爱创课堂前端培训

JavaScript进阶第一天

目录

**[一、 JavaScript](#_Toc13120_WPSOffice_Level1)** **[3](#_Toc13120_WPSOffice_Level1)**

**[二、 DOM](#_Toc5741_WPSOffice_Level1)** **[3](#_Toc5741_WPSOffice_Level1)**

[2.1 dom的组成](#_Toc2043_WPSOffice_Level2) [3](#_Toc2043_WPSOffice_Level2)

[2.2 节点类型](#_Toc22853_WPSOffice_Level2) [4](#_Toc22853_WPSOffice_Level2)

[2.3 childNodes属性](#_Toc19962_WPSOffice_Level2) [4](#_Toc19962_WPSOffice_Level2)

[2.4 写轮子统一浏览器表现](#_Toc26317_WPSOffice_Level2) [5](#_Toc26317_WPSOffice_Level2)

[2.5 节点属性](#_Toc25964_WPSOffice_Level2) [5](#_Toc25964_WPSOffice_Level2)

[2.6 节点关系](#_Toc17528_WPSOffice_Level2) [5](#_Toc17528_WPSOffice_Level2)

**[三、 元素操作](#_Toc13323_WPSOffice_Level1)** **[6](#_Toc13323_WPSOffice_Level1)**

[3.1 创建元素](#_Toc9950_WPSOffice_Level2) [6](#_Toc9950_WPSOffice_Level2)

[3.2 上树](#_Toc4782_WPSOffice_Level2) [6](#_Toc4782_WPSOffice_Level2)

[3.3 下树](#_Toc10135_WPSOffice_Level2) [6](#_Toc10135_WPSOffice_Level2)

[3.4 替换](#_Toc12352_WPSOffice_Level2) [7](#_Toc12352_WPSOffice_Level2)

[3.5 克隆](#_Toc3784_WPSOffice_Level2) [7](#_Toc3784_WPSOffice_Level2)

**[四、 jQuery中的元素操作](#_Toc23956_WPSOffice_Level1)** **[7](#_Toc23956_WPSOffice_Level1)**

[4.1 创建元素](#_Toc30262_WPSOffice_Level2) [7](#_Toc30262_WPSOffice_Level2)

[4.2 上树](#_Toc4269_WPSOffice_Level2) [8](#_Toc4269_WPSOffice_Level2)

[4.3 wrap](#_Toc31200_WPSOffice_Level2) [8](#_Toc31200_WPSOffice_Level2)

[4.4 unwrap](#_Toc9076_WPSOffice_Level2) [9](#_Toc9076_WPSOffice_Level2)

**[五、 animate函数](#_Toc2662_WPSOffice_Level1)** **[9](#_Toc2662_WPSOffice_Level1)**

# JavaScript

JavaScript这门语言由 DOM、 BOM、ECMAScript组成

DOM：document object model 文档对象模型 体现在代码里就是document对象。

BOM：browser object model 浏览器对象模型 体现在代码中就是window对象。

ECMAScript： 核心语法。

包含如何定义变量、 循环语句、运算符、表达式、流程控制语句、数据类型。

JS基础我们讲的是核心语法部分。JS进阶讲剩余的两部分。

# DOM

我们可以使用dom的getElementById获取元素

我们可以使用dom的onclick、onmousedown、onmouseup进行事件的设置

我们可以使用dom的style进行样式的设置。

等等……

## 2.1 dom的组成

Dom是由节点组成的。

Dom节点一共有12种。 元素/标签仅仅是其中的一种。

我们需要记住的有四种：

1 元素类型

3 文本类型

8 注释类型

9 文档类型

全部节点类型：

元素节点 　　Node.ELEMENT\_NODE(1)

属性节点 　　Node.ATTRIBUTE\_NODE(2)

文本节点 　　Node.TEXT\_NODE(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)

文档节点 　 Node.DOCUMENT\_NODE(9)

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

文档片段节点 　　Node.DOCUMENT\_FRAGMENT\_NODE(11)

DTD声明节点 Node.NOTATION\_NODE(12)

## 2.2 节点类型

每一个元素都是节点，但是每一个节点不一定是元素。 元素只是节点的一个分类。

**元素.nodeType**

|  |
| --- |
| 1. // 获取元素 2. var box = document.getElementById("box"); 3. console.log(box.nodeType); // 这里输出的是1 是元素类型 |

**文本.nodeType**

|  |
| --- |
|  |

**注释.nodeType**

|  |
| --- |
|  |

**文档.nodeType**

|  |
| --- |
|  |

## 2.3 childNodes属性

它指向调用该属性的节点的所有子节点集合。

Dom结构：

|  |
| --- |
| 1. <div id="box"> 2. <div id="box1"></div> 3. nihao 4. <!-- 你好帅 --> 5. </div> |

输出结果：

|  |
| --- |
|  |

经过观察，我们发现明明在书写dom结构的时候只有3个节点。但是却在输出的时候输出了5个节点。

这是因为高级浏览器会将空白折叠现象形成的空白符也当做一个文本类型的节点。

再观察IE的输出情况：

因为IE的控制台不能点开列表。 所以输出的是childNodes.length属性

|  |
| --- |
|  |

所以，此时IE反而比高级浏览器好用一点。

## 2.4 写轮子统一浏览器表现

|  |
| --- |
| 1. function getChildNodes(dom) { 2. // 定义一个数组 3. var arr = []; 4. // 先输出所有的节点的类型 如果是3 查看是不是空文本 5. for(var i = 0; i < dom.childNodes.length; i++) { 6. // 判断当前节点的类型是不是文本 7. if(dom.childNodes[i].nodeType === 3) { 8. // 使用正则表达式匹配dom.childNodes[i].data 是不是空白符 9. // 定义正则表达式 10. var reg = /^\s+$/; 11. if(reg.test(dom.childNodes[i].data)) { 12. // 既然是空白符 就不要进入了 13. } else { 14. // 如果是一个文本并且内容不是空的 也要进入数组 15. arr.push(dom.childNodes[i]); 16. } 17. } else { 18. // 如果不是文本 直接进入数组 19. arr.push(dom.childNodes[i]); 20. } 21. } 22. // 循环体内主要是在检查哪一项是空白 23. return arr; 24. } |

## 2.5 节点属性

**nodeName** **nodeValue**

|  |
| --- |
| 1. 元素类型的节点的nodeName 是该元素的标签名大写字符串 2. 文本类型的节点的nodeName 永远是#text字符串 3. 注释类型的节点的nodeName 永远是#comment 4. 元素类型的节点的nodeValue 永远是null 5. 文本类型的节点的nodeValue 就是文本的书写内容 6. 注释类型的节点的nodeValue 就是注释内容 |

文档类型的nodeName是#document nodeValue是null

## 2.6 节点关系

节点关系分为以下几种：

父子、 子父、 兄弟

父子：

|  |
| --- |
| 1. parent.childNodes[index]; // 第index个子元素 2. parent.firstChild; // 第一个子节点 3. parent.lastChild; // 最后一个子节点 |

子父：

|  |
| --- |
| 1. child.parentNode; |

兄弟：

|  |
| --- |
| 1. node.previousSibling; // 前一个兄弟 2. node.nextSibling; // 下一个兄弟 |

# 元素操作

## 创建元素

document.createElement(type);

type: 元素类型（标签名）字符串

返回值： 被创建出来的元素 天生是一个孤儿节点

EX：

|  |
| --- |
| 1. var p = document.createElement("p"); 2. console.log(p); |



## 上树

father.appendChild(child);

child: 被追加的子节点

father: 父节点

最终效果： 让child到father的childNodes中作为最后一个子元素存在

返回值： child

## 下树

father.removeChild(child);

child:被移除的子节点

father: 父节点

最终效果： 从father中移除child

返回值： child

# 下午复习：

节点： 一共分12类。

1 元素

3 文本

8 注释

9 文档

节点属性：

nodeType： 节点的属性类型值

nodeName： 节点的名称

nodeValue： 节点的值

节点关系：

childNodes: 子节点的集合。 在IE下，表现正常，在高级浏览器中，会将空白折叠现象形成的空白符当做文本节点。

父子：

firstChild： 第一个子节点

lastChild： 最后一个子节点

子父：

parentNode： 父节点

兄弟节点：

nextSibling: 下一个兄弟

previousSibling： 前一个兄弟

节点操作：

创建元素：

document.createElement(type)

上树：

father.appendChild(child)

下树：

removeChild(child)

插入：

insertBefore(newChild, oldChild);

## 替换

father.replaceChild(newChild, oldChild)

newChild: 新替换上的元素

oldChild: 被替换下的元素

返回值： oldChild

## 克隆

node.cloneNode(boolean);

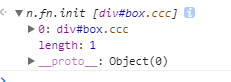
boolean: 是一个布尔值，如果为false，则只复制自身元素 如果为true则连子元素一起复制。

# jQuery中的元素操作

## 4.1 创建元素

使用$函数的功能创建元素

|  |
| --- |
| 1. $("<div id='box' class='ccc' data-info='nihao'>content</div>") |



## 4.2 上树

jQuery中的上树分很多种，可以子元素选择父元素 可以父元素选择子元素 可以兄弟选择兄弟

子元素选择父元素：

appendTo(selector|jQuery|element) // 往父元素的后面追加

prependTo(selector|jQuery|element) // 往父元素的前面追加

父元素选择子元素:

append(selector|jQuery|element|html); // 往父元素的后面追加

prepend(selector|jQuery|element|html); // 往父元素的前面追加

兄弟之间插入:

$("#box").after(selector | jQuery | element | function); 往#box后面放元素

$("#box").insertAfter(selector | jQuery | element); 将#box往后面参数据顶的元素后面放

$("#box").before(selector | jQuery | element | function); 往#box前面放元素

$("#box").insertBefore(selector | jQuery | element ); 将#box往后面参数决定的元素前面放

## 4.3 wrap

Dom结构：

|  |
| --- |
| 1. <p>A</p> 2. <p>B</p> 3. <p>C</p> 4. <p>D</p> 5. <p>E</p> |

JS代码：

|  |
| --- |
| 1. // wrap 让某一个元素外层多一层 2. $("p").wrap("<div></div>") |

结果：

|  |
| --- |
|  |

结论：

将参数转换成元素，并包裹住选中的元素

## 4.5 wrapAll

Dom结构：

|  |
| --- |
| 1. <p>A</p> 2. <p>B</p> 3. <p>C</p> 4. <p>D</p> 5. <p>E</p> |

JS代码：

|  |
| --- |
| 1. // wrapAll 将所有匹配的元素外层多一层元素 2. $("p").wrapAll("<div></div>") |

结果：

|  |
| --- |
|  |

结论：

将参数转换成元素，并包裹住所有选中的元素

## 4.4 unwrap

Dom结构：

|  |
| --- |
| 1. <div class="box"> 2. <h3 id="h3">我是h3</h3> 3. </div> |

JS代码：

|  |
| --- |
| 1. // unwrap 给选中的元素去掉一层元素 2. // $("#h3").unwrap(); |

结果：

|  |
| --- |
|  |

结论：

将所选择的的元素的父元素去掉

## 4.5 replaceWith()和replaceAll()方法

|  |
| --- |
| <p>A</p>  <p>B</p>  <p>C</p>  <p>D</p>  <p>E</p> |

执行语句：

|  |
| --- |
| $("<div></div>").replaceAll("p"); |

或者

|  |
| --- |
| $("p").replaceWith("<div></div>"); |

都将变为：

|  |
| --- |
| <div></div>  <div></div>  <div></div>  <div></div>  <div></div> |

## 4.6 empty()和remove()方法

empty()清空自己，自己还在。

remove()表示自杀，自己和后代都没了。

## 4.7 clone()方法

|  |
| --- |
| $("#box").clone() |

不用写参数，就表示所有的后代被复制。天生会复制后代。

比如：

|  |
| --- |
| <div id="box">  <ul>  <li>哈哈</li>  <li>哈哈</li>  <li>哈哈</li>  <li>哈哈</li>  </ul>  </div> |

jQuery：

|  |
| --- |
| $("#box").after($("#box").clone()); |

变为：

|  |
| --- |
| <div id="box">  <ul>  <li>哈哈</li>  <li>哈哈</li>  <li>哈哈</li>  <li>哈哈</li>  </ul>  </div>  <div id="box">  <ul>  <li>哈哈</li>  <li>哈哈</li>  <li>哈哈</li>  <li>哈哈</li>  </ul>  </div> |

如果加上true，就表示事件监听也被复制了。

# animate函数

思路：

1获取目标值

2再获取初始值

3得到总距离

4定义定时器的执行间隔

5获取时间

6得到总次数

7总距离/总次数 = 步长

8使用setInterval不停地改变dom元素的每一个css值 让元素不停地改变样式 造成动画的错觉。

9定义一个计数器 每改变一次样式就累加一次 直到计数器大于等于总次数 此时说明执行完毕

10强行拉到终点

11停表

12执行回调函数

代码：

|  |
| --- |
| 1. /\* 2. \* animate 函数能够实现动画 3. \* @dom 要运动的元素 4. \* @json css样式对象 5. \* @time 时间 以毫秒值为单位 6. \* callback 回调函数 7. \*/ 8. function animate(dom, json, time, callback) { 9. // 定义定时器的间隔 10. var interval = 20; 11. // 定义总次数 12. var allCount = time / interval; 13. // 获取初始值 14. // 因为不确定json中有多少条css样式 所以不能写具体的代码条数 15. // 使用对应的思想 所以 我们也定义一个json 16. var nowJSON = {}; 17. // 使用for循环获取初始值 18. for(var i in json) { 19. // 强制性的给nowJSON添加属性 并赋值 20. nowJSON[i] = parseInt(getComputedStyle(dom)[i]); 21. } 22. // 定义步长json 23. var stepJSON = {}; 24. for(var i in json) { 25. stepJSON[i] = (json[i] - nowJSON[i]) / allCount; 26. } 27. // 定义计数器 28. var count = 0; 29. var timer = setInterval(function() { 30. count++; 31. // 改变dom元素的css样式 32. for(var i in json) { 33. dom.style[i] = nowJSON[i] + stepJSON[i] \* count + "px"; 34. } 35. // 判断是否执行完毕 36. if(count >= allCount) { 37. // 停表 38. clearInterval(timer); 39. // 拉终 40. for(var i in json) { 41. dom.style[i] = json[i] + "px"; 42. } 43. // 回调函数执行 44. callback && callback(); 45. } 46. }, interval); 47. } |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |