



## 第三课：基础DOM操作

■ 主讲老师：万章



## 目录

DOM的核心理念

JavaScript获取元素

元素的自有属性

元素的自定义属性



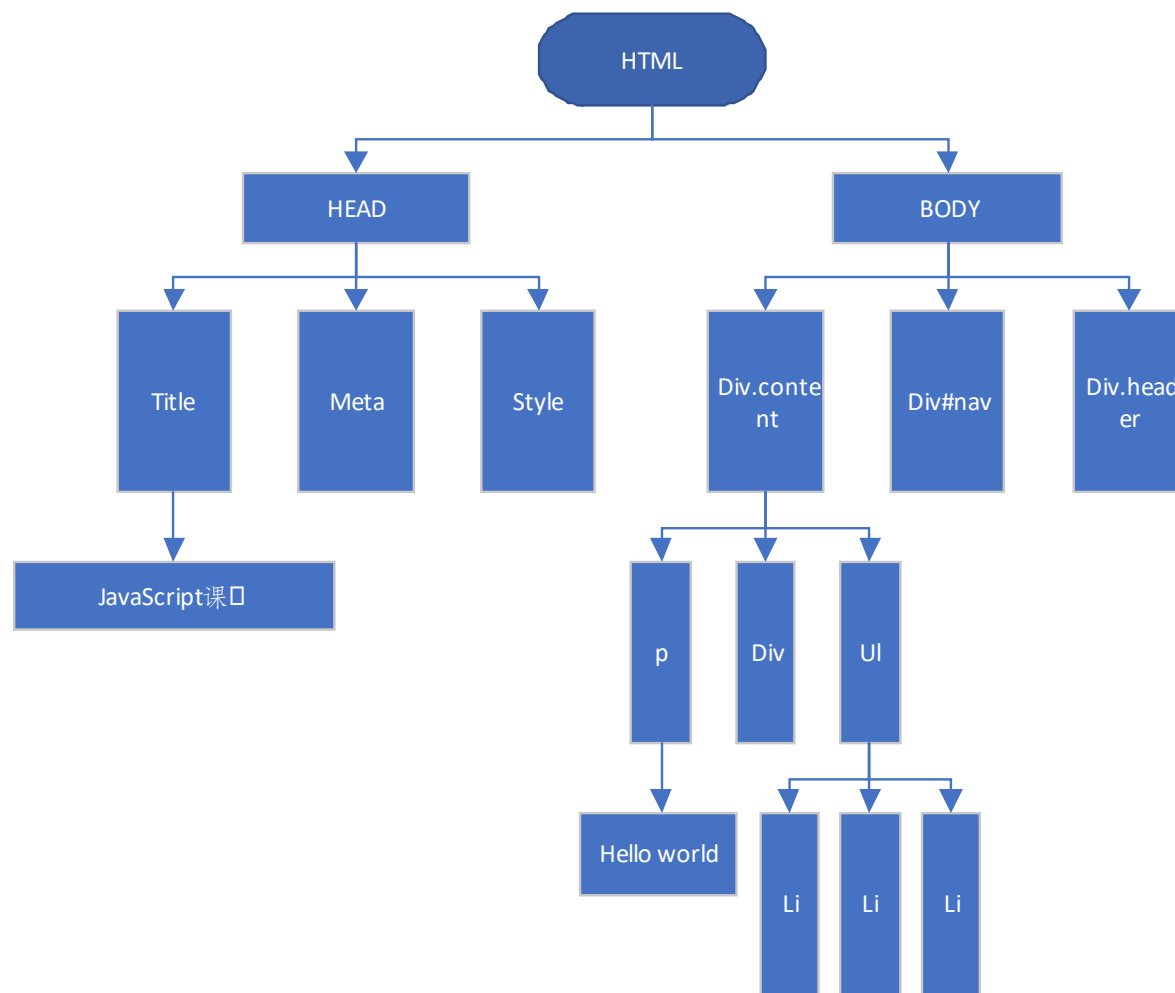
# DOM的核心理念

# 什么是DOM?

DOM (文档对象模型 document object model) 是针对HTML 文档的一个API  
(应用程序编程接口Application Programming Interface)

DOM描绘了一个层次化的节点树, 允许开发人员**添加、移除和修改**页面元素(元素的文字也是DOM的一个节点)

```
<!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>JavaScript课程</title>
</head>
<body>
  <div class="content">
    <p>hello world</p>
    <div></div>
    <ul>
      <li></li>
      <li></li>
      <li></li>
    </ul>
  </div>
  <div id="nav"></div>
  <div class="header"></div>
</body>
</html>
```





# JavaScript获取元素

# JavaScript获取元素之ID获取

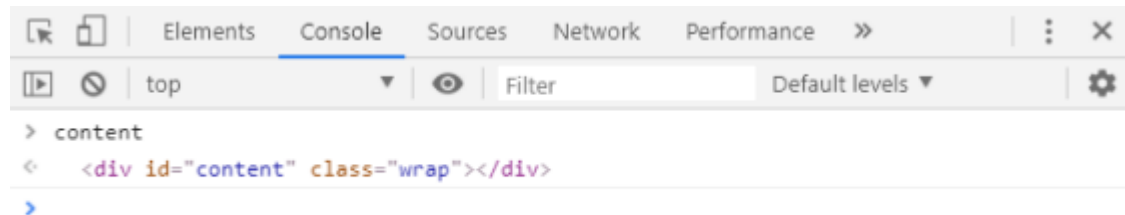
```
<div id="content" class="wrap"></div>
<script>
    var content=document.getElementById("content");
    /*
        document 文本
        get得到
        element元素
        By通过
        Id名
    */
</script>
```

DOM (文档对象模型 document object model)

document就是文档对象模型里面的文档对象

因为一个网页上id只会出现一次,所以我们通过id所获取的元素也只有一个, 就是元素本身

注:即便网页有多个元素的id名称为同一个, 那么js也只会选中第一个id名称符合条件的元素



# JavaScript获取元素之class获取

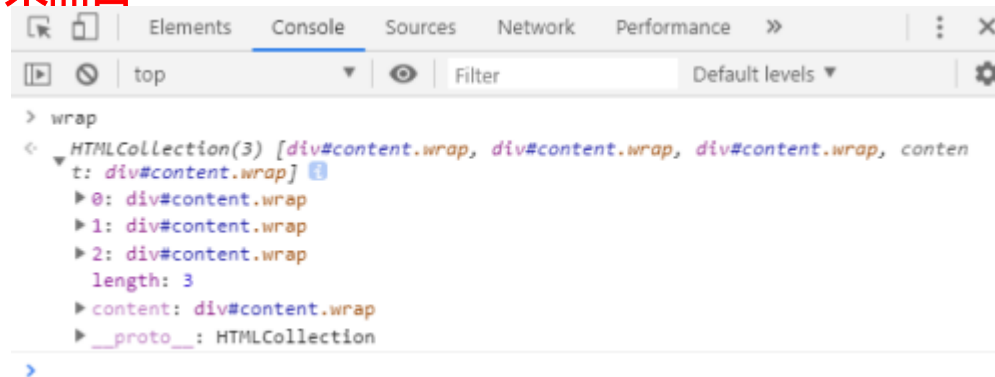
```
<div id="content" class="wrap"></div>
<div id="content" class="wrap"></div>
<div id="content" class="wrap"></div>
<script>
    var wrap=document.getElementsByClassName( "wrap" );
    /*
        document 文本
        get得到
        element元素
        By通过
        ClassName名称
    */
</script>
```

DOM（文档对象模型 document object model）

document就是文档对象模型里面的文档对象

因为一个网页上class会出现多次,所以我们通过class所获取的元素会有多个, 所以此时变量wrap指向的是内存空间里面的一个数组,数组里面存放了网页上所有class名称为wrap的元素

注:即便网页上只有一个元素的class名称符合条件,通过js获取到的依然会是一个数组,只是该数组里面的值只有一个而已



# JavaScript获取元素之标签名获取

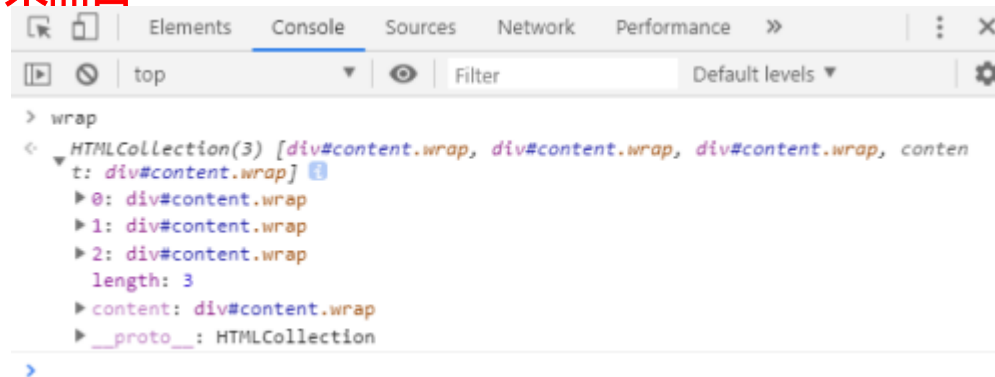
```
<div id="content" class="wrap"></div>
<div id="content" class="wrap"></div>
<div id="content" class="wrap"></div>
<script>
    var wrap=document.getElementsByTagName( "div" );
    /*
        doument 文本
        get得到
        element元素
        By通过
        TagName名称
    */
</script>
```

DOM (文档对象模型 document object model)

document就是文档对象模型里面的文档对象

因为一个网页上同一类标签会出现多次,所以我们通过TagName(标签名)所获取的元素会有多个, 所以此时变量wrap指向的是内存空间里面的一个数组, 数组里面存放了网页上所有标签名称为div的元素

注:即便网页上只有一个元素的标签名称符合条件,通过js获取到的依然会是一个数组,只是该数组里面的值只有一个而已





# JavaScript获取元素之name获取

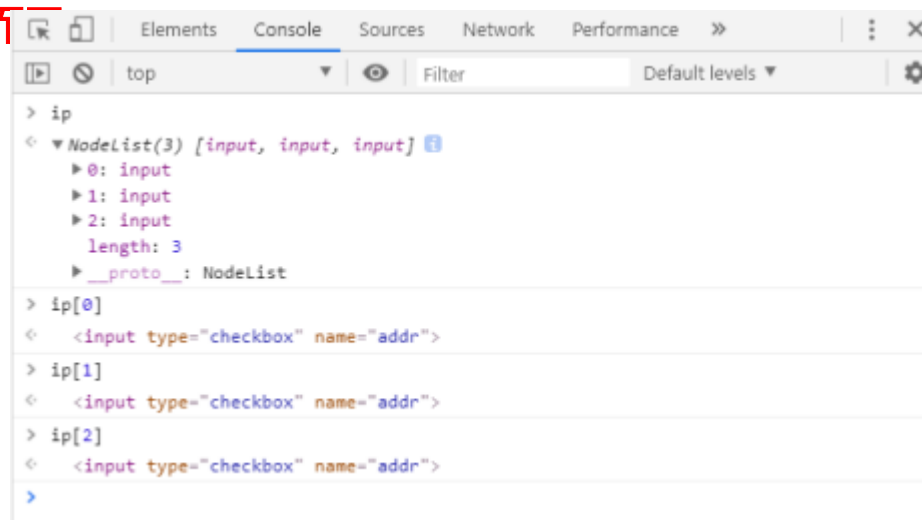
```
<input type="checkbox" name="addr">
<input type="checkbox" name="addr">
<input type="checkbox" name="addr">
<script>
    var ip=document.getElementsByName("addr");
    /*
        document 文本
        get得到
        element元素
        By通过
        Name名称
    */
</script>
```

DOM (文档对象模型 document object model)

document就是文档对象模型里面的文档对象

因为一个网页上拥有相同的name的值的元素会出现多次,所以我们通过Name所获取的元素会有多个,所以此时变量addr指向的是内存空间里面的一个数组,数组里面存放了网页上所有name名称为指定name值的元素

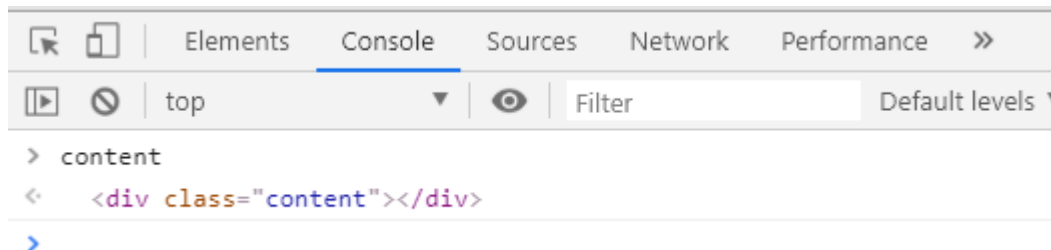
注:即便网页上只有一个元素的name名称符合条件,通过js获取到的依然会是一个数组,只是该数组里面的值只有一个而一



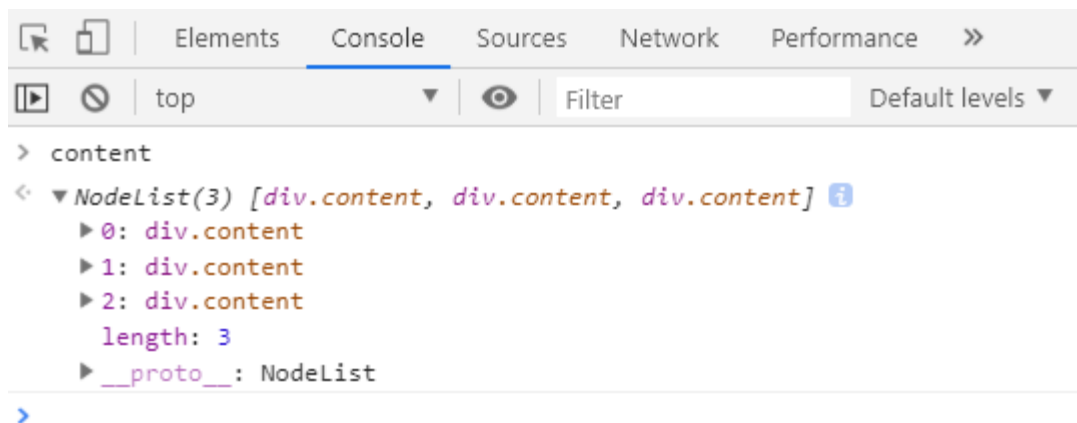
# JavaScript获取之选择器获取

```
<div class="content"></div>
<div class="content"></div>
<div class="content"></div>
<script>
  var content=document.querySelector(".content");
  /*
    document 文本
    query查询
    Selector选择器
  */
</script>
```

```
<div class="content"></div>
<div class="content"></div>
<div class="content"></div>
<script>
  var content=document.querySelectorAll(".content");
  /*
    document 文本
    query查询
    Selector选择器
    All 所有的元素
  */
</script>
```



document.querySelector(“选择器代码”)  
无论选择器实际选中的是多少,只会选中符合选择器条件的第一个元素,js代码返回的最终是单个元素



document.querySelectorAll(“选择器代码”)  
选择器选择中多少就是就是多少, js代码返回的是一个数组, 即便选择器只选择了一个元素, 返回的也是一个数组

## JavaScript获取之特殊元素获取



```
<script>
```

```
document.documentElement//获取整个结构文档,html架构
```

```
document.head//获取我们的头部标签head
```

```
document.body//获取body标签
```

```
document.title//获取标题标签
```

```
</script>
```



# 元素的自有属性

## 元素的自有属性之内容属性

ob.innerText="字符串"

设置一个元素内部的文字内容

注:

- 1)如果元素内容原来就有一些字符内容, 那么原来的内容会被替换成js代码所设置的
- 2)即便字符串内部有一些元素标签, 这些元素标签也会被当做字符串显示在网页上

```
<div class="content"></div>
<script>
  var content=document.querySelector(".content");
  content.innerText="万章大帅比";
</script>
```

```
<div class="content"></div>
<script>
  var content=document.querySelector(".content");
  content.innerText="<p>万章大帅比</p>";
</script>
```

万章大帅比

```
Elements Console >>
<!doctype html>
<html lang="en">
  ><head>...</head>
  ><body>
    ... <div class="content">万章大帅比</div> == $0
      ><script>...</script>
    </body>
  </html>
```

<p>万章大帅比</p>

```
Elements Console >>
<!doctype html>
<html lang="en">
  ><head>...</head>
  ><body>
    ... <div class="content"><p>万章大帅比</p></div> == $0
      ><script>...</script>
    </body>
  </html>
```

# 元素的自有属性之内容属性

ob.innerHTML="字符串"  
设置一个元素内部的HTML代码

注:

- 1)如果元素内容原来就有一些字符或是元素内容, 那么原来的内容会被替换成js代码所设置的
- 2)如果innerHTML的内容是一些纯文字, 那么效果如同innerText一样
- 3)如果innerHTML的内容中包含了一些标签元素, 那么这个标签元素会直接加入到网页中

```
<div class="content"></div>
<script>
    var content=document.querySelector(".content");
    content.innerHTML="万章大帅比";
</script>
```

```
<div class="content"></div>
<script>
    var content=document.querySelector(".content");
    content.innerHTML="<p>万章大帅比</p>";
</script>
```

万章大帅比

Elements Console

```
<!doctype html>
<html lang="en">
  <head>...</head>
  <body>
    ... <div class="content">万章大帅比</div> == $0
      <script>...</script>
    </body>
  </html>
```

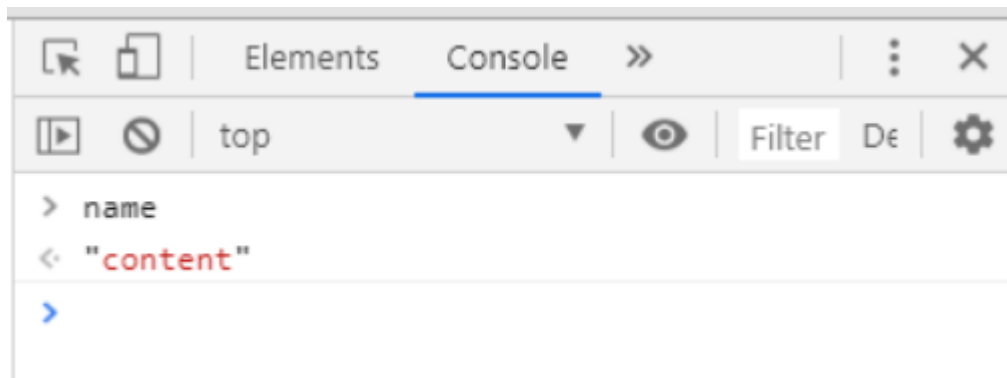
万章大帅比

Elements Console

```
<!doctype html>
<html lang="en">
  <head>...</head>
  <body>
    ... <div class="content"> == $0
      <p>万章大帅比</p>
    </div>
    <script>...</script>
  </body>
</html>
```

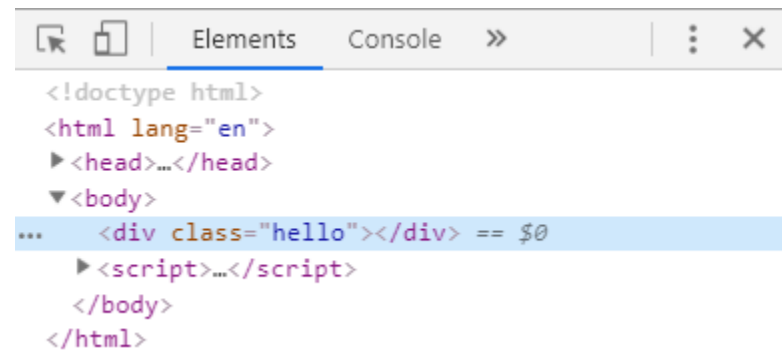
## 元素的自有属性之class名称属性

```
<div class="content"></div>
<script>
  var content=document.querySelector(".content");
  var name=content.className;
</script>
```



获取class名称

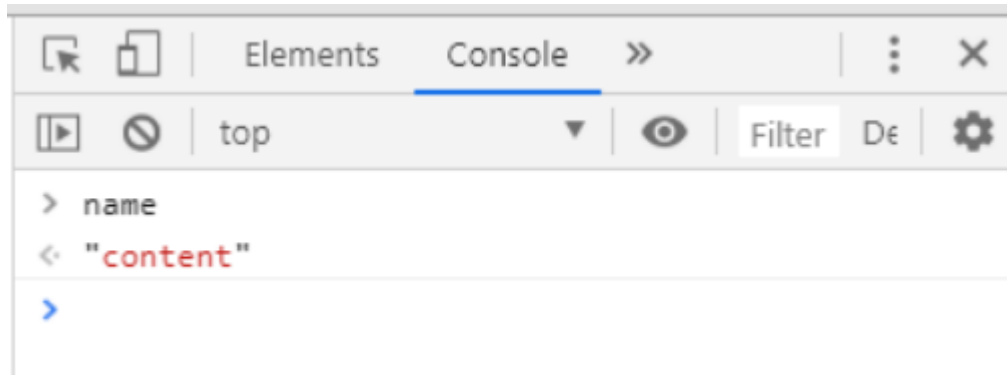
```
<div class="content"></div>
<script>
  var content=document.querySelector(".content");
  content.className="hello";
</script>
```



设置class名称

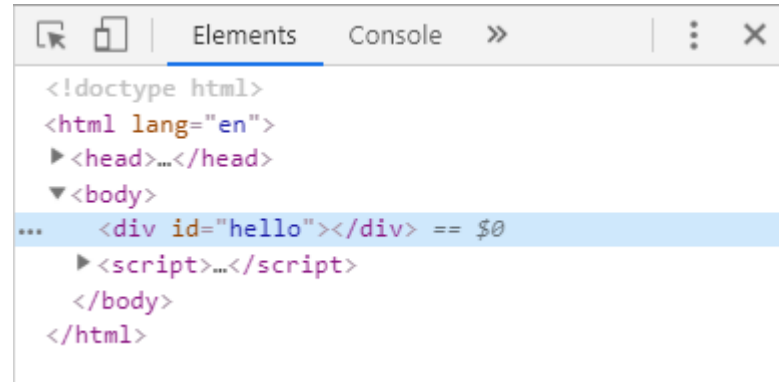
## 元素的自有属性之id名称属性

```
<div id="content"></div>
<script>
  var content=document.querySelector("#content");
  var name=content.id;
</script>
```



获取id名称

```
<div id="content"></div>
<script>
  var content=document.querySelector("#content");
  content.id="hello";
</script>
```



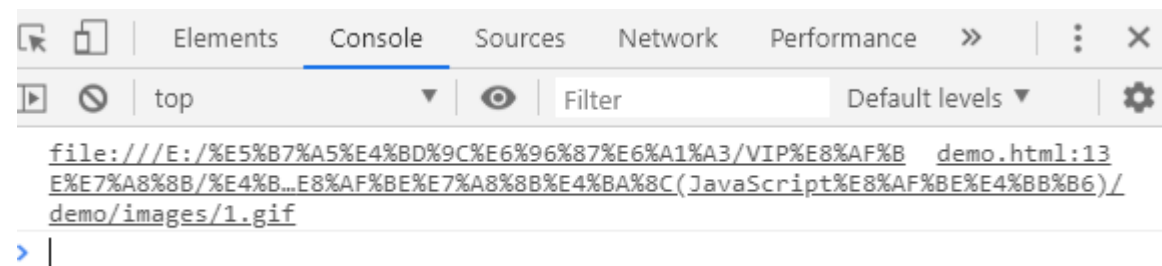
设置id名称



## 元素的自有属性之src属性

```

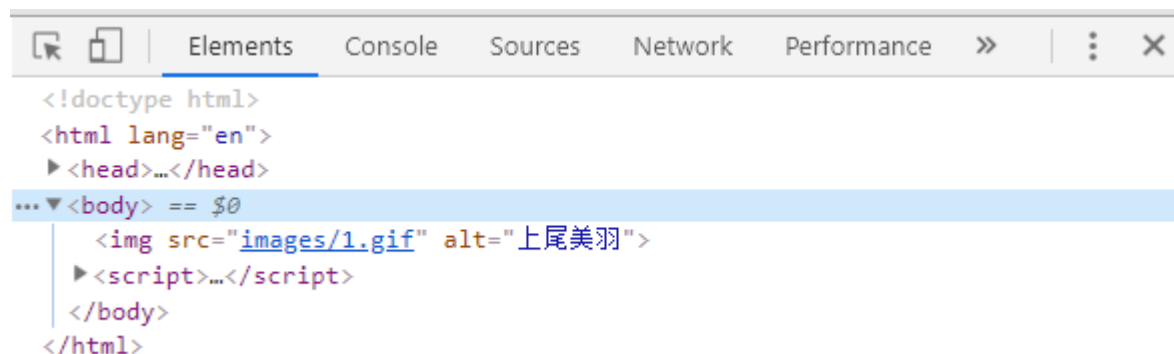
<script>
    var content=document.querySelector("img");
    console.log(content.src)
</script>
```



获取img元素的src属性

注:js获取到的src路径是本地盘符出发的绝对路径

```
<img src="" alt="上尾美羽">
<script>
    var content=document.querySelector("img");
    content.src="images/1.gif"
</script>
```



设置img元素的src属性

注:用js设置img元素的src路径时, 设置的字符串是啥就是啥

## 元素的自有属性之通用获取方法

```

<div name="world"></div>
<script>
  var content=document.querySelector("img");
  var wrap=document.querySelector("div");
  console.log(content.alt)
  console.log(content.name)
  console.log(wrap.name);
</script>
```

Elements	Console	Sources	Network	Performance	>>	⋮	×
▶	⊘	top	▼	👁	Filter	Default levels ▼	⚙
上尾美羽						demo.html:15	
hello						demo.html:16	
undefined						demo.html:17	

即便是同一属性,在不同的元素下同样的方式不一定能得到相同的结果

所以ob.属性的方式并不能得到所有我们需要的元素属性的值  
这时我们就要引入一个新的方法

ob.getAttribute("属性名称");//获取 attribute 属性

```

<div name="world"></div>
<script>
  var content=document.querySelector("img");
  var wrap=document.querySelector("div");
  console.log(content.getAttribute("alt"))
  console.log(content.getAttribute("name"))
  console.log(content.getAttribute("src"))
  console.log(wrap.getAttribute("name"));
</script>
```

Elements	Console	Sources	Network	Performance	>>	⋮	×
▶	⊘	top	▼	👁	Filter	Default levels ▼	⚙
上尾美羽						demo.html:15	
hello						demo.html:16	
images/1.gif						demo.html:17	
world						demo.html:18	

## 元素的自有属性之通用设置方法

如上述所言, 因为`ob.属性名称`的方式获取元素属性的方式不一定能都奏效,那么通过该方式设置元素属性同样也不一定就能奏效,所以我们需要一个通用的设置元素属性的方式  
这时我们就要引入一个新的方法

`ob.setAttribute("属性名称","属性值");//set设置 attribute属性`

作用就是给ob元素设置一个属性, 属性名称为"属性名称",属性值为"属性值"

```
<img>
<div></div>
<script>
  var content=document.querySelector("img");
  var wrap=document.querySelector("div");
  content.setAttribute("alt","上尾美羽");
  content.setAttribute("name","hello");
  content.setAttribute("src","images/1.gif");
  wrap.setAttribute("name","world");
</script>
```



```
<!doctype html>
...<html lang="en"> == $0
  <head>...</head>
  <body style>
    
    <div name="world"></div>
    <script>...</script>
    <!-- Code injected by live-server -->
    <script type="text/javascript">...</script>
  </body>
</html>
```

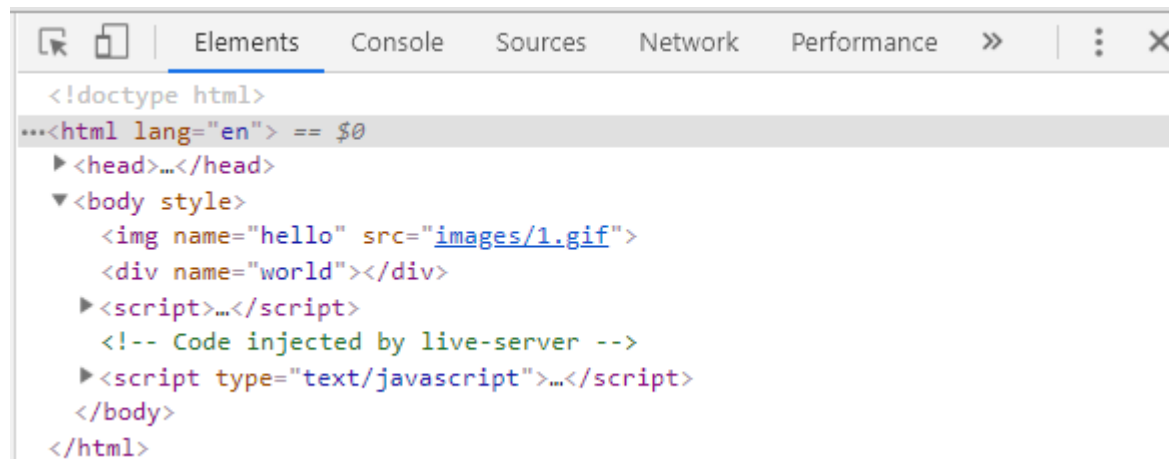
## 元素的自有属性之通用删除方法

有些时候我们不再需要某个元素再有啥属性时, 我们可以用js来删除元素的属性, 比如表单元素的默认被选中, 默认禁用这样的功能就是用一个属性来实现的

删除属性的方式是:

ob.removeAttribute("属性名称");//删除ob元素的名称为"属性名称"的属性

```
<img alt="上尾美羽">
<div></div>
<script>
  var content=document.querySelector("img");
  var wrap=document.querySelector("div");
  content.setAttribute("name","hello");
  content.setAttribute("src","images/1.gif");
  wrap.setAttribute("name","world");
  content.removeAttribute("alt");
</script>
```



```
<!doctype html>
...<html lang="en"> == $0
  <head>...</head>
  <body style>
    
    <div name="world"></div>
    <script>...</script>
    <!-- Code injected by live-server -->
    <script type="text/javascript">...</script>
  </body>
</html>
```



# 元素的自定义属性

# 元素的自定义属性

所谓自定义属性就是HTML标签原生不支持,但是开发人员为了开发方便自行给元素加上去的属性  
设置方式就是前面讲到的setAttribute  
读取方式就是getAttribute  
删除的方式也是removeAttribute

```
<img alt="上尾美羽">
<script>
  var content=document.querySelector("img");
  content.setAttribute("size","1920*1080");
</script>
```



```
<!doctype html>
...<html lang="en"> == $0
  <head>...</head>
  <body style>
    <img alt="上尾美羽" size="1920*1080">
    <div></div>
    <script>...</script>
    <!-- Code injected by live-server -->
    <script type="text/javascript">...</script>
  </body>
</html>
```