JavaScript基础

可以对照着这个看[js案例]

js在html中的使用

在网页中实现js有两种办法，

- 一种是写在网页内部，写在<script> </script>这俩之间，这个可以随便放放哪都行（head,body都行），而且还可以放好多个js块，同一个Html中多个js块之间参数和方法共通
- 另一种是从外部文件导入<script type="text/javascript" src="test.js"></script>
- 当下的很多库或是框架就是基于js的，我们可以将它们保存下载下来放到项目文件夹中，当js文件来引入，如JQuery库：

```
<script src="jquery-3.6.0.min.js" type="text/javascript" charset="utf-8">
</script>
```

JS的基本内容

1. 执行顺序：

   js程序按照在HTML文件中出现的顺序逐行执行，如果需要在整个HTML文档中执行（如函数，全局变量等）最好将其放在文件的head中

2. JS大小写敏感
3. 每行结尾的分号可有可无
4. 常量使用const定义，变量通过var定义
5. 如果对一个变量重复定义新的定义会覆盖旧的定义
6. 赋值时null表示一个变量被赋予了一个空值，而undefined则表示这个变量尚未被赋值

js中的运算符

1. == 与 ===的区别：

   == 只对数据表面值是否相等进行，而不判断值得数据类型是否相等，而 ===不仅对值进行判断而且对数据类型也进行判断

   !=与!==也是同理，!=仅仅判断值是否相等而! ==为不绝对等于，值和数据类型都会进行判断

2. 条件运算符：操作数？结果1：结果2 是js所支持的一种特殊的三目运算符，操作数为真则输出结果1，操作数为假则输出结果2，如之前所用的

```
video2[video2.paused? 'play':'pause']() //如果视频在暂停状态则播放，如果在播放则暂停
                                        //为真就变成了video2.play()了
```

3. typeof运算符：

   用于返回它的操作数当前的数据类型

4. new运算符：通过new来创建一个新的对象，具体使用方法如下：

```
Object1=new Object //创建一个新的对象
arr=new Array()//创建一个新的数组对象
data3=new Data("August 8 2008")//创建一个新的日期对象并给其赋值JS
```

JS中的函数

函数的定义语句通常被放在head中或是外接的js文件中，而函数的调用语句通常会放在body中，如果在函数定义之前调用函数，执行将会出错。

函数的调用方法：

1. 直接在js块内调用，直接写
2. 通过事件触发函数，如单击事件
3. **通过链接来调用函数**：可以在超链接< a>中调用函数，在< a>的标签中的href值为"javascript:关键字"来调用函数，当用户单击该连接时，相关函数将被执行，如下所示：

```
<script>
    function test(){
        alert("你点击了这个链接")
    }
</script>
<a href="javascript:test()">单击这个链接触发test函数</a>
```

我们可以在函数中再去定义函数，叫做**嵌套函数**，这样可以使内部函数可以轻松的获得外部函数的参数以及函数的全局变量

JS中的全局函数

- eval():如果参数是表达式，则 eval() 计算表达式。如果参数是一个或多个 JavaScript 语句，则 eval() 执行这些语句
- isNaN():
- parseInt():将字符型转换为整形数字，可以转换为不同的进制，如果字符串不易数字开头则返回NaN
- parseFloat():将字符串转换为浮点型
- encodeURI():将字符串转换为有效的URL链接
- decodeURI():对上面的URL进行解码成字符串，这两个互为逆向操作。

JS对象编程

最主要的两个对象为Window窗口对象和Document文档对象

- window对象代表的是打开的浏览器窗口，通过window对象可以控制窗口的大小和位置。
- document对象代表浏览器窗口中的文档，是window对象的子对象

window对象的使用

(如果只有当前一个窗口使用window对象的方法时可以把window省略)

window对象可以直接调用其方法和属性，语法为

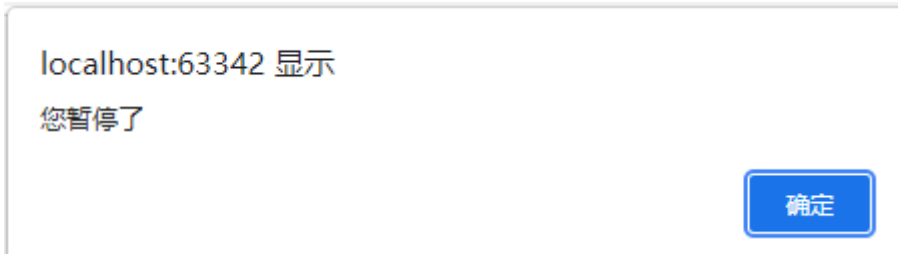
```
window,属性名
window.方法名(参数列表)

window.alert("您暂停了")  //弹出警示框（只有一个确认按键）  可以直接alert()
window.document.write("啊啊啊")//将创建一个新的页面内容只有啊啊啊

window.prompt("请输入您的地址","格式为XX省XX市") //弹出输入框，返回值为你的输入内容，如果没输入或是点取消则返回null
+

window.confirm("请确认") //带取消的确认框,有返回值，点击确认则返回Ture

//也可以这么用
<button onclick='window.alert("您按下了这个按钮")'>点一下试试？</button>
```



点一下试试？

注意：

警告对话框是由当前正在运行的页面弹出的，在对该对话框进行处理前不能对当前页面进行操作，并且其后面的代码也不会被执行，只有将警告对话框进行处理后（如单击“确定”按钮或关闭对话框）才能继续对当前页面进行操作，后面的代码才能继续执行，

```
function zuhe(){
if(window.confirm("你确定要买么？")){
 var cont=window.prompt("请输入地址","格式为XX省，XX市");
 if(cont!=null){
   window.alert("购买成功,订单马上发往"+cont);
 }else{
   alert("那真是太可惜了")
 }

}else{
 alert("那真是太可惜了")
}
```

就是先确认再输入再警告，除了alert外，其他的两个函数都有返回值，null就代表为空

还有以下常用的window对象的方法：（windows 下的子类都能用，如表单中输入框可以直接`form1.user.focus()`）

- open():打开新的网址或是本地文件

  ```
  myWindow=window.open('url','target','width=200,height=100');//创建了一个名为mywindow的窗口，额外的参数如长和宽需要用单引号包起来
  ```

- close():关闭被引用的对话框

- focus():使指定窗口获得焦点`mywindow.focus()`焦点就在mywindow这个窗口上了

  常用在表单上，可以指定表单首先需要输入什么

- blur():使窗口不获得焦点（如键盘事件等都还是在原来的网页触发，想要在新窗口触发就得点一下这个新窗口）

- scrollTo(x,y):当指定页面有滚动条时使用这个调整滚动条使界面滚动到指定坐标，x,y就是两个数不带单位代表向下或是向左右拖动多少像素的滚动条。
- scrollBy():和上面的相比这是相对位置，相对当前滚动多少
- setTimeout(timer):在指定的毫秒数后调用函数,只执行一次

  ```
  setTimeout(function(){ alert("Hello"); }, 3000);
  ```

- setInterval(timer):每过指定的时间，都会触发函数

  ```
  setInterval(function(){ alert("Hello"); }, 3000);
  ```

- moveTo(x,y):将目标窗口移动到指定位置
- moveBy(x,y):将目标窗口相对移动
- resizeTo(x,y):改变目标窗口的大小。（不能设置那些不是通过 window.open 创建的窗口）
- resizeBy(x,y):改变当前窗口的大小，可以是负值如`MyWindow.resizeBy(-100,100)`就是使宽减100高加100变成了100*300了

改变窗口的基本上都对主窗口没有用，只对使用open打开的窗口有用

screen对象：

screen对象时js中的屏幕对象，用来返回屏幕的各种参数，有以下属性width,height为当前屏幕的宽和高，pixelDepth为当前屏幕的每个像素的位数，colorDepth为当前显示器的色深，availWidth和availHeight为当前显示器的可用宽度和高度（除去任务栏之后的高度）

访问窗口历史

利用history对象可以实现访问窗口历史，history对象是一个只读的URL字符串数组，主要用来储存一个最近所访问的网页的URL地址的列表，常用的属性以及方法如下：

- length:历史列表的长度
- current:当前文档的URL
- next：历史列表的下一个URL
- previous：历史列表的前一个url

方法：

- back():退回到前一页
- forward():重新进图下一页
- go():进入指定的网页,参数为数字，正数为向下移动，负数为向前移动

可以利用超链接标签a 和js来实现在页面中跳转

```
<a href="javascript:window.history.forward()">forward</a>
<a href="javascript:window.history.go(1)">forward</a>

<a href="javascript:window.history.go(window.history.length-1)">末尾</a>//可以这样跳转到最新的网页
```

Document文档对象

文档对象（Document）代表浏览器窗口中的文档，该对象是window的子对象（可以直接window.document来访问），document对象可以访问HTML中的包含任何HTML标记中的内容，如表单，图像，表格和超链接等。

其常用属性有：（括号中的内容可以省略）

- alinkColor属性：(参数名=)document.alinkcolor(=设定的颜色)用来获得或改变当鼠标放在链接上的颜色
- linkColor属性：(参数名=)document.linkColor(=设定的颜色):用来改变或获取页面中未被单击的链接的颜色
- vlinkColor属性：(参数名=)document.vlinkColor(=设定的颜色):用来获取或改变被点击过的链接的颜色

以上三个属性都已经废弃了，有些浏览器可能不支持，可以直接使用css进行编辑

可以使用document对象来获取当前页面的信息

```
document.title  :获取页面标题
document.domain  ：获取页面域名
document.URL：获取页面URL
```

document对象的方法：

- document.open（）：打开一个文档，并接收write和writeln方法创建的页面内容
- close():
- write():向文档中写入HTML或是js语句
- writeln:也是写入，但是以换行符结束（在最后自带一个换行符）
- createElement（）:创建一个HTML标记
- getElementById（）:获取指定ID 的HTML标记

document对象的常用事件

事件可以直接使用不用加括号如`a.onblur=function(){}`,方法才用加括号如`a.blur()`是清除a的焦点

- onblur:元素或窗口本身失去焦点时触发

  焦点指的是文本输入时的那个一闪一闪的输入杠，如定义一个文本输入框，点击它准备输入的时候它就获得了焦点，点其他的地方使这个输入杠消失了就是失去焦点事件了。

- onchange:改变< select>元素种的选项或其他表单元素失去桥店，并在其获取角点后内容发生改变时触发
- onclick:鼠标单击触发
- **ondblclick**:鼠标双击时触发
- onfocus:任何元素或是窗口本身获得焦点时触发

  （通常和blur()方法一起使用，使其获得焦点后执行相应函数再取消焦点，否则会一直触发这个事件）

```
<input type="text" id="focus">
<script>
    var f=document.getElementById('focus')
    f.onfocuse=function(){
        alert('明智的选择')
        f.blur()//必须加这个否则会无限循环触发焦点事件
    }
</script>
```

- onkeydown,onkeypress,onkeyup:键盘时间
- onmousedown,onmousemove,onmouseout,onmouseover``onmouseup:鼠标事件

- onload：界面加载完后触发
- onunload:界面卸载后触发
- onreset:单击重置按钮时，在form上触发(form.reset=function(){})
- onsubmit:单击提交按钮时在form上触发
- onresize:窗口的大小发生变化时触发（窗口操作或事件只对使用open打开的生效）
- ***onscroll***:滚动条发生滚动时触发（可以在主窗口生效）
- ***onslect***:选中表单中的文本时触发，如输入文本框或是文本域等所有可以输入文本的地方内容被选中了都会触发。

JavaScript事件处理

js是事件驱动的，可以使得在图形界面环境下的一切操作变得简单化，常将鼠标或热键的动作称为事件（Event），对事件进行处理的程序或是函数称为事件处理函数

鼠标事件

鼠标点击或是移动或是停留在对象上锁触发的事件

- 鼠标单击事件onclick：鼠标被单击时被触发的事件，是指在鼠标按下之后鼠标在目标上没有移动直到放开鼠标健的这一过程

  单击事件一般应用于button对象，checkbox对象，Image对象，Link对象，Radio对象，Reset对象和Sumbit对象
- 鼠标的按下和松开事件：onmousedown和onmouseup,前者是在鼠标按下时触发，后者是在鼠标松开后触发
  - 鼠标的移入移出事件：onmouseover 和onmousemove以及onmouseleave前者是当鼠标移动到这个对象上时触发的动作，后者是当鼠标在对象上移动时触发的事件，最后一个是离开对象时触发的事件

  在鼠标移动事件onmousemove中，可以用document对象实时读取鼠标在页面中的位置

```
event.clientX
event.clientY//读取鼠标当前所在位置的坐标
```

鼠标事件实例如下

```
<!--可以通过按键来使用事件-->
<button onmousedown="change()">改变div的值</button>
function change(){
    a.style.width='500px'
}
```

```
var a=document.getElementById('aa')

//也可以直接在指定元素上使用事件
a.onmousemove=function (){
  a.style.backgroundColor='blue'
}
a.onmouseleave=function (){
```

```
  a.style.backgroundColor='purple'
}

a.onmousedown=function (){
  a.style.fontSize="100px"
}
a.onmouseup=function (){
  a.style.fontSize="10px"
}
```

可以在HTML文件中使用onmousedown来当一个触发，在下面的键盘事件中也同理，当发现你鼠标被点击后进入js程序，在js中根据event.button的值来确定单击的是左键（0）还是右键（2）或是中间(1)

注意：在不同的浏览器中可能会代表不同的意义，如在ie中0为没有按键，1为按下左键

```
<div class="mouse" id="bb" onmousedown="danji_shuangji()">分别用左右键点击试试？</div>

var b=document.getElementById('bb')
function  danji_shuangji(){
var i=event.button
if(i==2){
b.style.backgroundColor='red'
}
if(i==0){
b.style.backgroundColor='green'
}
}
```

下面的键盘事件也是同理，首先使用键盘按下来触发，再在js函数中判断按下的是什么键

键盘事件

使用event.keyCode来获取键码值，对于一些特殊的键位可以直接用event,altKey或event.ctrlKey键盘事件包括如下：

- onkeypress事件：键盘上的某个键被按下且释放所触发的事件，一般用于键盘上的单键操作
- onketdown事件：按下时所触发的事件
- onkeyup事件：松开时所触发的事件

后两种事件一般用于快捷键的操作

键盘上有键码值，我们可以根据键码值来判断按下的是哪个按键。

26个常用字母以及数字的键码值

按键	键值	按键	键值	按键	键值	按键	键值
A	65	J	74	S	83	1	49
B	66	K	75	T	84	2	50
C	67	L	76	U	85	3	51
D	68	M	77	V	86	4	52
E	69	N	78	W	87	5	53
F	70	O	79	X	88	6	54
G	71	P	80	Y	89	7	55
H	72	Q	81	Z	90	8	56
I	73	R	83	0	48	9	57

小键盘以及功能键的键码值

按键	键值	按键	键值	按键	键值	按键	键值
0	96	8	104	F1	112	F9	120
1	97	9	105	F2	113	F10	121
2	98	*	106	F3	114	F11	122
3	99	+	107	F4	115	F12	123
4	100	Enter	108	F5	116		
5	101	-	109	F6	117		
6	102	.	110	F7	118		
7	103	/	111	F8	119		

各种控制键的键码值

按键	键值	按键	键值	按键	键值	按键	键值
Backspace	8	Esc	27	Right Arrow(->)	39	-_	189
Tab	9	Spacebar	32	Down Arrow	40	.>	190
Clear	12	Page Up	33	Insert	45	/?	191
Enter	13	Page Down	34	Delete	46	`~	192
Shift	16	End	35	Num Lock	144	[{	219
Control	17	Home	36	;:	186	|	220
Alt	18	Left Arrow	37	=+	187	]}	221
Cape Lock	20	Up Arrow	38	,<	188	""	222

键盘的触发一般可以放在body里这样不管目前的焦点在哪都可点击键盘触发事件

```
<body class="bg" id="bg" onkeydown="jianpan()">
    <div class="mouse" id="cc" >按按ctrl和alt或a试试？ </div>
```

```
function jianpan(){
  if(event.ctrlKey){
    c.style.backgroundColor='yellow'
    c.style.color='black'
  }
  if(event.altKey){
    c.style.backgroundColor='white'
    c.style.color='black'
  }
  if(event.keyCode=='65'){
    c.style.color='red'
  }
}
```

如果想在页面中屏蔽按键或是鼠标的话在body中加了触发后可以使用以下代码：

```
//禁用键盘上的某个按键
if (event.keyCode==13){
     event.keyCode=0;
     event.returnValue=false;
     alert("当前页面不允许使用回车键")
 }
//也可以禁用组合快捷键
if((event.ctrlKey)&&(event.keyCode==67)){
    event.returnValue=false;
    alert("当前页面不允许使用快捷键复制")
  }
```

```
//或是禁用鼠标的左键或右键
function  pingbi(){
  if(event.button==2){
    event.returnValue=false;
    alert("本页面禁止使用鼠标右键")
  }
```

注意：在屏蔽键盘单键或是组合键是可以不同alert直接屏蔽就行了，但是屏蔽鼠标右键时要是不加alert就屏蔽失败。

JavaScript与表单事件

在扫描检测与操作表单域之前，首先应当确定要访问的表单，JS中有三种方式来访问表单

1. 通过document.forms[]按编号访问（从0开始）
2. 通过名称name访问如document.form1
3. 通过Id访问document.getElementById(form)

如果想继续访问表单中的各个表单元素的某些属性，可以先获得表单，再.name.属性值，如下所示：

```
<form name='form1' id='form'>
    <textarea name=text cols="30" rows="10">ssss</textarea>
</form>
<script>
    document.form1.text.value="aaaaa" //此时文本域中所显示的就是aaaa了而不是ssss
</script>

<!--或是直接设置id，通过ID来确定元素-->
```

所有的表单元素都可以通过.value属性获得用户的输入值,包括但不限于输入框，文本域文件域，单选以及多选框，和菜单

与选择相关的表单就要设置value值了，这时选的是哪个选项就返回对应选项的value值

JavaScript内部对象

js中的内部对象按照使用方法可以分为动态对象和静态对象，在引用动态对象的属性和方法时，必须先使用new关键词来创建一个对象实例，然后才能使用对象实例名.成员的方式来访问其属性和方法。

引用静态对象时，不需要先new一个，可以直接访问其属性和方法

1.Object对象

Object对象提供了对象的基本功能，这些功能构成了所有其他对象的基础，而且构造简单，不需要再定义构造函数。

使用object对象可以在程序运行时为js对象随意添加属性，因此可以容易的创建自定义对象

```
obj=new Object([value])//创建一个新的object对象,value为可选项，可以是任意的js数据类型
```

Object对象的属性

1. prototype属性：为指定对象添加一个方法，如下所示

```
function array_max(){
  var i,max=this[0]
  for(i=1;i<this.length;i++){
    if(max<this[i]){
      max=this[i]
    }
  }
  return max
}
Array.prototype.max=array_max;
var x=new Array(1,2,3,4,5)
var y=x.max()
```

   正如这个案例所示，为Array对象添加了一个函数array_max()作为一个方法，之后对于array的数组对象就可以使用max的方法来求一个数组中的最大值了

2. constructor属性：获得对象声明时的声明类型，如返回事件对象Data,数组对象Array,字符串对象String等

```
x=New String("hi")
if(x.constructor==String) //就这样可以进行对象类型的判断

//也可以对使用函数创建的对象进行判断
function MyFnc{

}
y=new MyFnc;
if(y.constructor==MyFnc)
```

Object对象的方法

1. toLocaleString()方法：有两种用处，
   1. 将指定对象的数据类型转换为字符串类型
   2. 将一堆数组连起来,使用地区特定的分隔符把生成的字符串连接起来，形成一个字符串

```
var arr = new Array(3)
arr[0] = "George"
arr[1] = "John"
arr[2] = "Thomas"
document.write(arr.toLocaleString())//输出为：George, John, Thomas，因为是中国所以使用逗号链接
```

   还有一个toString()方法，直接用,来分割或准换类型，不管地区

2. valueOf():返回指定对象的原始值