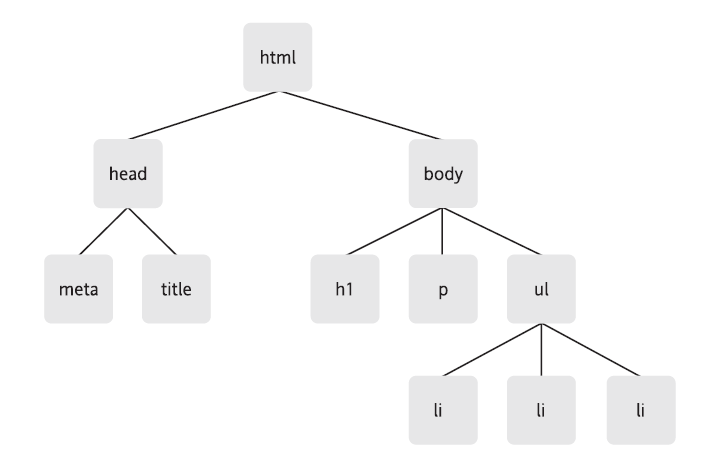
# DOM

1. **DOM：**document object model，文本对象模型，是针对xml、html的基于树的API。将文档表示为一棵树，定义了Node的接口，以及许多节点类型来表示xml节点。
2. **DOM树：**节点node的层次



1. **节点Node：**

<p title=”hello”>.</p>

1. **JavaScript代码放在哪个位置：**

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  
<html>  
<head>  
 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
 <title>Insert title here</title>

**<!-- ====== javascript代码位置 ======== -->**  
 <script type="text/javascript">  
  **//window.onload事件在整个HTNL文档被加载完后，才执行  
 //因此，在这里，可以获取整个文档的任何元素**  
  **window.onload = function()**{  
  **//1. 获取 button**  
 var buttons = document.getElementsByTagName("button");  
  **//2. 为 button 添加 onclick 响应函数**  
 buttons[0].onclick = function(){  
 alert("helloworld!! \nbuttons.length=" + buttons.length);  
 }  
 }  
 </script>  
</head>  
<body>  
 <button>ClickMe</button>  
</body>  
</html>

1. **节点：**

**操作节点方式：先用getElementById，根据属性id的值，获取节点的位置，在操作其他属性、子节点值，eg：读写节点**

* **节点分类：**
* **元素节点（标签）**:<html>、<head>、<body>…
* **属性节点:**元素节点包含的属性，用于描述元素节点
* **文本节点：**包含在元素节点中，nodeValue在文本节点用的多，eg：元素节点的子节点，内容为文本
* **节点属性：**任何节点都有
* **nodeType：节点类型**
* **nodeName：节点名**
* **nodeValue：节点值**
* **获取节点**
* document.**getElementById（常用）**: 根据 id 属性获取对应的单个节点
* document.getElementsByTagName: 根据标签名获取指定节点名字的数组, 数组对象 length 属性可以获取数组的长度
* document.getElementsByName: 根据节点的 name 属性获取符合条件的节点数组,
* **读写属性节点**
* **子节点：**

只有元素节点才有子节点

* **示例代码：**

**<!-- "!DOCTYPE" 声明不是一个 HTML 标签；它是用来告知 Web 浏览器页面使用了哪种 HTML 版本-->**  
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
 <script>  
 window.onload = function(){  
  **//元素节点: li**  
 var bjNode = document.getElementById("bj");  
 alert("element node: \n" + bjNode.nodeType + "--" + bjNode.nodeName + "--" + bjNode.nodeValue);  
 **//属性节点: name输入框中的input，使用id进行定位：name**  
 var nameNode = document.getElementById("name");  
  **//文本节点：元素节点的第一个子节点**  
 var textNode = bjNode.firstChild;

**//修改节点内容：第一个子节点内容“北京”改为“厦门”**

var bjFirstNode = document.getElementById("bj").firstChild;

bjFirstNode.nodeValue = "厦门";

}  
 </script>  
</head>

<body>  
<p>城市列表：</p>  
 <ul **id**="city"> **<!-- 父节点 -->**

**<!-- 子节点：子节点的第一个节点是：北京-->**  
 <li **id**="bj" name="BeiJing">北京</li>  <li>上海</li>  
 <li>东京</li>  
 <li>西雅图</li>  
 </ul>  
  
 <br><br>  **<!-- 这是一个文本节点：含有三种属性：使用id进行定位 -->**  
 name:<input type="text" name="username" **id**="name" value="Chris">  
</body>  
</html>

1. **常用：**

* **<a>**: 定义超链接
* **herf**: hypertext teference，引用超文本连接，URL
* **<table>**:表格
* **<p>**:段落
* **<br>**：换行
* **&nbsp**：英文的空格
* **<body>**：文档主题，显示内容的代码块
* **<!DOCTYPE>**： 声明位于文档中的最前面的位置，处于 <html> 标签之前。

声明不是一个 HTML 标签；它是用来告知 Web 浏览器页面使用了哪种 HTML 版本。

* **<li>**:列表项目，用在下面的表单中

**<ol>**:有序表

**<ul>**:无序表

**<menu>**: 菜单表

**示例代码：**

<ol>  
  <li>Coffee</li>  
</ol>  
<ul>  
  <li>Coffee</li>  
</ul>

* **<div>**: 定义 HTML 文档中的一个分隔区块或者一个区域部分。

常用于组合块级元素，以便通过 CSS 来对这些元素进行格式化。

* **<select>**：下拉表单，配合<option>使用
* **<textarea>**：文本区
* **<input>**：选择type=“”，可以改变输入方式，eg：checkbox勾选、radio原点选择…
* **<span>:** 用于对文档中的行内元素进行组合。<span> 标签没有固定的格式表现。当对它应用样式时，它才会产生视觉上的变化。如果不对 <span> 应用样式，那么 <span> 元素中的文本与其他文本不会任何视觉上的差异。

**例如：文本着色**

<p>我的母亲有 <span style="color:blue">蓝色</span> 的眼睛。</p>

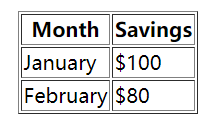


* **<h1>---<6>**：定义html的6个标题，字体依次减小
* **<b>**粗体</b>
* **<table** border=”1”></table>：表格
* **<tr>**</tr>：表格中的一行
* **<td>**</td>：标准单元格，文字左对齐、普通字体
* **<th>**</th>：表头单元格，文字粗体、居中

**常见组合：**

<table border="1">  
<tr>  
 <th>Month</th>  
 <th>Savings</th>  
</tr>  
<tr>  
 <td>January</td>  
 <td>$100</td>  
</tr>

</table>



* **<hr>**：横线

1. **鼠标点击内容，调用自定义函数，示例代码：**

**window.onload = function()**{

**//自定义互换两个节点的函数**

function **replaceEach**(aNode, bNode){ }

**//1. 获取所有的 li 节点**

var liNodes = document.**getElementsByTagName**("li");

**//2. 为每一个 li 节点添加 Onclick 响应函数**

for(var i = 0; i < liNodes.length; i++){

//手动为每个 li 节点添加一个 index 属性

liNodes[i].index = i;

//鼠标点击内容后，调用自定义函数

liNodes[i].**onclick** = **function()**{

**replaceEach**(this, liNodes[targetIndex]);

}

}

}