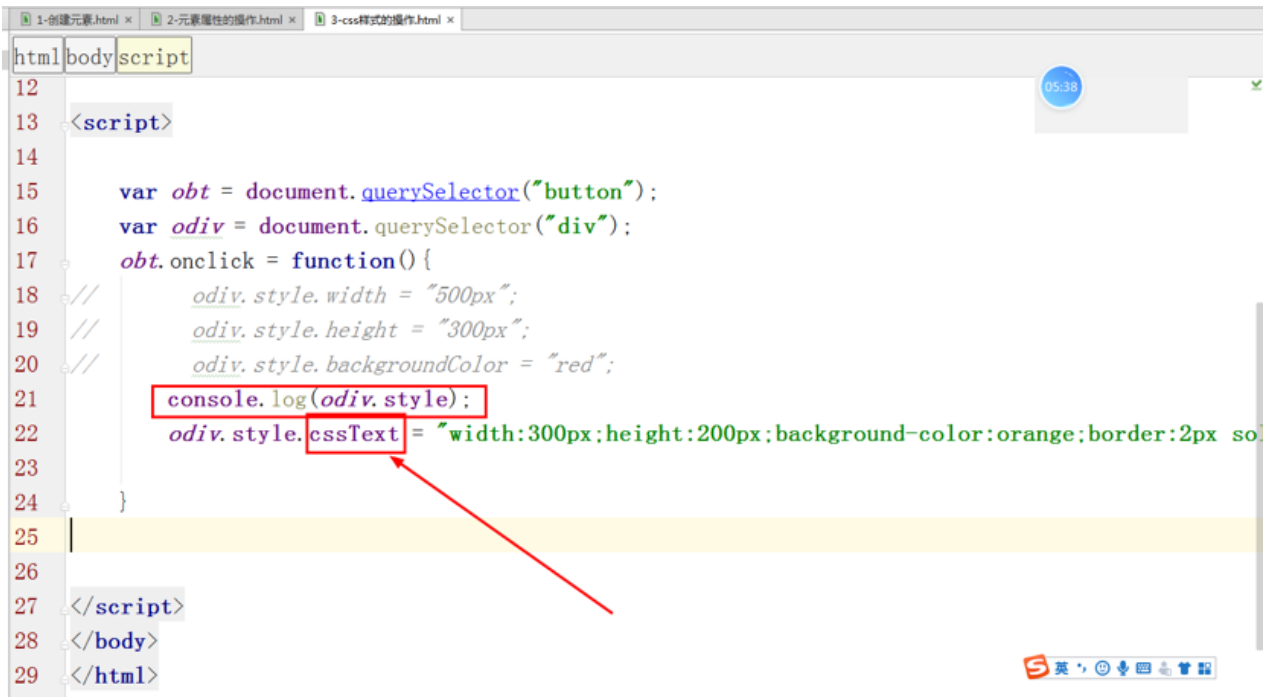
getAttribute('属性名')：获取属性的值

removeAttribute('属性名')：删除属性

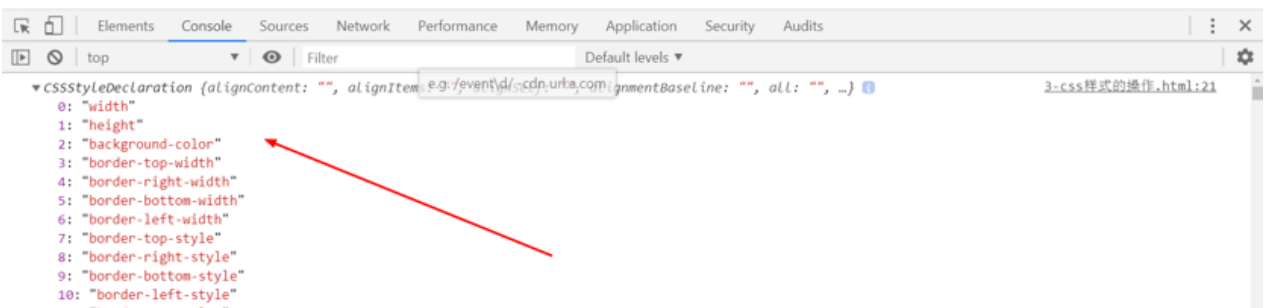
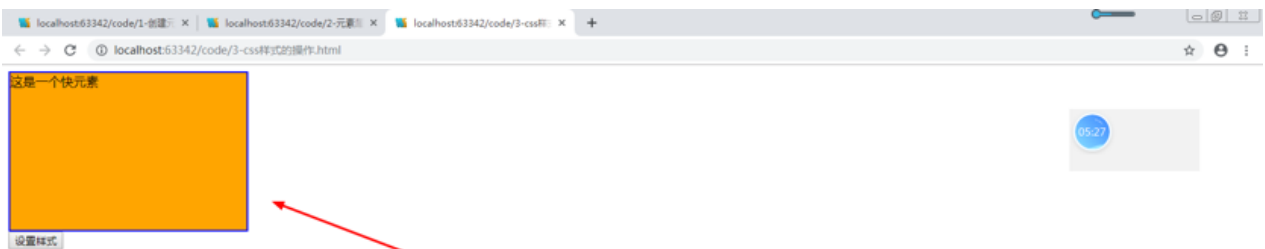


## 7.2. style属性的操作

style对象的cssText属性



```
12
13 <script>
14
15     var obt = document.querySelector("button");
16     var odiv = document.querySelector("div");
17     obt.onclick = function() {
18         //     odiv.style.width = "500px";
19         //     odiv.style.height = "300px";
20         //     odiv.style.backgroundColor = "red";
21         console.log(odiv.style);
22         odiv.style.cssText = "width:300px;height:200px;background-color:orange;border:2px solid orange";
23     }
24
25
26
27 </script>
28 </body>
29 </html>
```



## 7.3. style属性其他方法的操作

style对象提供了三个方法来读写行内css规则：

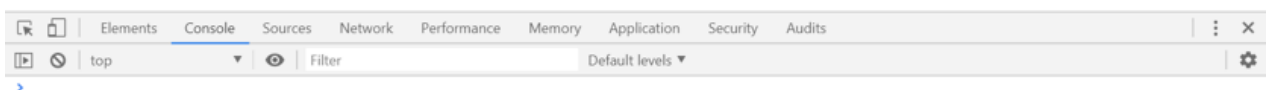
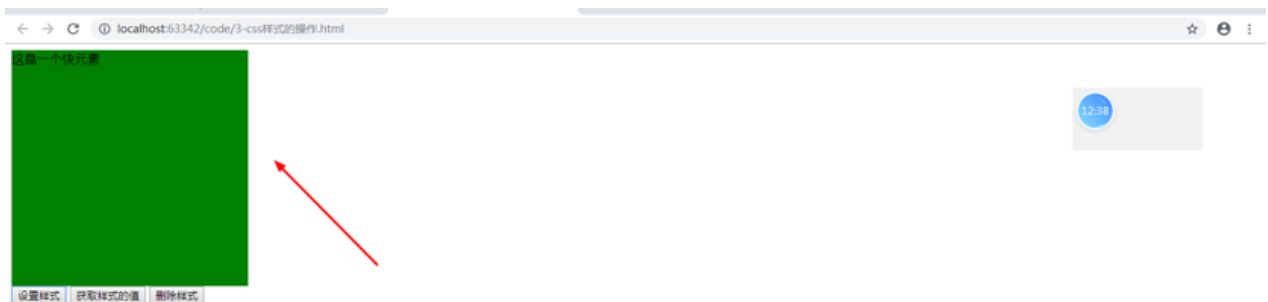
setProperty(propertyName,value)： 设置某个CSS属性。

getPropertyValue(propertyName)： 读取某个CSS属性的值。

removeProperty(propertyName)： 删除某个CSS属性。

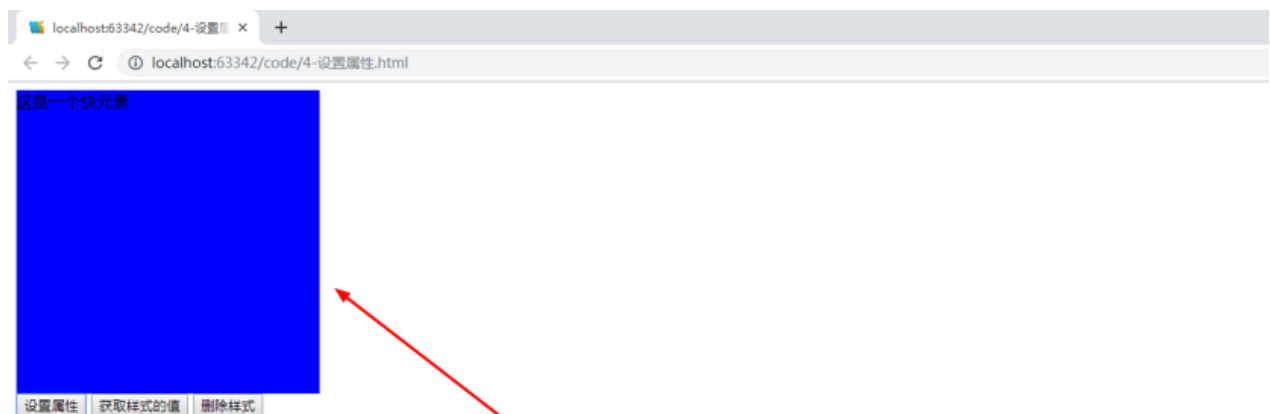
```
1-创建元素.html x 2-元素属性的操作.html x 3-css样式的操作.html x
html body script
8
9 <div>这是一个快元素</div>
10
11 <button>设置样式</button>
12 <button>获取样式的值</button>
13 <button>删除样式</button>
14
15 <script>
16
17     var obt = document.querySelectorAll("button");
18     var odiv = document.querySelector("div");
```

```
1-创建元素.html x 2-元素属性的操作.html x 3-css样式的操作.html x
html body script
19 //设置样式
20 obt[0].onclick = function() {
21     // odiv.style.width = "500px";
22     // odiv.style.height = "300px";
23     // odiv.style.backgroundColor = "red";
24     // console.log(odiv.style);
25     // odiv.style.cssText = "width:300px;height:200px;background-color:orange;border:2px solid orange";
26     odiv.style.setProperty("width", "300px");
27     odiv.style.setProperty("height", "300px");
28     odiv.style.setProperty("background", "green");
29 }
30 //获取样式
31 obt[1].onclick = function() {
32     console.log(odiv.style.getPropertyValue("width"));
33     console.log(odiv.style.getPropertyValue("background"));
34 }
35 //删除样式
36 obt[2].onclick = function() {
37     odiv.style.removeProperty("background");
38 }
```



## 7.4. setAttribute和setAttributeNode的区别

```
4-设置属性.html x 09:10
html body script
8
9 <div>这是一个快元素</div>
10
11 <button>设置属性</button>
12 <button>获取样式的值</button>
13 <button>删除样式</button>
14
15 <script>
16
17 var obt = document.querySelectorAll("button");
18 var odiv = document.querySelector("div");
19 //设置样式
20 obt[0].onclick = function(){
21 // odiv.setAttribute("style", "color:red;font-size:30px;font-weight:bolder;");//设置属性样式
22 var ostyle = document.createAttribute("style");//创建一个style属性
23 ostyle.value = "width:300px;height:300px;background:blue;";
24 odiv.setAttributeNode(ostyle);//给div元素添加一个style属性
25 }
```



7.5. 作业：使用DOM操作，在页面空白处单击，创建并显示一个宽度500，高度300，背景蓝色的div块元素。

## 7.6. 作业：使用DOM操作，创建一张图片，并添加属性，给图片添加边框样式

```
1-创建元素.html x
html body script
7 <body>
8 <script>
9
10 var oimg = document.createElement("img");//创建图片元素
11 // oimg.src = "";
12 oimg.setAttribute("src", "m1.jpeg");//给图片设置属性
13
14 var ostyle = document.createAttribute("style");//创建一个属性
15
16 ostyle.value = "border:10px dashed red";//设置属性值
17
18 oimg.setAttributeNode(ostyle);//给图片添加属性style
19
20 document.body.appendChild(oimg);
21
```

