



# 第17课：DOM对象高级 和BOM进阶

■ 主讲老师：万章



## 四知

自定义数据属性

元素的尺寸和位置

滚动条大小和位置

元素计算样式获取

浏览器事件初识



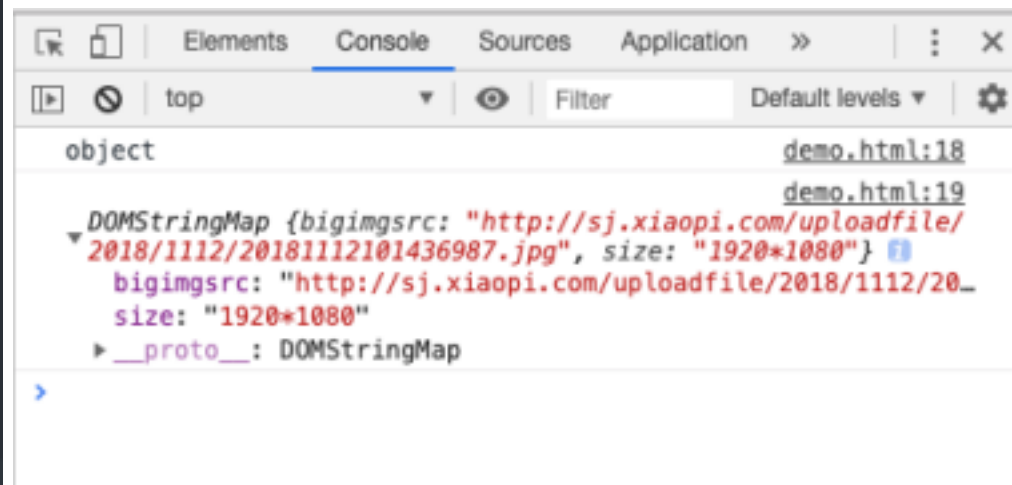
# 自定义数据属性

## 自定义数据属性

HTML5 规定可以为元素添加非标准的属性，但要添加前缀 data-，目的是为元素提供与渲染无关的信息，或者提供语义信息。这些属性可以任意添加、随便命名，只要以 data-开头即可。添加了自定义属性之后，可以通过元素的 dataset 属性来访问自定义属性的值。

```
<body>
  ;

  <script>
    let imgEle=document.querySelector("img");
    let data=imgEle.dataset;
    console.log(typeof data);
    console.log(data);
    imgEle.onload=function(){
      imgEle.src=data.bigimgsrc;
    }
  </script>
</body>
```





# 元素的尺寸和位置

## 元素的尺寸和位置

Obj.offsetHeight: 元素在垂直方向上占用的空间大小, 以像素计。包括元素的高度、 (可见的) 水平滚动条的高度、上边框高度和下边框高度

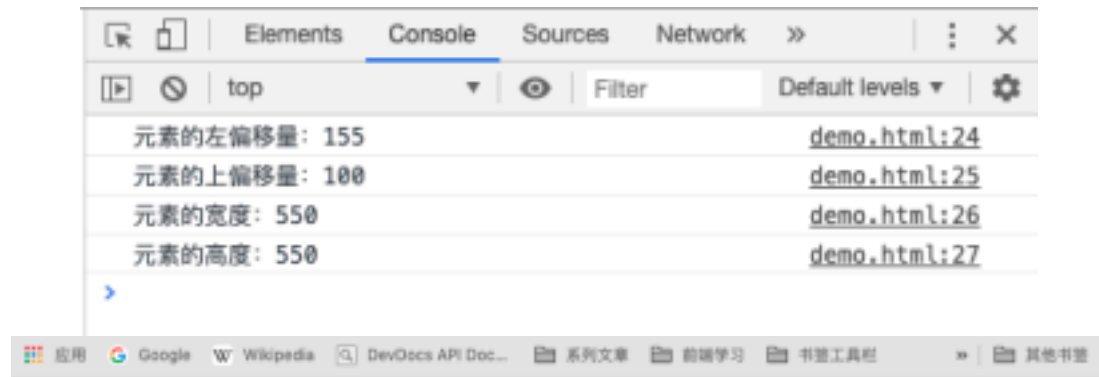
Obj.offsetWidth: 元素在水平方向上占用的空间大小, 以像素计。包括元素的宽度、 (可见的) 垂直滚动条的宽度、左边框宽度和右边框宽度。

Obj.offsetLeft: 元素的左外边框至包含元素的左内边框之间的像素距离

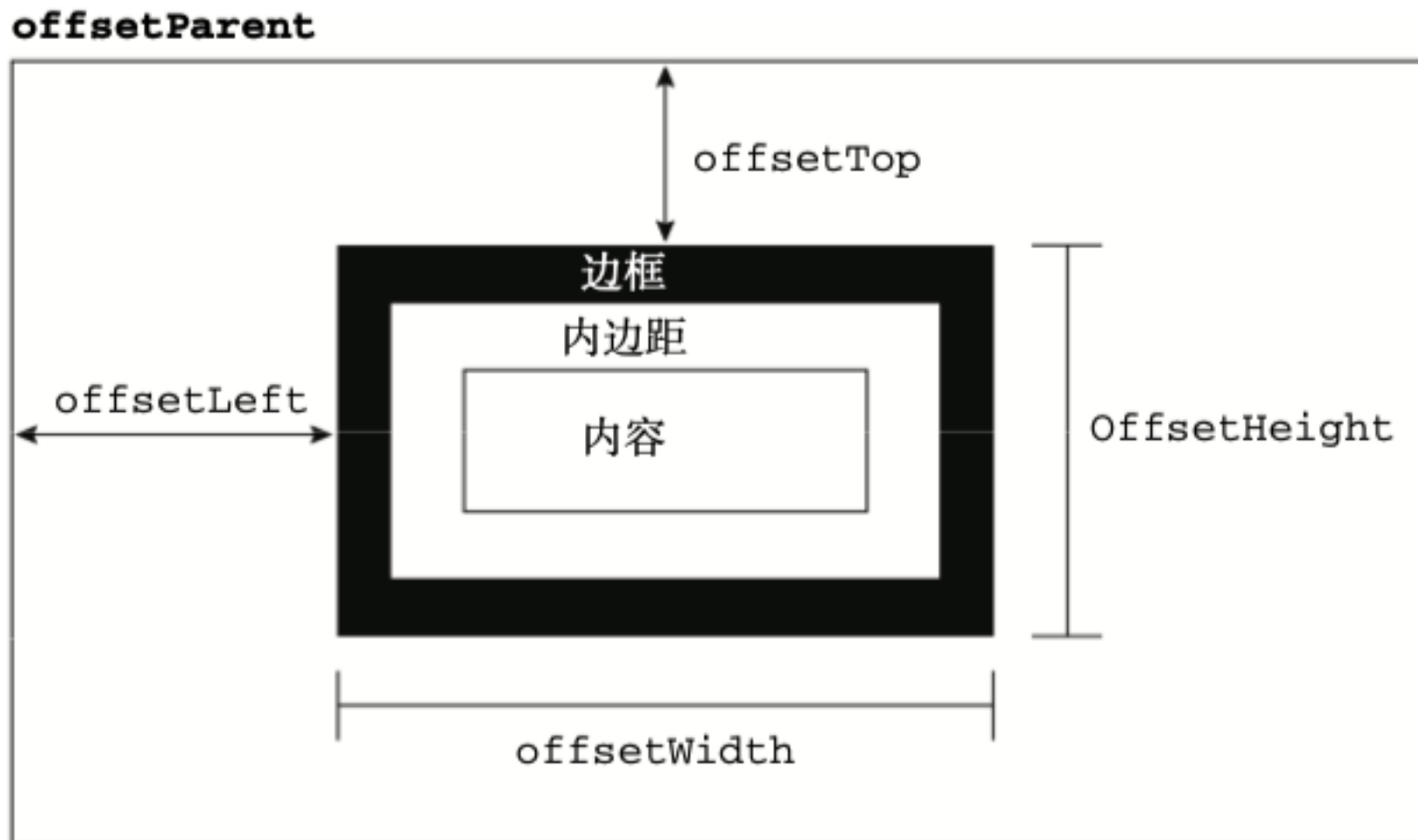
Obj.offsetTop: 元素的上外边框至包含元素的像素距离

```
<style>
  .nav{
    width: 500px;
    height: 500px;
    padding: 20px;
    border: 5px solid red;
    margin: 100px auto;
  }
</style>
</head>

<body>
  <div class="nav"></div>
  <script>
    let nav=document.querySelector(".nav");
    console.log("元素的左偏移量: "+nav.offsetLeft);
    console.log("元素的上偏移量: "+nav.offsetTop);
    console.log("元素的宽度: "+nav.offsetWidth);
    console.log("元素的高度: "+nav.offsetHeight);
  </script>
</body>
```



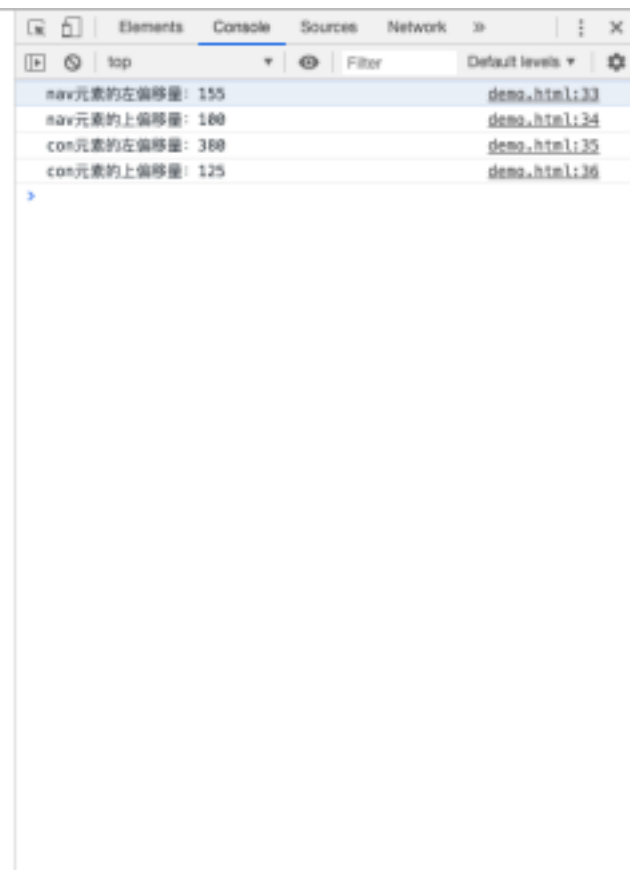
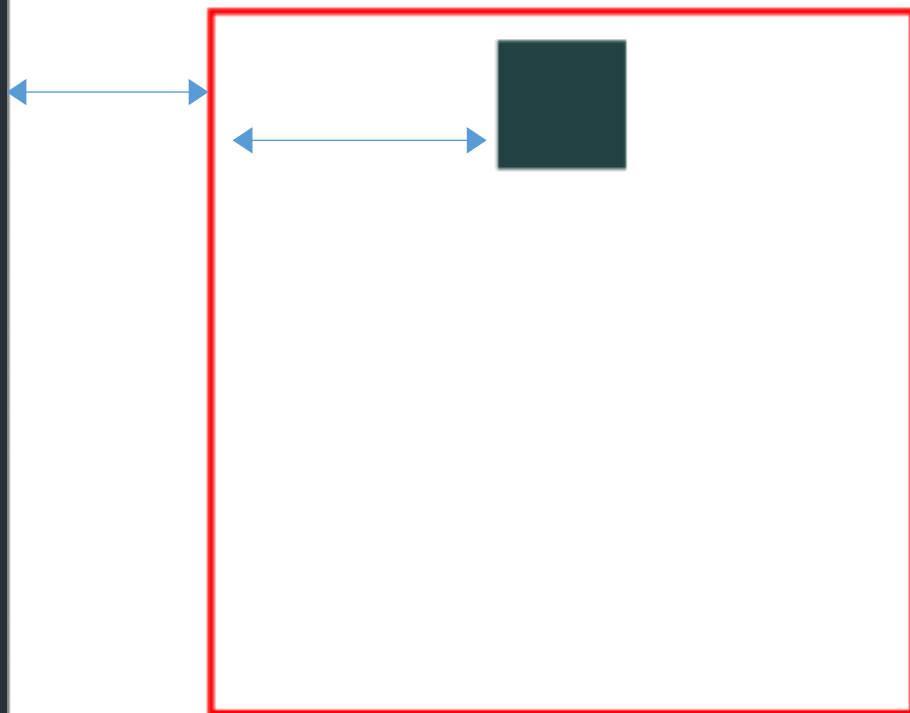
## 元素的位置



注意：元素的offsetTop和Left默认都是相对与浏览器界面边缘的。  
但是要是给元素的某个父元素加一个相对定位，那么就是相对于这个父元素的

# 元素的位置

```
<style>
.nav{
  width: 500px;
  height: 500px;
  padding: 20px;
  border: 5px solid red;
  margin: 100px auto;
}
.con{
  width: 100px;
  height: 100px;
  background-color: rgb(38, 75, 75);
  margin: 0 auto;
}
</style>
</head>
<body>
<div class="nav">
  <div class="con"></div>
</div>
<script>
let nav=document.querySelector(".nav");
let con=document.querySelector(".con");
console.log("nav元素的左偏移量: "+nav.offsetLeft);
console.log("nav元素的上偏移量: "+nav.offsetTop);
console.log("con元素的左偏移量: "+con.offsetLeft);
console.log("con元素的上偏移量: "+con.offsetTop);
</script>
</body>
```



Con元素的左偏移量=nav元素的左偏移量+nav元素的左边框+nav元素的左内边距+  
con元素相对于nav元素的左margin  
=155+5+20+200=380px



## 元素的位置

给元素的某个父元素加一个相对定位，那么该元素的offsetLeft和Top的参考标准就是相对于这个父元素的边框

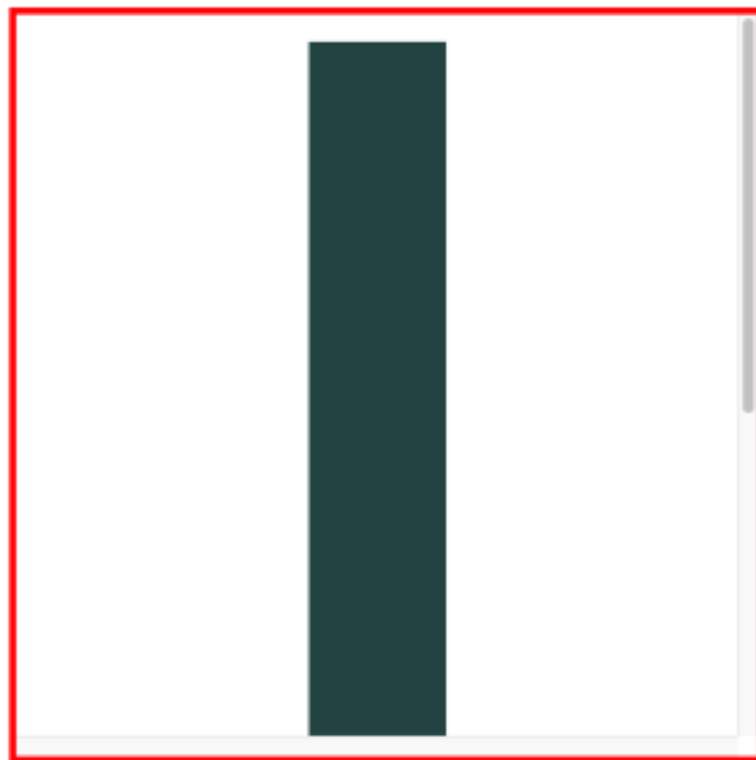
```
<style>
  .nav{
    position: relative;
    width: 500px;
    height: 500px;
    padding: 20px;
    border: 5px solid red;
    margin: 100px auto;
  }
  .con{
    width: 100px;
    height: 100px;
    background-color: rgb(38, 75, 75);
    margin: 0 auto;
  }
</style>
</head>

<body>
  <div class="nav">
    <div class="con"></div>
  </div>
  <script>
    let nav=document.querySelector(".nav");
    let con=document.querySelector(".con");
    console.log("nav元素的左偏移量: "+nav.offsetLeft);
    console.log("nav元素的上偏移量: "+nav.offsetTop);
    console.log("con元素的左偏移量: "+con.offsetLeft);
    console.log("con元素的上偏移量: "+con.offsetTop);
  </script>
</body>
```



Con元素的左偏移量=nav元素的左内边距+con元素相对于nav元素的左margin  
=20+200=220px

## 元素的尺寸和位置（有滚动条的情况下）

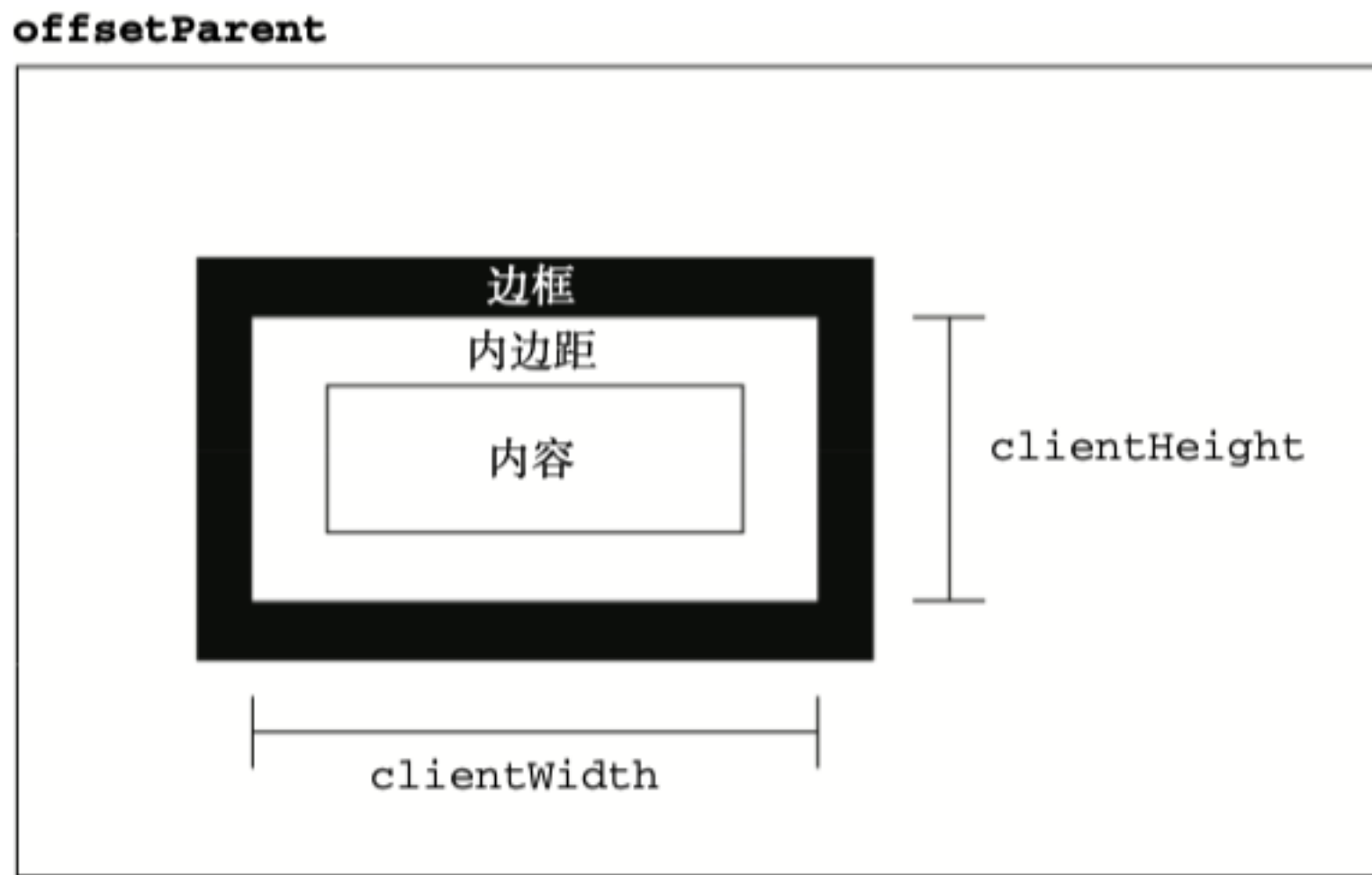


Elements	Console	Sources	Network	⌵	⌵	⌵
top	Filter	Default levels	⌵	⌵	⌵	⌵
nav元素的左偏移量: 155						demo.html:35
nav元素的上偏移量: 100						demo.html:36
nav元素的宽度: 550						demo.html:37
nav元素的高度: 550						demo.html:38
con元素的左偏移量: 213						demo.html:39
con元素的上偏移量: 20						demo.html:40

Con元素的左偏移量=nav元素的左内边距+con元素相对于nav元素的左margin  
=20+193(滚动条的出现导致子元素可以用作外边距的区域减小了)=213px

元素的宽度和尺寸没有变，滚动条在原来的宽度内出现

## 元素的大小（客户区大小）



`Obj.clientHeight`、`Obj.clientWidth`;

客户区大小就是元素内部的空间大小，因此滚动条占用的空间不计算在内

元素的尺寸和位置信息（终极方

# 案) Obj.getBoundingClientRect()



Top、bottom:元素的上下边相对于浏览器上边缘的间距

left、right:元素的左右边相对于浏览器上边缘的间距

Width: 元素的宽度 (包含width, padding, border)

height: 元素的高度 (包含height, padding, border)

X,y: 元素左上角相对于浏览器上左边缘的坐标值





# 滚动条大小和位置

# DOM元素的滚动条大小

scrollHeight: 在没有滚动条的情况下, 元素内容的总高度

scrollWidth: 在没有滚动条的情况下, 元素内容的总宽度

scrollLeft: 被隐藏在内容区域左侧的像素数。通过设置这个属性可以改变元素的横向滚动条位置

scrollTop: 被隐藏在内容区域上方的像素数。通过设置这个属性可以改变元素的竖向滚动条位置。



Elements	Console	Sources	Network	⌵	⌵	⌵
top	Filter	Default levels	⌵	⌵	⌵	⌵
nav元素的左偏移量: 155						demo.html:35
nav元素的上偏移量: 188						demo.html:36
nav元素的宽度: 558						demo.html:37
nav元素的高度: 558						demo.html:38
nav元素的滚动条高度: 948						demo.html:39
nav元素的滚动条宽度: 828						demo.html:40
nav元素的竖向滚动条高度: 8						demo.html:41
nav元素的横向滚动条宽度: 8						demo.html:42



# 元素计算样式获取

## 元素计算样式获取方式之getComputedStyle

window . getComputedStyle( { }, 伪元素名称 )

window.getComputedStyle(element, [pseudoElt]);//返回的是实时的 CSSStyleDeclaration 对象，这个方法接受两个参数： 要取得计算样式的元素和一个伪元素字符串（例如“:after”）。如果不需要伪元素信息，第二个参数可以是 null。Null则为直接获取元素本身的样式，如果是其他 “:after、:before”，则获取其他伪元素的样式

当元素的样式更改时，它会自动更新本身。 要获得里面的某个样式的话们可以再用 .getPropertyValue(“样式名称” ) 或是.样式名称

此处获取的是元素的最终样式，如果元素有多条css语句作用于同一个样式，那么我们js获取的也只是最终的css的值



## 元素计算样式获取方式之currentStyle ( )

Obj .currentStyle ( “样式名称” )

该方法适用于IE浏览器，目前已很少有人使用  
只有在处理兼容性问题时方才会奏效



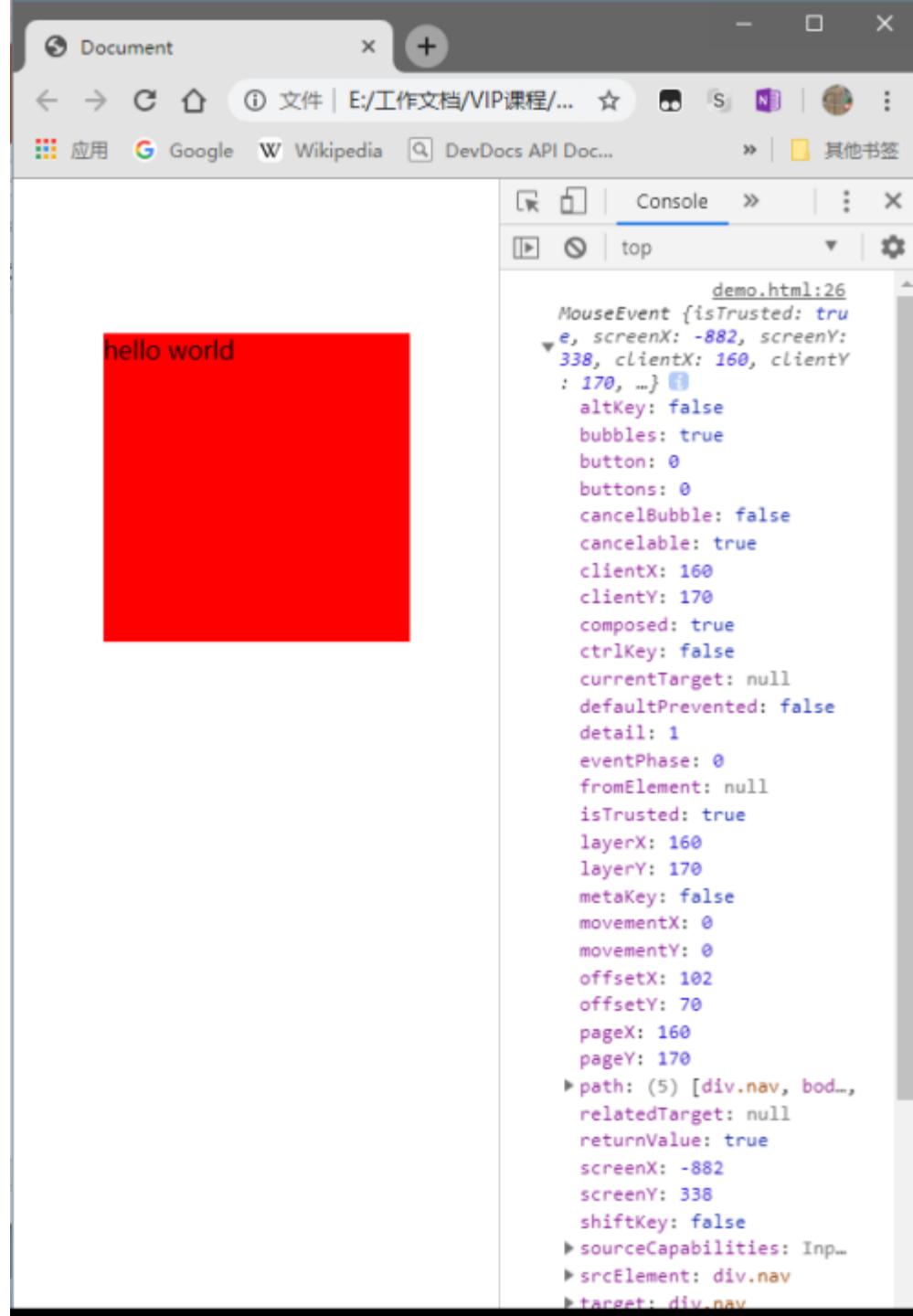
# 浏览器事件初识

# 浏览器事件初识

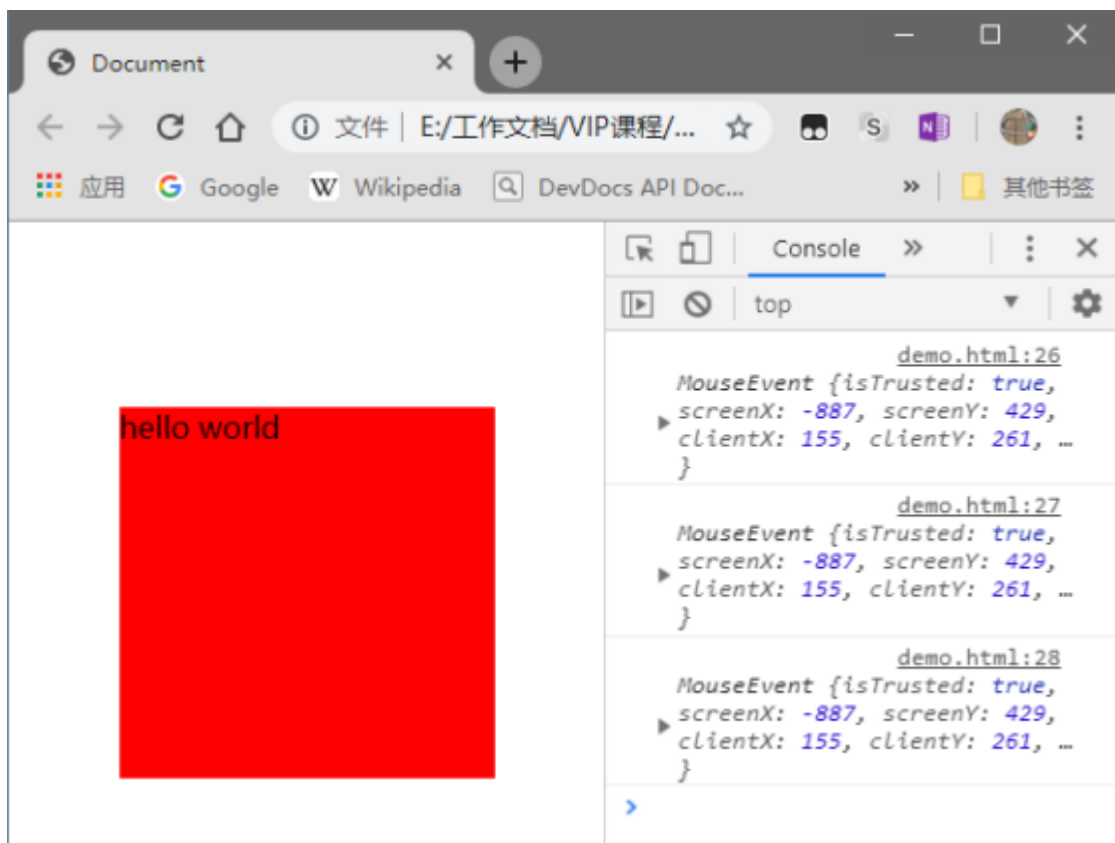
由事件触发的函数  
在运行的时候,浏览器  
会往函数中传入  
一个参数,这个参数  
就是浏览器鼠标事件,  
属于BOM

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>
  <style>
    body{
      margin: 0;
    }
    .nav{
      width: 200px;
      height: 200px;
      background-color: rgb(78, 78, 145);
      margin: 100px auto;
    }
  </style>
</head>
<body>
  <div class="nav">hello world</div>

  <script>
    let nav=document.querySelector(".nav");
    nav.onclick=function(ev){
      console.log(ev);
      nav.style.backgroundColor="red";
    }
  </script>
</body>
</html>
```



# 浏览器事件初识



```
<body>
  <div class="nav">hello world</div>

  <script>
    let nav=document.querySelector(".nav");
    nav.onclick=function(ev){
      console.log(ev);
      console.log(window.event);
      console.log(event);
      nav.style.backgroundColor="red";
    }
  </script>
</body>
```

获得鼠标事件的方式有多种:

第一种的方法就是直接在定义函数的时候传入一个参数

第二种的方法就是在函数里面直接获取window.event或是event(道理和var一个变量一样, var a; 那么a或是window.a 都可以获取到变量的值)

这两个方法都能获取同一个事件(原理是 浏览器同一时刻只能执行一个事件 所以同时获取到的一定是同一个)

# 浏览器事件参数初识

鼠标事件对象的属性数量非常多, 我们先拿几个常见的属性开始了解

## 1. 鼠标位置类:

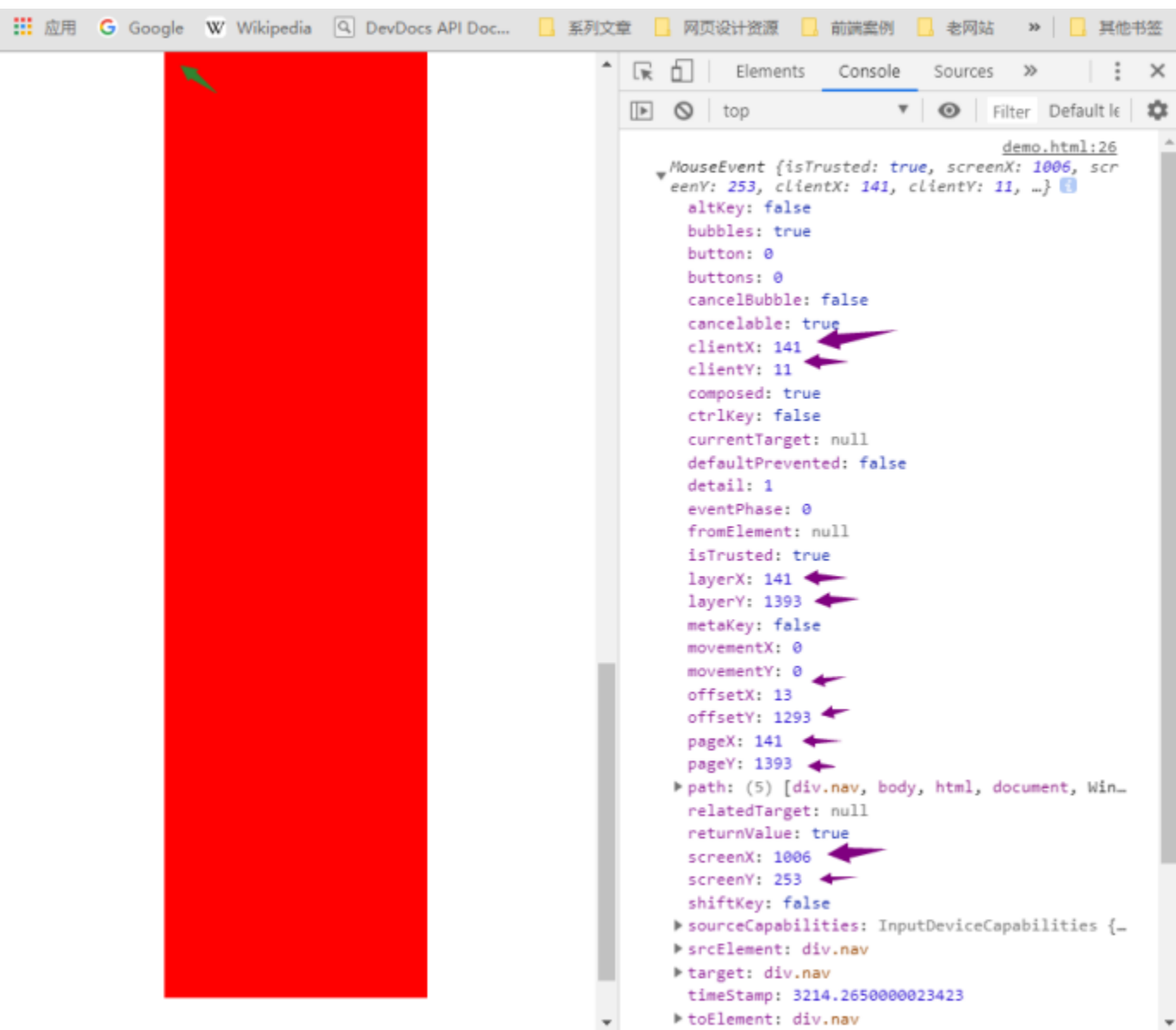
1. clientX,clientY
2. offsetX,offsetY
3. pageX,pageY
4. screenX,screenY
5. x,y
6. layerX,layerY

## 2. 目标元素类

7. target
8. srcElement
9. path

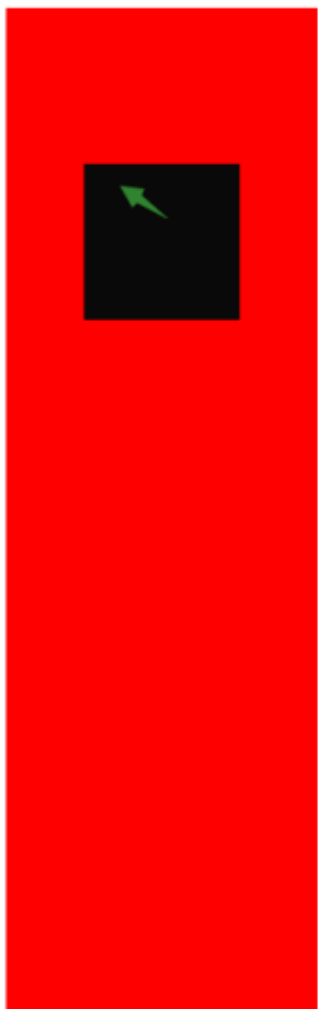
```
demo.html:28
MouseEvent {isTrusted: true, screenX: 800, screenY: 304, clientX: 800, clientY: 170, ...}
  altKey: false
  bubbles: true
  button: 0
  buttons: 0
  cancelBubble: false
  cancelable: true
  clientX: 800
  clientY: 170
  composed: true
  ctrlKey: false
  currentTarget: null
  defaultPrevented: false
  detail: 1
  eventPhase: 0
  fromElement: null
  isTrusted: true
  layerX: 800
  layerY: 170
  metaKey: false
  movementX: 0
  movementY: 0
  offsetX: 78
  offsetY: 70
  pageX: 800
  pageY: 170
  path: (5) [div.nav, bod..., relatedTarget: null, returnValue: true, screenX: 800, screenY: 304, shiftKey: false, sourceCapabilities: Inp..., srcElement: div.nav, target: div.nav, timeStamp: 566.20499999..., toElement: div.nav, type: "click", view: Window {postMessa..., which: 1, x: 800, y: 170, __proto__: MouseEvent
```

# 浏览器事件参数之鼠标位置类



1. `clientX`、`clientY` 点击位置距离当前 `body` 可视区域的 `x`，`y` 坐标
2. `pageX`、`pageY` 对于整个页面来说，包括了被滚动条滚过去的 `body` 部分的长度
3. `screenX`、`screenY` 点击位置距离当前电脑屏幕最左上角的 `x`，`y` 坐标
4. `offsetX`、`offsetY` 相对于被点中的元素的坐左上角的偏移量(即便当前元素有部分被滚动条隐藏了, 这个偏移量也是按照实际元素的高度来的)
5. `x`、`y` 和 `clientX`、`clientY` 一样
6. `layerX` 和 `layerY`, 鼠标相比较于当前坐标系的位置, 即如果触发元素没有设置绝对定位或相对定位, 以页面为参考点, 如果有, 则以当前触发元素的左上角为坐标系

# 浏览器事件参数之目标元素类



```
demo.html:36
▼ MouseEvent {isTrusted: true, screenX: -1209, screenY: 443, clientX: 217, clientY: 223, ...}
  altKey: false
  bubbles: true
  button: 0
  buttons: 0
  cancelBubble: false
  cancelable: true
  clientX: 217
  clientY: 223
  composed: true
  ctrlKey: false
  currentTarget: null
  defaultPrevented: false
  detail: 1
  eventPhase: 0
  fromElement: null
  isTrusted: true
  layerX: 38
  layerY: 23
  metaKey: false
  movementX: 0
  movementY: 0
  offsetX: 39
  offsetY: 23
  pageX: 217
  pageY: 223
  path: (6) [div.con, div.nav, body, html, docu...
  relatedTarget: null
  returnValue: true
  screenX: -1209
  screenY: 443
  shiftKey: false
  sourceCapabilities: InputDeviceCapabilities {...}
  srcElement: div.con
  target: div.con
  timeStamp: 1164.1800000070361
  toElement: div.con
```

1. **event.srcElement**: 表示可以获取当前触发事件的对象。
2. **event.target**: 事件属性可返回事件的目标节点（触发该事件的节点），如生成事件的元素、文档或窗口。

**srcElement**是IE下的属性  
**target**是Firefox下的属性  
Chrome浏览器同时有这两个属性  
这两个其实是同一个东西

```
▼ path: Array(6)
  ► 0: div.con
  ► 1: div.nav
  ► 2: body
  ► 3: html
  ► 4: document
  ► 5: Window {postMessage: f, blur: f, focus: ...}
  length: 6
```

**path**是一个数组,里面第一个是触发事件的元素, 第二个是该元素的父元素,然后类推

# 浏览器事件参数之事件类型类



```
cancelable: true
clientX: 217
clientY: 223
composed: true
ctrlKey: false
currentTarget: null
defaultPrevented: false
detail: 1
eventPhase: 0
fromElement: null
isTrusted: true
layerX: 38
layerY: 23
metaKey: false
movementX: 0
movementY: 0
offsetX: 39
offsetY: 23
pageX: 217
pageY: 223
path: (6) [div.con, div.nav, body, html, docu...
relatedTarget: null
returnValue: true
screenX: -1209
screenY: 443
shiftKey: false
sourceCapabilities: InputDeviceCapabilities {...
srcElement: div.con
target: div.con
timestamp: 1164.1800000070361
toElement: div.con
type: "click"
view: Window {postMessage: f, blur: f, focus: ...
  which: 1
  x: 217
  y: 223
  __proto__: MouseEvent
```

## 焦点事件

事件名称	何时触发
<code>focus</code>	元素获得焦点 (不会冒泡)
<code>blur</code>	元素失去焦点 (不会冒泡)

## 表单事件

事件名称	何时触发
<code>reset</code>	点击重置按钮时
<code>submit</code>	点击提交按钮

## 鼠标事件

Event Name	Fired When
<code>mouseenter</code>	指针移到有事件监听的元素内
<code>mouseover</code>	指针移到有事件监听的元素或者它的子元素内
<code>mousemove</code>	指针在元素内移动时持续触发
<code>mousedown</code>	在元素上按下任意鼠标按钮
<code>mouseup</code>	在元素上释放任意鼠标按钮
<code>click</code>	在元素上按下并释放任意鼠标按钮
<code>dblclick</code>	在元素上双击鼠标按钮
<code>contextmenu</code>	右键点击 (右键菜单显示前)
<code>wheel</code>	滚轮向任意方向滚动
<code>mouseleave</code>	指针移出元素范围外 (不冒泡)
<code>mouseout</code>	指针移出元素, 或者移到它的子元素上
<code>select</code>	文本被选中/被选中
<code>pointerlockchange</code>	鼠标被锁定或者解除锁定发生时
<code>pointerlockerror</code>	可能因为一些技术的原因鼠标锁定被禁止时。