

window对象（上）

0.1. 本节主要内容：



0.1. 学习目标：

节数	知识点	要求
第一节 BOM的概念	什么是BOM	了解
	BOM包含哪些内容	了解
第二节 window对象	关于window对象	了解
	window对象常用的属性	了解
	window对象常用的方法	了解
第三节 提示框	提示框	掌握
第四节 间隔调用和延迟调用	间隔调用和延迟调用	掌握

1. BOM的概念

1.1. 什么是BOM?

BOM: **Browser Object Model** 是浏览器对象模型, BOM由多个对象构成, 其中代表浏览器窗口的**window**对象是**BOM**的顶层对象也是核心对象, 其他对象都是该对象的子对象。

1.2. BOM包含哪些内容?

1、浏览器介绍

2、BOM对象包含

(1) **window** 对象, 是 JS 的最顶层对象, 其他的 BOM 对象都是 window 对象的属性。

(2) document 对象, 文档对象;

(3) location 对象, 浏览器当前URL信息;

(4) navigator 对象, 浏览器本身信息;

(5) screen 对象, 客户端屏幕信息;

(6) history 对象, 浏览器访问历史信息;

在浏览器中, window对象有双重角色, 它既是通过javascript访问浏览器窗口的一个接口, 又是ECMAScript规定的Global对象。

所有 JavaScript 全局对象、函数以及变量均自动成为 window 对象的成员。

全局变量是 **window** 对象的属性。

全局函数是 window 对象的方法。

```
10 <script>
11 // window.location.href = "http://www.huawei.com";
12 // window.document.write("hello!");
13 var age = 20;
14 function f1() {
15     console.log("我是函数f1");
16 }
17 window.f1();
18 console.log(window.age);
19 </script>
20
```

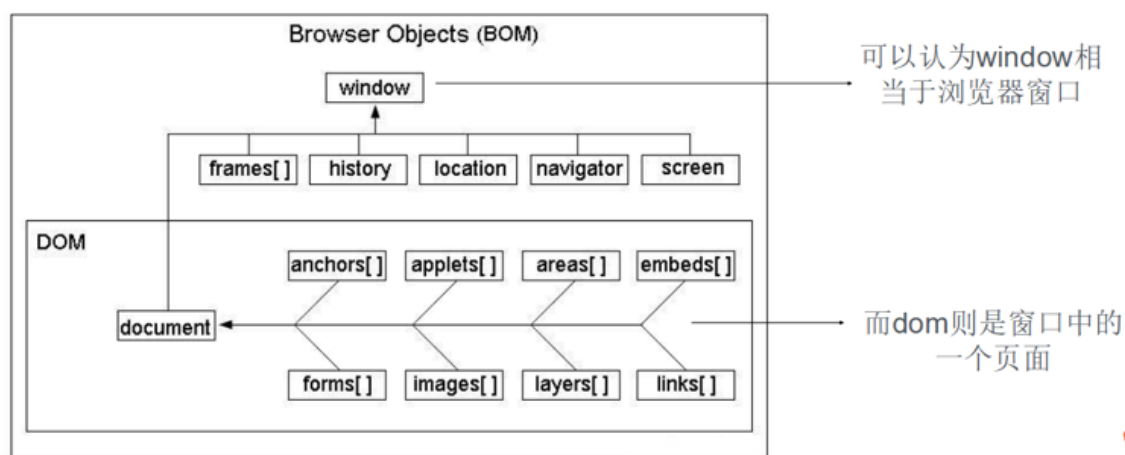
1.3. BOM和DOM的关系

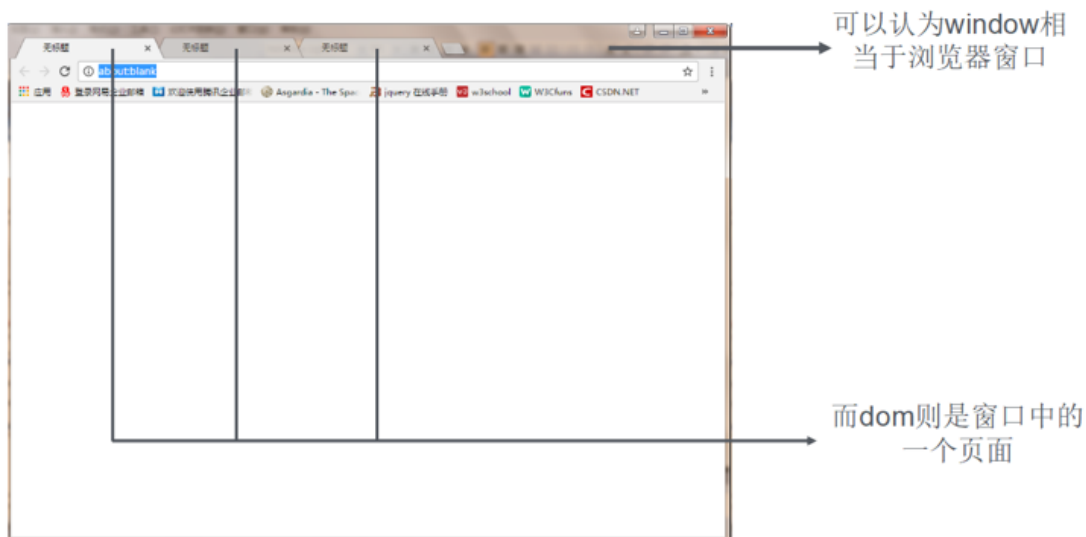
(1)DOM通过document对象来访问、控制、修改html和xhtml等文档中的内容

(2)BOM通过 window 对象来访问、控制、修改浏览器中的内容

联系：BOM包含DOM。 区别：DOM描述了处理网页内容的方法和接口，即操作页面内部

BOM描述了与浏览器进行交互的方法和接口，即操作页面之间





2. window对象

所有浏览器都支持 window 对象。它表示浏览器窗口。

所有 JavaScript 全局对象、函数以及变量均自动成为 window 对象的成员。

全局变量是 window 对象的属性。

全局函数是 window 对象的方法。

2.1. window对象

因为window对象是js中的顶级对象，因此所有定义在全局作用域中的变量、函数都会变成window对象的属性和方法，在调用的时候可以省略**window**。

例如：

打开窗口 `window.open(url);` 【等价于`open(url);`】

关闭窗口 `window.close();` 【等价于`close();`】

获取事件 window.event 【等价于event;】

获取文档 window.document 【等价于document】



```
10 <script>
11 // window.location.href = "http://www.huawei.com";
12 // window.document.write("hello!");
13 /* var age = 20;
14 function f1() {
15     console.log("我是函数f1");
16 }
17 window.f1();
18 console.log(window.age);*/
19 // window.open("http://www.baidu.com");
20 var obt = document.querySelector("button");
21 obt.onclick = function() {
22     window.close(); //关闭页面
23 }
24
25 </script>
26
```

2.2. window对象中常用的属性

1、window.name

属性：

window.name是window对象的一个属性，默认值为空。

特性：

window.name值在不同的页面（甚至不同域名）加载后依旧存在，并且可以支持非常长的name值（2MB左右）

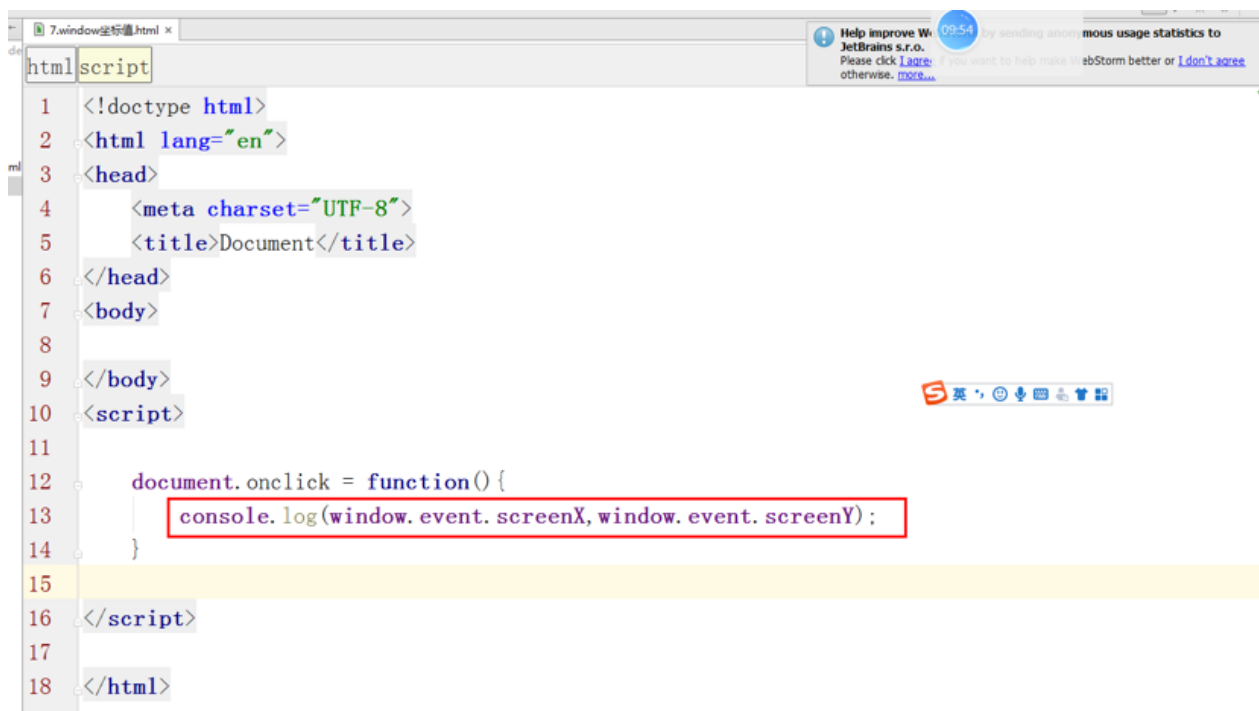
```
1 <!doctype html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Document</title>
6 </head>
7 <body>
8   <a href="6. 获取windowname属性的值.html">链接</a>
9 </body>
10 <script>
11   window.name = "window对象传值";
12   // console.log("name=", window.name);
13 </script>
14 </html>
```

```
1 <!doctype html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Document</title>
6 </head>
7 <body>
8 </body>
9 </body>
10 <script>
11   console.log(window.name);
12 </script>
13 </html>
```

2.3. 浏览器距离屏幕的距离

window.screenX

window.screenY



```
1 <!doctype html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Document</title>
6 </head>
7 <body>
8
9 </body>
10 <script>
11
12   document.onclick = function() {
13     console.log(window.event.screenX, window.event.screenY);
14   }
15
16 </script>
17
18 </html>
```

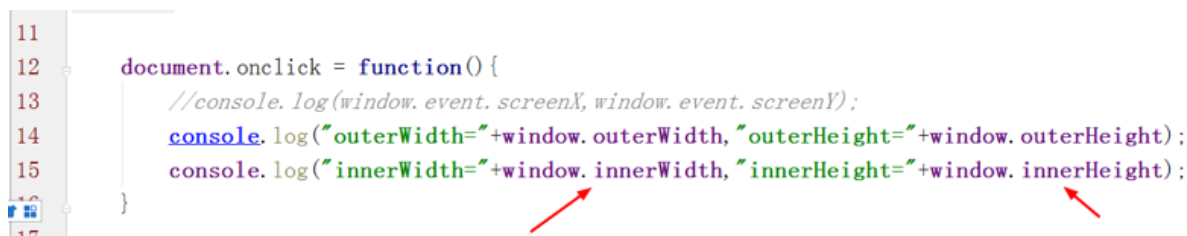
2.4. window尺寸属性

outerHeight属性设置或返回一个窗口的外部高度，包括所有界面元素（如工具栏/滚动条）。

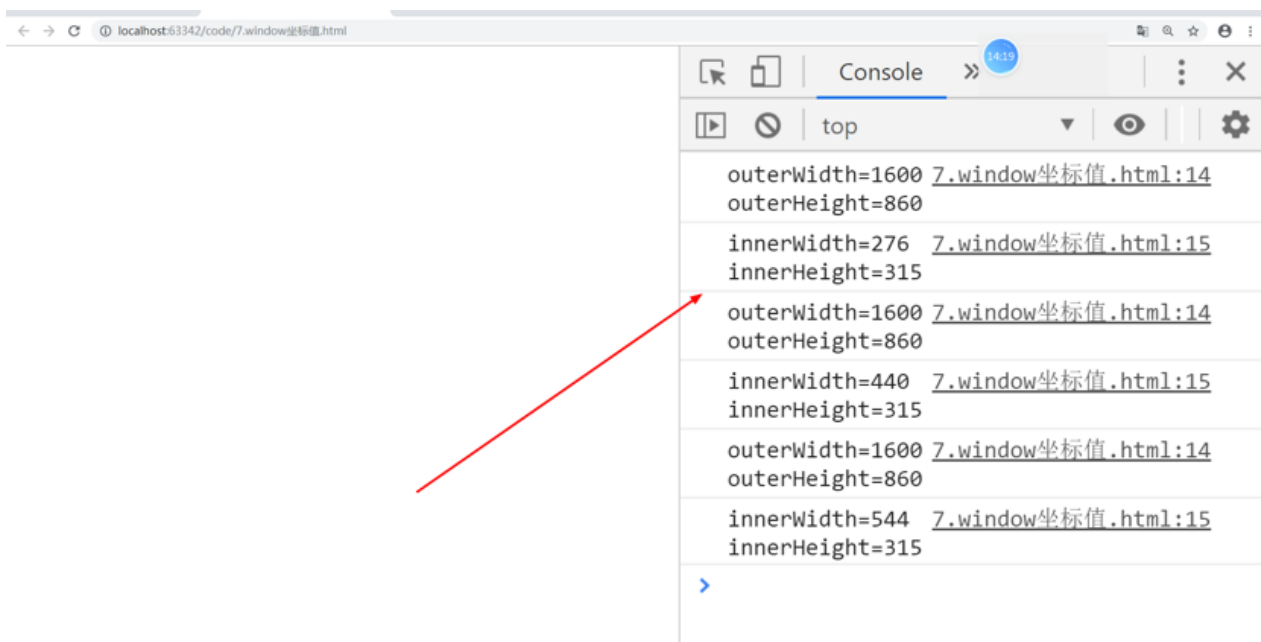
outerWidth属性设置或返回窗口的外部宽度，包括所有的界面元素（如工具栏/滚动条）。

innerheight 返回窗口的文档显示区的高度。

innerwidth 返回窗口的文档显示区的宽度。



```
11
12 document.onclick = function() {
13   //console.log(window.event.screenX, window.event.screenY);
14   console.log("outerWidth="+window.outerWidth, "outerHeight="+window.outerHeight);
15   console.log("innerWidth="+window.innerWidth, "innerHeight="+window.innerHeight);
16 }
17
```



2.5. document对象

2.6. window.navigator对象

window.navigator对象包含大量有关Web浏览器的信息，在检测浏览器及操作系统上非常有用。（这个对象和event一样是一个全局变量，并且是唯一的）

navigator.appCodeName //浏览器代码名的字符串表示

navigator.appName //官方浏览器名的字符串表示

navigator.appVersion //浏览器版本信息的字符串表示

navigator.userAgent //返回和浏览器内核相关的信息

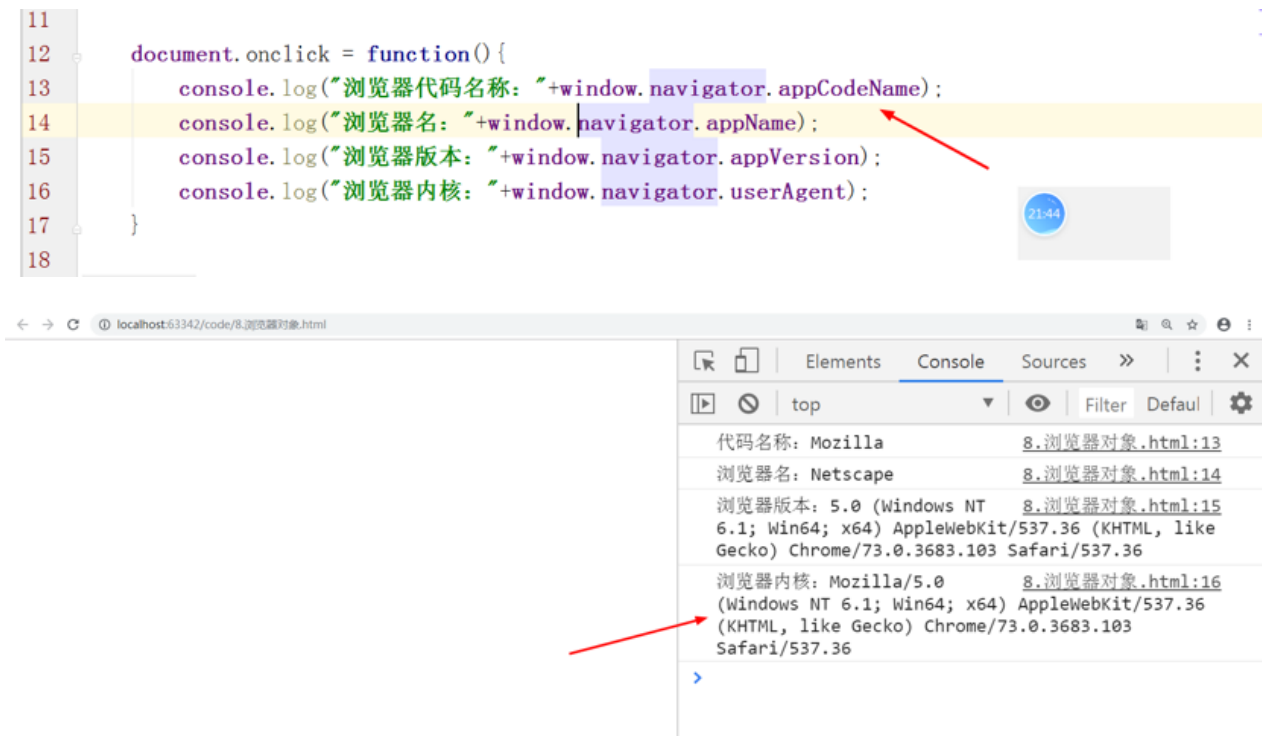
navigator.cookieEnabled //如果启用cookie返回true，否则返回false

navigator.javaEnabled() //如果启用java返回true，否则返回false

navigator.platform //浏览器所在计算机平台的字符串表示

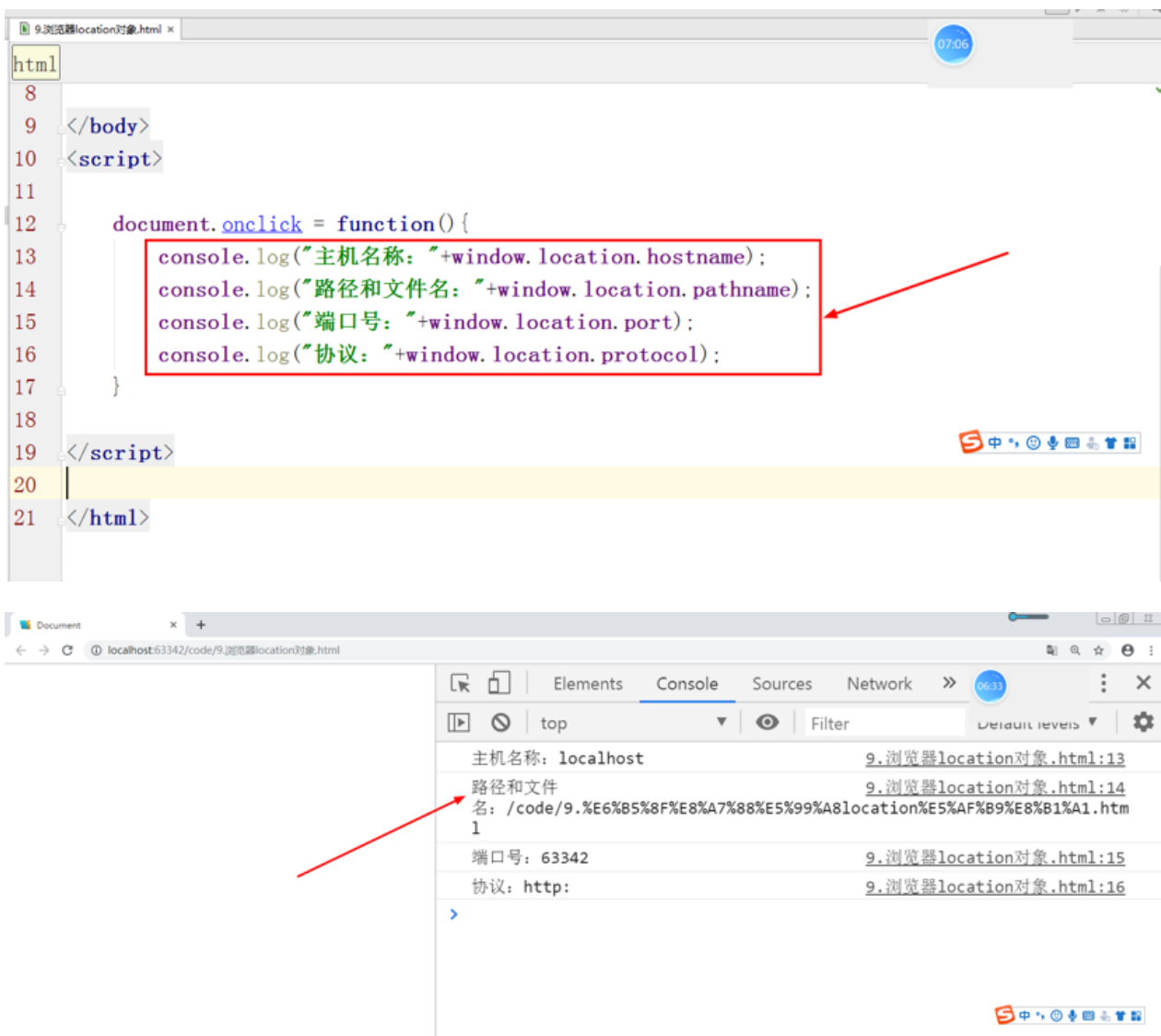
navigator.plugins //安装在浏览器中的插件数组

ps:如果window.navigator.userAgent出现了Mobile，可以确定用户使用的是移动设备。



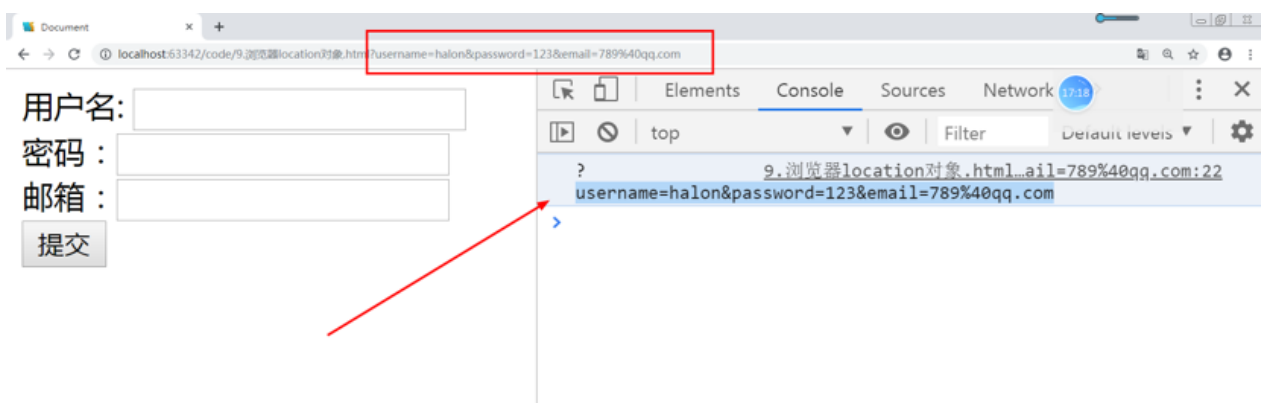
2.7. Location 对象

- location.hostname 返回 web 主机的域名
- location.pathname 返回当前页面的路径和文件名
- location.port 返回 web 主机的端口（80 或 443）
- location.protocol 返回所使用的 web 协议（http: 或 https:）



search 属性是一个可读可写的字符串，可设置或返回当前 URL 的查询部分（问号 ? 之后的部分）

```
html<script>
</body>
8 <form action="/" >
9   用户名: <input type="text" name="username"/><br>
10  密码: <input type="text" name="password"/><br>
11  邮箱: <input type="text" name="email"/><br>
12  <input type="submit" value="提交"/><br>
13 </form>
14 </body>
15 <script>
16
17 // document.onclick = function() {
18 /* console.log("主机名称: "+window.location.hostname);
19 console.log("路径和文件名: "+window.location.pathname);
20 console.log("端口号: "+window.location.port);
21 console.log("协议: "+window.location.protocol);*/
22 console.log(location.search);
23 //
24
```



2.8. screen对象

2.9. window对象中常用的方法

方法	描述
● alert()	显示带有一段消息和一个确认按钮的警告框。
● prompt()	显示可提示用户输入的对话框。
● confirm()	显示带有一段消息以及确认按钮和取消按钮的对话框。
● open()	打开一个新的浏览器窗口或查找一个已命名的窗口。
● close()	关闭浏览器窗口。
● print()	打印当前窗口的内容。
● focus()	把键盘焦点给予一个窗口。
● blur()	把键盘焦点从顶层窗口移开。
● moveBy()	可相对窗口的当前坐标把它移动指定的像素。

<code>moveTo()</code>	把窗口的左上角移动到一个指定的坐标。
<code>resizeBy()</code>	按照指定的像素调整窗口的大小。
● <code>resizeTo()</code>	把窗口的大小调整到指定的宽度和高度。
<code>scrollBy()</code>	按照指定的像素值来滚动内容。
● <code>scrollTo()</code>	把内容滚动到指定的坐标。
● <code>setInterval()</code>	每隔指定的时间执行代码。
● <code>setTimeout()</code>	在指定的延迟时间之后来执行代码。
● <code>clearInterval()</code>	取消 <code>setInterval()</code> 的设置。
● <code>clearTimeout()</code>	取消 <code>setTimeout()</code> 的设置。

●：扩展了解

●：必须掌握

2.10. 实例：三级联动菜单

```

1  <!doctype html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Document</title>
6  </head>
7  <body>
8
9  <p align="center">
10     <span>省: </span><select name="" id="" class="shengSelect">
11         <option value="">请选择</option>
12     </select>
13     <span>市: </span><select name="" id="" class="shiSelect"></select>
14     <span>县: </span><select name="" id="" class="xianSelect"></select>
15 </p>
16
17 </body>

```

```

html script
17 </body>
18 <script>
19
20 window.onload = function() {
21     //获取相应的元素
22     var shengSelect = document.querySelector(".shengSelect");
23     var shiSelect = document.querySelector(".shiSelect");
24     var xianSelect = document.querySelector(".xianSelect");
25
26     var shengArr = ["北京", "东京", "纽约"]; //省的数据
27
28     var shiArr = [
29         ["朝阳区", "海淀区", "大兴区"],
30         ["新宿区", "文京区", "中央区"],
31         ["曼哈顿", "皇后区", "布朗克斯区"]
32     ];

```

```

html script
33 var xianArr = [
34     ["朝阳一县", "朝阳二县", "朝阳三县"], ["海淀一县", "海淀二县", "海淀三县"], ["大兴一县", "
35     ["新宿一县", "新宿二县", "新宿三县"], ["文京一县", "文京二县", "文京三县"], ["中央一县", "
36     ["曼哈顿一县", "曼哈顿二县", "曼哈顿三县"], ["皇后一县", "皇后二县", "皇后三县"], ["布朗克
37 ];
38
39 //给省添加数据
40 for(var i = 0; i < shengArr.length; i++) {
41     var opt = new Option(shengArr[i]); //创建option选项
42     console.log(opt);
43     shengSelect.options.add(opt); //给select添加option选项
44     console.log(shengSelect.options);
45 }
46 //省的onchange事件

```

```

html script
46 //省的onchange事件
47 var shengIndex = 0;
48 shengSelect.onchange = function() {
49     //获取省的索引值
50     console.log(this.selectedIndex);
51     console.log(shengSelect.selectedIndex);
52     console.log(event.target.selectedIndex); /*
53     //this:shengSelect, 获取下拉列表框中具体的值
54     console.log(this.options[this.selectedIndex].value);
55     shengIndex = this.selectedIndex - 1; //获取省的索引值
56
57     if(shengIndex == -1) {
58         shiSelect.options.length = 0; //清空市里面的数据
59     }

```

```
html script
60 else{
61     //添加市
62     shiSelect.options.length = 0; //清空市里面的数据
63     for(var j = 0; j < shiArr[shengIndex].length; j++) {
64         var opt = new Option(shiArr[shengIndex][j]); //创建市的option选项
65         shiSelect.options.add(opt); //给市添加数据
66     }
67     //添加县, 每个省的第一个市的第一个县的数据
68     xianSelect.options.length = 0;
69
70     for(var k = 0; k < xianArr[shengIndex][0].length; k++) {
71         var opt = new Option(xianArr[shengIndex][0][k]); //添加县的数据
72         xianSelect.options.add(opt);
73     }
74 }
75
76
77 //市的onchange
78 shiSelect.onchange = function() {
79     var shiIndex = this.selectedIndex; //获取市的索引值
80     xianSelect.options.length = 0;
81     for(var i = 0; i < xianArr[shengIndex][shiIndex].length; i++) {
82         var opt = new Option(xianArr[shengIndex][shiIndex][i]);
83         xianSelect.options.add(opt);
84     }
85 }
86
87
88 </script>
89
```

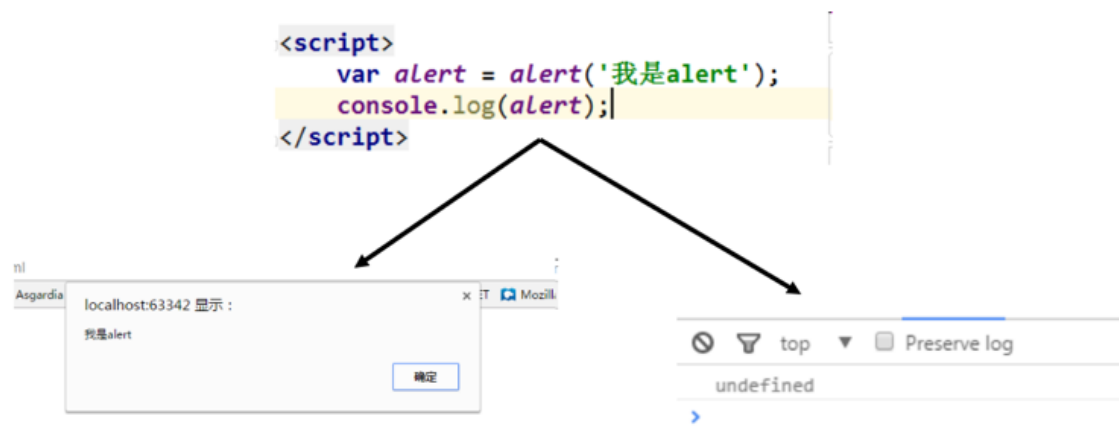
省： 纽约 ▼ 市： 布朗克斯区 ▼ 县： 布朗克斯一县 ▼

3. 提示框

3.1. 警告框

(1) `alert(alertMsg);`

表示警示框，作用是提示用户信息，该方法执行后无返回值。



3.2. 输入框

`prompt(alertMsg,defaultMsg);`

表示警示框，作用是提示用户信息，该方法执行后根据情况不同返回值略有不同。

a) 点击取消, 返回值为 `null`

b) 没有默认值

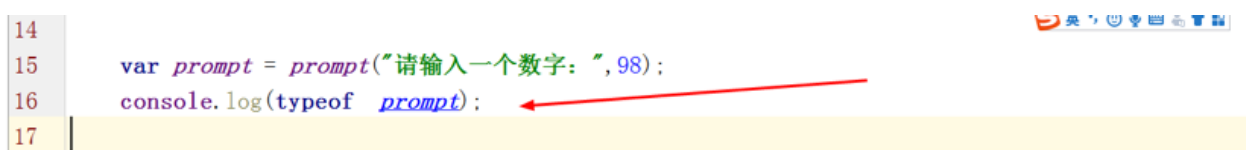
如果用户没有输入内容，返回一个空字符串

如果用户输入了内容，返回值为用户输入的内容

c) 有默认值

如果用户没有输入内容，返回默认值

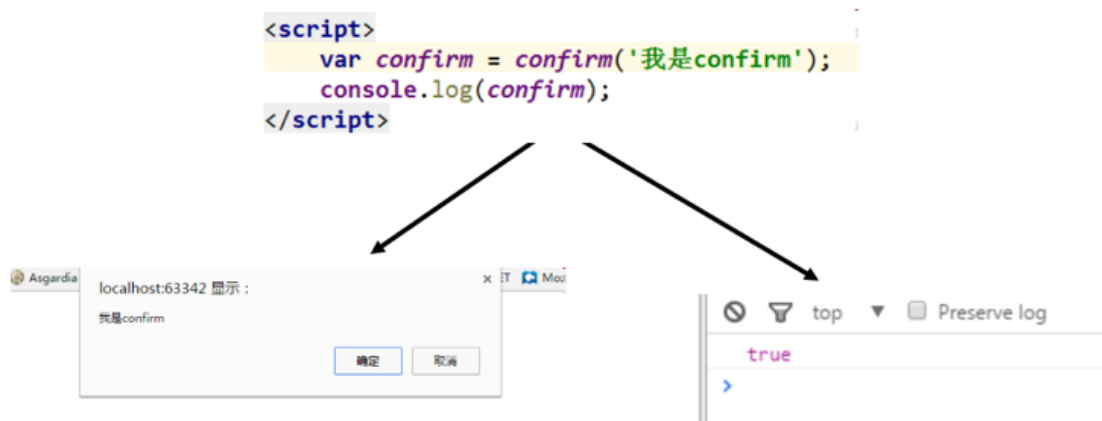
如果用户修改了默认，返回值为用户输入的内容



3.3. 确认框

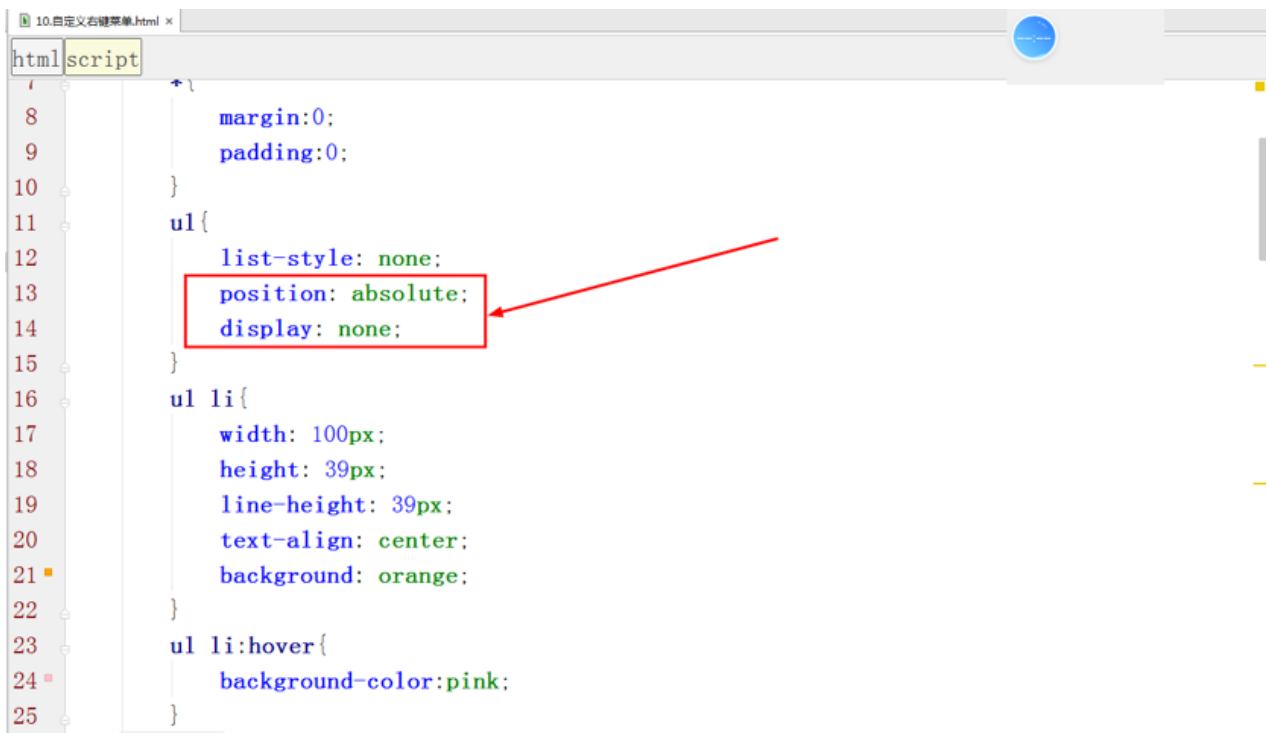
(3) confirm(alertMsg)

表示警示框，作用是提示用户信息，点击确认返回true，点击取消返回false。



3.4. 实例：自定义右键菜单

```
28 <body>
29   <ul>
30     <li>赶快说话</li>
31     <li>离开页面</li>
32     <li>选中搜索</li>
33     <li>输入搜索</li>
34   </ul>
35 </body>
```




```
36 <script>
37     var oul = document.querySelector("ul");
38     var oli = document.querySelectorAll("li");
39     //禁用右键菜单
40     document.oncontextmenu = function() {
41         return false;
42     }
43     //事件委托: 利用事件的冒泡元素, li委托ul来做这件事。
```

```
43     //事件委托: 利用事件的冒泡元素, li委托ul来做这件事。
44     oul.onmousedown = function() {
45         switch(event.target.innerHTML) {
46             case "赶快说话":
47                 alert("ok!");
48                 break;
49             case "离开页面":
50                 window.close();
51                 break;
52             case "选中搜索":
53                 window.open("http://www.baidu.com");
54                 break;
55             case "输入搜索":
56                 var re = prompt("请输入内容: ", "中国制造2025!");
57                 window.open("https://www.baidu.com/s?wd="+re);
58                 break;
59         }
60     }
```

```
10.自定义右键菜单.html x
htmlscript
79     /*
80     document.onmousedown = function(e) {
81         if(e.button == 2) {
82             oul.style.display = "block";
83             oul.style.left = e.clientX + "px";
84             oul.style.top = e.clientY + "px";
85         }
86         else {
87             oul.style.display = "none";
88         }
89     }
90 }
91 </script>
92
93 </html>
```

赶快说话

离开页面

选中搜索

输入搜索

3.5. 实例：放大镜效果

```
46 <div class="box">
47   <div class="fdj"></div>
48 </div>
49 <div class="show">
50   
51 </div>
```

```
2-作业放大效果.html ×
html body script
7   *{
8     margin:0;
9     padding:0;
10  }
11  body{
12    height: 2000px;
13  }
14
15  /*背景图片的原图400*4000*/
16  .box{
17    width: 400px;
18    height: 400px;
19    background: url("img.png") no-repeat;
20    position: relative;
21    float: left;
22  }
```

```
2-作业放大镜.html x
html body script
23      /*放大镜*/
24      .fdj{
25          width: 100px;
26          height: 100px;
27          background: rgba(200, 100, 100, 0.5);
28          display: none;
29          position: absolute;
30      }
31      /*包含放大之后大图的盒子*/
32      .show{
33          width: 400px;
34          height: 400px;
35          overflow: hidden;
36          position: relative;
37          float: left;
38          display: none;
39      }
40      img{
41          position: absolute;
```

```
2-作业放大镜.html x
html body script
52      <script>
53          var box = document.querySelector(".box");//获取原图元素
54          var fdj = document.querySelector(".fdj");//获取放大镜元素
55          var show = document.querySelector(".show");//包含大图的盒子元素
56          var img = document.querySelector("img");//获取大图元素
57
58          //鼠标经过放大镜和大图盒子显示
59          box.onmouseover = function() {
60              fdj.style.display = "block";
61              show.style.display = "block";
62          }
```

```
2-作业放大镜.html x
html body script
63      //鼠标移动获取放大镜和大图的距离
64      box.onmousemove = function(e) {
65          var x = e.clientX - 50;//距离左侧的距离
66          var y = e.clientY - 50;//距离顶部的距离
67          if(x<0){
68              x = 0;
69          }
70          if(y<0){
71              y = 0;
72          }
73          if(x>box.offsetWidth - fdj.offsetWidth){
74              x = box.offsetWidth - fdj.offsetWidth;
75          }
76          if(y>box.offsetHeight - fdj.offsetHeight){
77              y = box.offsetHeight - fdj.offsetHeight;
78          }
79          fdj.style.left = x + "px";
80          fdj.style.top = y + "px";
81          var rate = 1600 / 300 ;//比率
```

```

82     img.style.left = -x * rate + "px";
83     img.style.top = -y * rate + "px";
84 }
85 //鼠标离开放大镜和大图盒子隐藏
86 box.onmouseout = function() {
87     fdj.style.display = "none";
88     show.style.display = "none";
89 }

```



4. 间隔调用和延迟调用

4.1. 定时器

setInterval(表达式, 毫秒数)

语法: `var timer = null;`

`timer = setInterval(需要执行的函数, 执行间隔时间ms);`

例如:

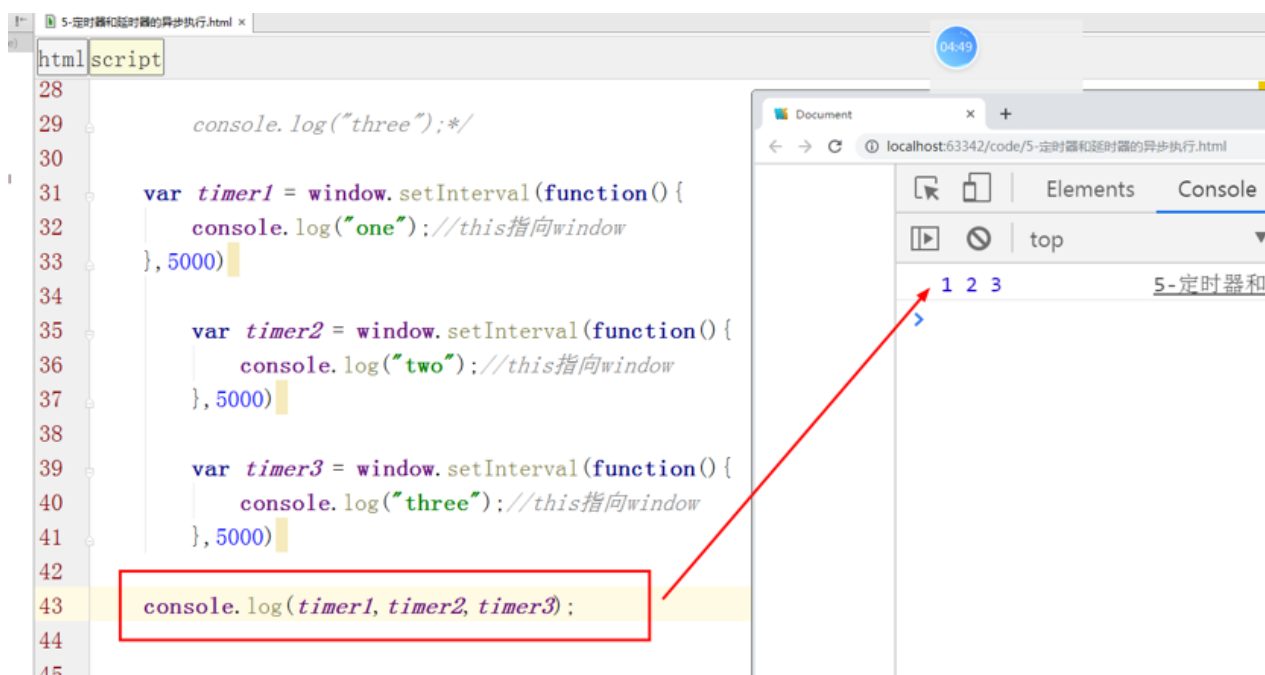
```

var timer = null;
timer = setInterval(function(){
    console.log('hello world!');
}, 2000);

```

总结说明:

- 1、要把定时器下边的任务执行完毕后 才会去执行定时器的内容
- 2、定时执行中**this**指向**window**
- 3、每创建一个定时器 就会有一个唯一的**id**被返回 id从开始 之后累加



- 4、清除定时器时 不仅可以使使用变量 也可以使用唯一Id清除



5、当定时执行的函数是包含参数时 则应该 将函数和参数 使用引号包裹起来

```
53     var timer = setInterval('gd(95)', 1000);
54
55     function gd(a) {
56         console.log(a);
57     }
58
59
```

```
54     var timer = setInterval(function() {
55         gd(123)
56     }, 1000);
57     function gd(a) {
58         console.log(a);
59     }
60
```

```
54     obt.onclick = function() {
55         var timer = setInterval(function() {
56             gd("aa");
57         }, 1000);
58         function gd(a) {
59             console.log(a);
60         }
61     }
62
```

首先明确两点：

1.JS 执行机制是单线程。

2.JS的Event loop是JS的执行机制

按照这种的分类方式JS的执行机制是：

异步的编程思想：

首先，判断JS是同步还是异步，同步进入主线程，异步进入Event table

其次，异步任务在Event table中注册函数，当满足特定的条件，被推入Event queue（消息队列）最后，同步任务进入主线程后一直执行，直到主线程空闲后，才会去Event queue中查看是否有可执行的异步任务，如果有就推入主线程中执行。

清除间隔调用

既然间隔调用每隔一段时间就会自动执行一次，那么清除间隔调用就势必存在。

语法：clearInterval(变量标识)

例如：clearInterval(timer);

上述代码就能够将刚刚创建的定时器移除掉，令其不在间隔一段时间后自动再次执行。

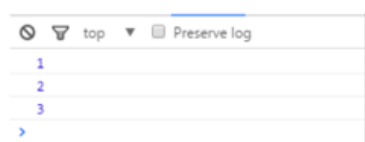
注意：

(1)间隔调用的返回值是一个数字队列，因此通过访问数字队列来清除间隔调用也被允许。

```
<script>
  var timer1 = setInterval(function () {console.log('hello world!');},2000);
  var timer2 = setInterval(function () {console.log('hello world!');},2000);
  var timer3 = setInterval(function () {console.log('hello world!');},2000);

  console.log(timer1);
  console.log(timer2);
  console.log(timer3);
</script>
```

执行结果为：



而如果想要清除某个间隔调用，可以直接通过下方的代码来清除

```
clearInterval(1);
clearInterval(2);
clearInterval(3);
```

注意：

(2) 如果间隔调用的函数需要传入参数，则间隔调用需要使用如下的方式声明

语法：var timer = null;

timer = setInterval(字符串,执行间隔事件ms);

例如：var timer = null;

```
function show(words){console.log(words);}
```

```
timer = setInterval('show("hello world!")',2000);
```

(3) 间隔调用不是立即执行，而是在【任务队列中的任务完成后】才执行间隔调用

(4) 因为间隔调用函数的实际执行者是window，因此间隔调用内部的this指向window

4.2. 延时器

延迟调用又叫延迟调用函数。是一种能够等待一定时间后在执行的函数。

语法：var timer = null;

timer = setTimeout(需要执行的函数，等待的时间);

例如：var wait = null;

```
wait = setTimeout(function(){
```

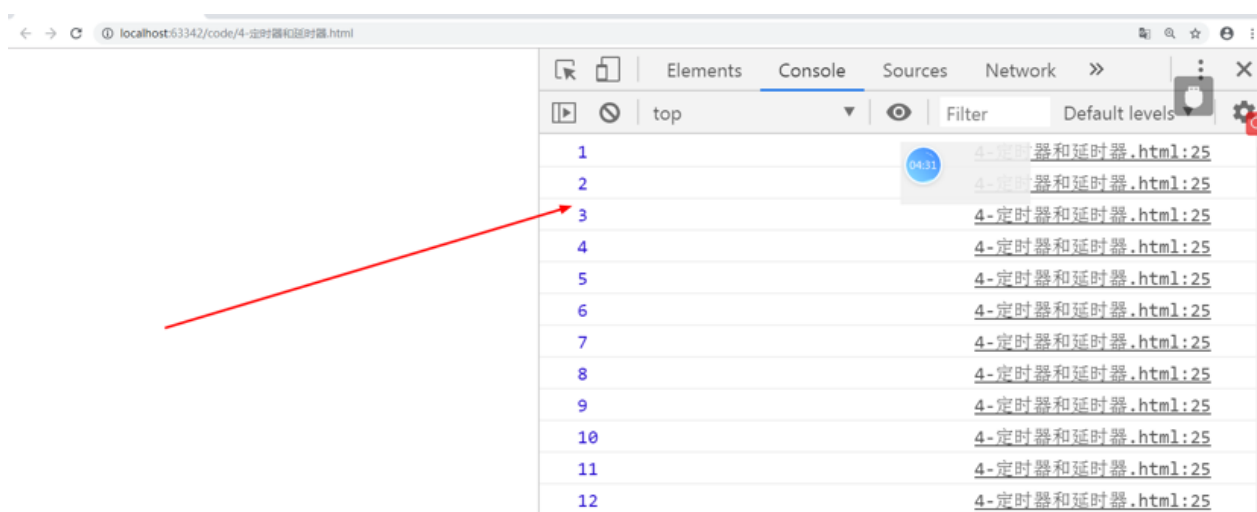
```
console.log('hello world!');
```

```
},2000);
```


根据语法所述，上述代码所表示的含义为：等待2s后打印一句【hello world! 】

注意：延迟调用除了在语法上和间隔调用略有不同外，其余语法均相同。

```
22     var n = 0;
23     function f1(){
24         n++;
25         console.log(n);
26         setTimeout("f1()",1000);//实现定时器的功能
27     }
28     f1();
29
30 </script>
```



(1)阅读下列代码，口算打印结果

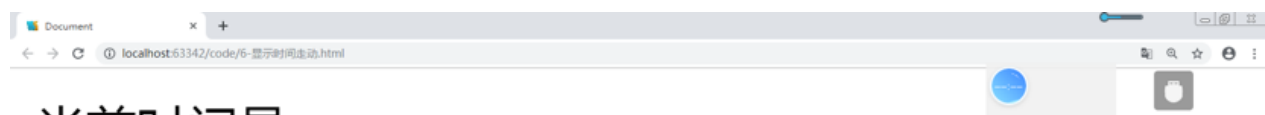
setInterval(function () {console.log(1111);},0);

setTimeout(function () {console.log(2222);},0); (2)阅读下列代码，口算打印结果

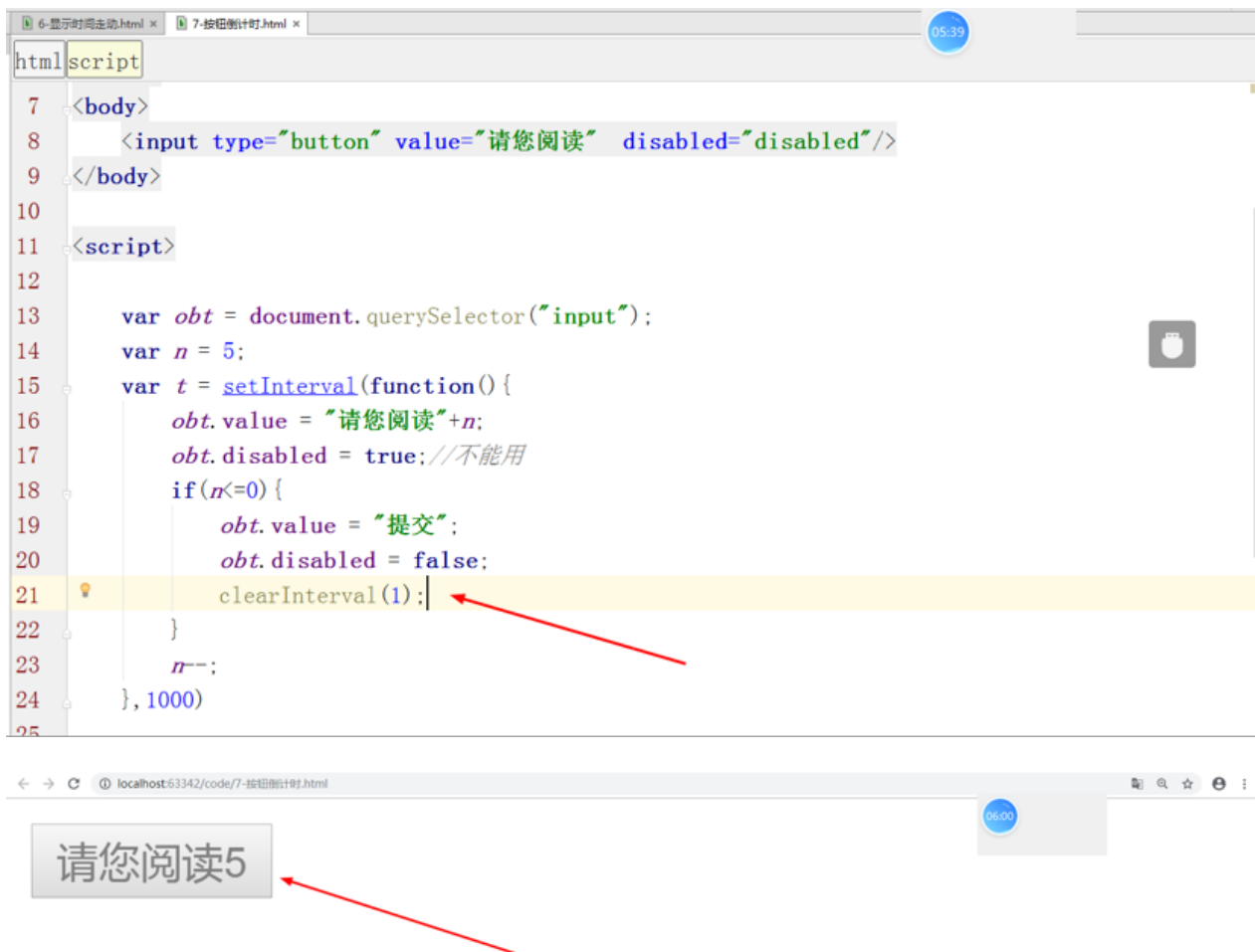
```
var div = document.getElementsByTagName("div").item(0);
div.onclick = function () { setTimeout(function ()
{console.log(this);}, 1000); };
```

4.3. 实例：时间走动

```
html script
10 当前时间是: <div class="mytime"></div>
11 <script>
12
13 function timego() {
14     var mytime = document.querySelector(".mytime");
15     var date = new Date();
16     var hour = date.getHours();
17     var minute = date.getMinutes();
18     var second = date.getSeconds();
19
20     hour = hour > 12 ? "PM" : "AM";
21     minute = minute < 10 ? "0" + minute : minute;
22     second = second < 10 ? "0" + second : second;
23
24     mytime.innerHTML = hour + ":" + minute + ":" + second;
25     // setTimeout("timego()", 1000);
26     // mytime.innerHTML = date.toLocaleTimeString();
27 }
28
29
30 </script>
```



4.4. 实例：倒计时



4.5. 实例：进度条的加载



```
6-显示时间走动.html x 7-按钮倒计时.html x 8-进度条.html x
html body script
23
24 function go() {
25     var sidebar = document.querySelector(".sidebar");
26     sidebar.style.width = parseInt(sidebar.style.width) + 1 + "%";
27     sidebar.innerHTML = sidebar.style.width;
28     if(sidebar.style.width == "100%"){
29         clearInterval(t);
30     }
31 }
32
33 var t = setInterval("go()", 30);
34
35 </script>
36 </body>
37
38
```

