# webApi第三天

1. 创建元素的方式

补充一点：script你可以写在页面任何位置 如果你写在head里面 它会自定放到head里面，如果你写在body里面它会在body里面，如果你写在body外面 head外面 ，它会自定帮你把script放在</body>前面

我们能否通过js去往页面上生成标签元素呢？

* 1. document.write()

优点：操作简便 可以识别标签

缺点: 无法精确某一个元素里面添加内容

在事件里面写该方法的时候 它会把之前的内容全部清除（原因是因为它会开启一个新的文档流）

一般不推荐使用这个方法

b) 可以是body或者某一个元素.innerHTML /innerText

这个方法可以获取或者是设置

缺点：如果大量做循环往里面添加内容 性能上很差

可以优化一下：使用字符串拼接，循环拼接完之后，再一次性写入页面 提高性能

c) document.createElement(你创建的标签) 在文档中创建元素

需要使用appendChild()往页面上添加

在性能上较好，和优化后的innerHTML差不多

缺点：操作繁琐

总结： 如果大量做字符串拼接 使用innerHTML 优化一下 推荐 （ajax数据返回会做大量拼接生成 一般使用innerHTML）

如果要是生成内容数量较少 建议使用document.createElement();

1. 获取和设置样式的函数封装
   1. Js中直接操作样式 操作d（读取和设置）的是行间的样式

无法获取样式表中的样式

我们实际开发的时候一般是样式和html结构分离

b) 标准浏览器下： getComputedStyle(obj元素).属性

在IE内核的浏览器下：obj元素.currentStyle.属性

function getStyle(obj,attr) {

if(obj.currentStyle) {

return oBox.currentStyle[attr];

} else {

return getComputedStyle(obj)[attr];

}

}

· c) 几个属性的说明

Display:none; 不占位置 页面会重排

Visibility:hidden; 占位置

Opacity:0

1. DOM节点知识：
   1. 节点的概念：
      1. 节点：我们可以把页面中的所有的标签（元素）以及属性 文本还有注释等等都可以称之为节点
      2. 整个文档都是由节点来构成的，节点和节点之间有关系，我们可以通过这些关系去做节点的获取以及修改等操作，非常快捷和方便 提高开发效率
      3. 整个页面的节点结构 可以看成一个树状所以说我们称之为DOM树结构
   2. 节点的类型：
      1. 在DOM里面有12中节点类型 DOM（xml（可扩展标记语言）和html（超文本标记语言））
         1. Xml （一般用来做前后端数据交互的格式 老式的格式）新式格式json
         2. Var preson = {

Name:”尼古拉斯赵四”,

Age:25

}

* + - 1. Demo.xml

<person>

<name>尼古拉斯赵四</name>

<age>25</age>

</person>

节点类型：

dom 所有的节点类型 节点类型nodeType

元素节点 　　Node.ELEMENT\_NODE(1) 1

属性节点 　　Node.ATTRIBUTE\_NODE(2) 2

文本节点 　　Node.TEXT\_NODE(3) 3

CDATA节点 Node.CDATA\_SECTION\_NODE(4)

实体引用名称节点 　　 Node.ENTRY\_REFERENCE\_NODE(5)

实体名称节点 　　Node.ENTITY\_NODE(6)

处理指令节点 　Node.PROCESSING\_INSTRUCTION\_NODE(7)

注释节点 　 Node.COMMENT\_NODE(8) 8

文档节点 　 Node.DOCUMENT\_NODE(9) 文档类型节点 　　Node.DOCUMENT\_TYPE\_NODE(10) 10

文档片段节点 　　Node.DOCUMENT\_FRAGMENT\_NODE(11) DTD声明节点 Node.NOTATION\_NODE(12)-->

我们可以查看html页面中的所有的节点的类型 以及名称 以及内容

nodeType nodeName nodeValue

元素(标签)节点　　 Node.ELEMENT\_NODE(1) 1

属性节点　　 Node.ATTRIBUTE\_NODE(2) 2

文本节点　　 Node.TEXT\_NODE(3) 3