# DOM

## DOM

全称为Document Object Model，文档对象模型。

JS中通过DOM来对HTML文档进行操作。只要理解了DOM就可以随心所欲的操作WEB页面。

-文档

文档表示的就是整个的HTML网页文档

-对象

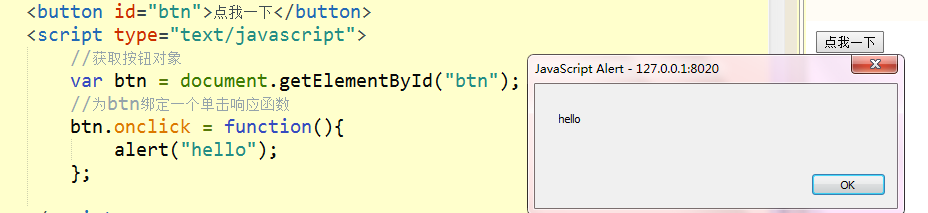
对象表示将网页中的每一个部分都转换为了一个对象。

-模型

使用模型来表示对象之间的关系，这样方便我们获取对象。

## 事件

* 事件，就是文档或浏览器窗口中发生的一些特定的交互瞬间。
* JavaScript 与 HTML 之间的交互是通过事件实现的。
* 对于 Web 应用来说，有下面这些代表性的事件：点击某个元素、将鼠标移动至某个元素上方、按下键盘上某个键，等等。
* 可以为元素的指定的事件属性绑定回调函数的形式来响应事件,该函数会在指定事件触发时才执行



## 文档加载

浏览器在加载页面时，是按照自上向下的顺序进行加载的，当它加载到JS代码时，会将JS代码立即执行， 如果将JS代码编写到网页的上边，会导致JS执行时页面还没有完全加载，将会导致无法正常获取到DOM对象

将JS代码编写到body标签里的下边，可以确保代码在页面加载完毕才执行，这样可以保证正常获取到DOM对象

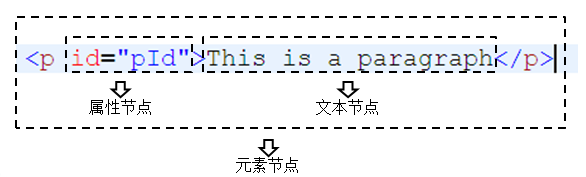
如果想将JS代码编写到网页的上边，需要使用window的onload事件，会在页面加载完毕之后触发，它所对应的回调函数，会在页面加载完毕之后执行

\* 以后写JS代码时，如果需要将JS代码编写到网页是上边，上来就把window.onload = function(){}写上

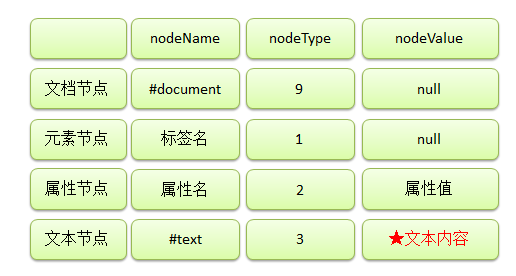
但是写到网页上边，先加载却不执行，既然不执行，这么早加载没有必要；不如写到body标签里的下边，页面加载完了JS代码才加载，可以确保用户最先看到的是页面，JS代码随用随加载，追求性能极致就写下面，但对于现在的网速，两者基本差不多，写网页上边，方便管理，但写上边一定要写window.onload

## 节点Node

* 节点Node，是构成我们网页的最基本的组成部分，网页中的每一个部分都可以称为是一个节点。
* 常用节点分为四类
  + **文档节点**：整个HTML文档
  + **元素节点**：HTML文档中的HTML标签
  + **属性节点**：元素的属性
  + **文本节点**：HTML标签中的文本内容
* 虽然都是节点，但是实际上他们的具体类型是不同的。
* 比如：标签我们称为元素节点、属性称为属性节点、文本称为文本节点、文档称为文档节点。
* 节点的类型不同，属性和方法也都不尽相同。

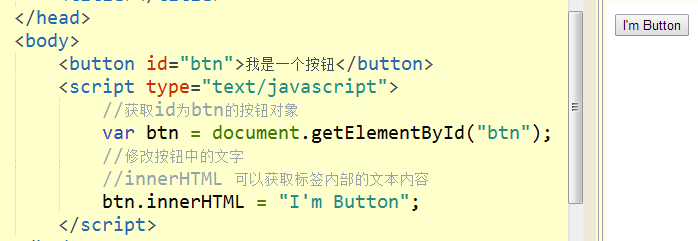


### 节点的属性



### 文档节点document

* 文档节点document，代表的是整个HTML文档，网页中的所有节点都是它的子节点。
* document对象作为window对象的属性存在的，我们不用获取可以直接使用。
* 通过该对象我们可以在整个文档访问内查找节点对象，并可以通过该对象创建各种节点对象。



### 元素节点Element

* HTML中的各种标签都是元素节点，这也是我们最常用的一个节点。
* 浏览器会将页面中所有的标签都转换为一个元素节点，我们可以通过document的方法来获取元素节点。
* 比如：
  + document.getElementById()

根据id属性值获取一个元素节点对象

### 文本节点Text

* 文本节点表示的是HTML标签以外的文本内容，任意非HTML标签的文本都是文本节点。
* 它包括可以字面解释的纯文本内容。
* 文本节点一般是作为元素节点的子节点存在的。
* 获取文本节点时，一般先要获取元素节点。再通过元素节点获取文本节点。
* 例如：
  + 元素节点.firstChild;
  + 获取元素节点的第一个子节点，一般为文本节点

### 属性节点Attr

* 属性节点表示的是标签中的一个一个的属性，这里要注意的是属性节点并非是元素节点的子节点，而是元素节点的一部分。
* 可以通过元素节点来获取指定的属性节点。
* 例如：
  + 元素节点.getAttributeNode("属性名");
* 注意：我们一般不使用属性节点。

**以下获取节点具体实例请参照day13中07.dom查询网页**

定义一个函数，专门用来为指定id的元素绑定单击响应函数

\* 参数：

\* idStr 要绑定的元素的id属性值

\* fun 要绑定的单击响应函数，该函数在元素的单击事件触发时执行

function myClick(idStr , fun){}

## 查询节点

### 获取元素节点

通过document对象调用

getElementById() 通过id属性获取一个元素节点对象

getElementsByTagName() 通过标签名获取一组元素节点对象，返回的是类数组对象，即使一个也返回类数组

getElementsByName() 通过name属性获取一组元素节点对象，返回的是类数组对象，即使一个也返回类数组

### 获取元素节点的子节点

* 通过具体的元素节点调用

1. getElementsByTagName()
   * 方法，返回当前节点的指定标签名后代节点
2. childNodes
   * 属性，表示当前节点的所有子节点
   * 根据DOM标准，空白的内容也会被当成子节点
   * 注意在IE8及以下的浏览器中没有将空白当成子节点
   * 所以IE8中childNodes不会返回空白的节点

children可以用来获取当前元素的所有子元素（不包括文本）推荐使用

1. firstChild
   * 属性，表示当前节点的第一个子节点（包括文本）
2. lastChild
   * 属性，表示当前节点的最后一个子节点（包括文本）
3. firstElementChild
   * 属性，表示当前节点的第一个子元素（不包括文本）
   * 不支持IE8及以下浏览器
4. lastElementChild
   * 属性，表示当前节点的最后一个子元素（不包括文本）
   * 不支持IE8及以下浏览器

### 获取父节点和兄弟节点

* 通过具体的节点调用

1. parentNode
   * 属性，表示当前节点的父节点，可以叠加
2. previousSibling
   * 属性，表示当前节点的前一个兄弟节点（包括文本）
3. nextSibling
   * 属性，表示当前节点的后一个兄弟节点（包括文本）

### 元素节点的属性

* 获取，元素对象.属性名

例： element.value

element.id

element.className

由于class是JS中的保留字，所以不能直接使用.class来读取class属性值，而需要使用className代替

* 设置，元素对象.属性名=新的值

例： element.value = “hello”

element.id = “id01”

element.className = “newClass”

### 其他属性

* nodeValue
  + 文本节点可以通过nodeValue属性获取和设置文本节点的内容
* innerHTML
  + 元素节点通过该属性获取和设置标签内部的html代码
  + innerHTML不能用于自结束标签
* innerText
  + 可以用来获取标签内部的文本内容
  + 它会自动去除HTML标签

innerHTML和innerText只是读取的时候有区别，赋值的时候没区别

### 查询的其他方法

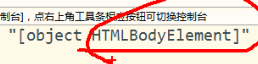
#### 获取body标签

//var body = document.getElementsByTagName("body")[0];

使用getElementsByTagName返回的是一个数组，但body只有一个，所以后面加一个索引，来使其就返回body，一般不使用这个方法

**document.body**可以用来获取页面中的body元素

var body = document.body;



#### 获取跟标签

**document.documentElement**可以获取页面的根元素

var root=document.documentElement



#### 获取页面中的所有元素

var all = document.getElementsByTagName("\*");

另一种方法：**document.all**，但var all = document.all输出为undefined，typeof也是undefined，但可以正常使用，有值，只是伪装起来了，可以调length方法，也可以看索引。

#### class属性值

**document.getElementsByClassName()**

\* - 可以根据class属性值获取一组元素节点对象，

\* 但是该方法不支持IE8及以下的浏览器

#### CSS选择器

**querySelector()**

\* - 该方法可以根据CSS选择器去页面中查找对象

\* - 需要一个选择器的字符串作为参数

\* - querySelector()只会返回一个符合条件的元素

\* 如果符合条件的元素有多个，它只会返回第一个

var box2 = document.querySelector("#box2");

**querySelectorAll()**

\* - 和querySelector()功能类似，都可以根据CSS选择器去查找

\* - 不同的是querySelectorAll()它会将所有符合条件的元素封装到一个数组中返回，即使查询到的只有一个，也会返回数组

var box1 = document.querySelectorAll(".box1");

## 增删改节点

### createElement()

\* - 可以根据标签名来创建一个指定的元素节点对象并返回

var div=document.createElement(“div”); //创建div标签

### createTextNode()

\* - 可以根据文本内容创建一个文本节点对象并返回

var text=document.createTextNode(“文本”); //创建文本节点对象

### appendChild()

\* - 可以向父节点中添加新的子节点

\* - 语法：父节点.appendChild(子节点)

div.appendChild(text); //<div>文本</div>

### insertBefore()

\* - 可以在指定的子节点前插入新的子节点

\* - 这个方法需要由父节点调用

\* - 语法：父节点.insertBefore(新的节点,旧的节点)，如果不知道父节点，可以使用 旧的节点.parentNode 来代替父节点

### replaceChild()

\* - 可以使用指定的子节点替换已有的子节点

\* - 语法：父节点.replaceChild(新的节点,旧的节点); 如果不知道父节点，可以使用 旧的节点.parentNode 来代替父节点

### removeChild()

\* - 用于删除一个子节点

\* - 语法：父节点.removeChild(子节点); 如果不知道父节点，可以使用 子节点.parentNode 来代替父节点

## confirm()

\* - 可以用来弹出一个带有确认和取消按钮的提示框

\* - 如果用户点击确认则返回true

\* 如果点击取消则返回false

## 取消超链接默认行为

点击超链接以后，它会立即跳转页面，这个是超链接的默认行为，如果要取消元素的默认行为，可以在元素响应函数的最后 return false，或者给超链接的href设置为"javascript:;"

# this补充

this是函数的上下文对象，根据函数的调用方式不同，this的值也不同

1.以函数的形式调用，this是window

2.以方法的形式调用，this是调用方法的对象

3.以构造函数的形式调用，this是新创建的那个对象

4.以call()和apply()调用，this是方法的第一个参数

5.在事件的响应函数中，事件是为谁绑定的this就是谁