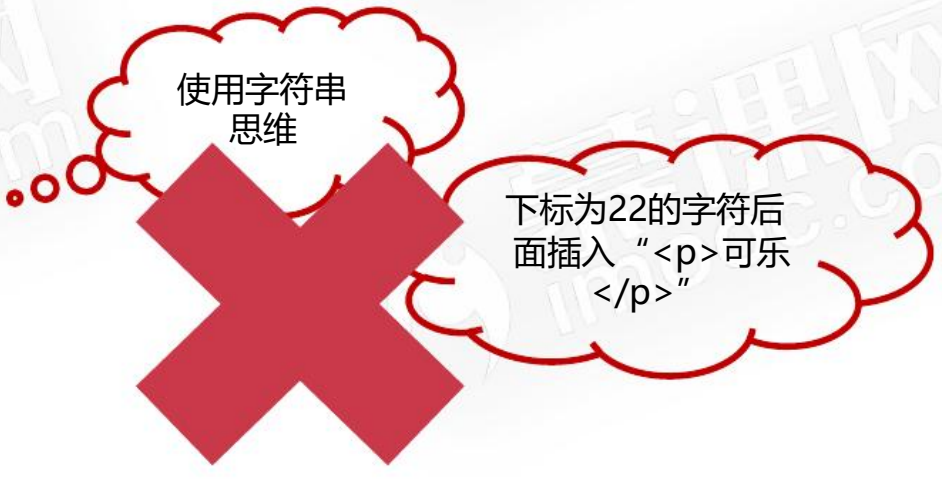
DOM基本概念

DOM是JS操控HTML和CSS的桥梁

DOM使JS操作HTML变得优雅

- 比如下面的HTML结构，现在想用JavaScript在“牛奶”后面插入一个p标签对，内容是“可乐”

```
<div id="box">
    <p>牛奶</p>
    <p>咖啡</p>
    <p>果汁</p>
</div>
```



DOM使JS操作HTML变得优雅

- 比如下面的HTML结构，现在想用JavaScript在“牛奶”后面插入一个p标签对，内容是“可乐”

```
<div id="box">
    <p>牛奶</p>
    <p>咖啡</p>
    <p>果汁</p>
</div>
```


DOM简介

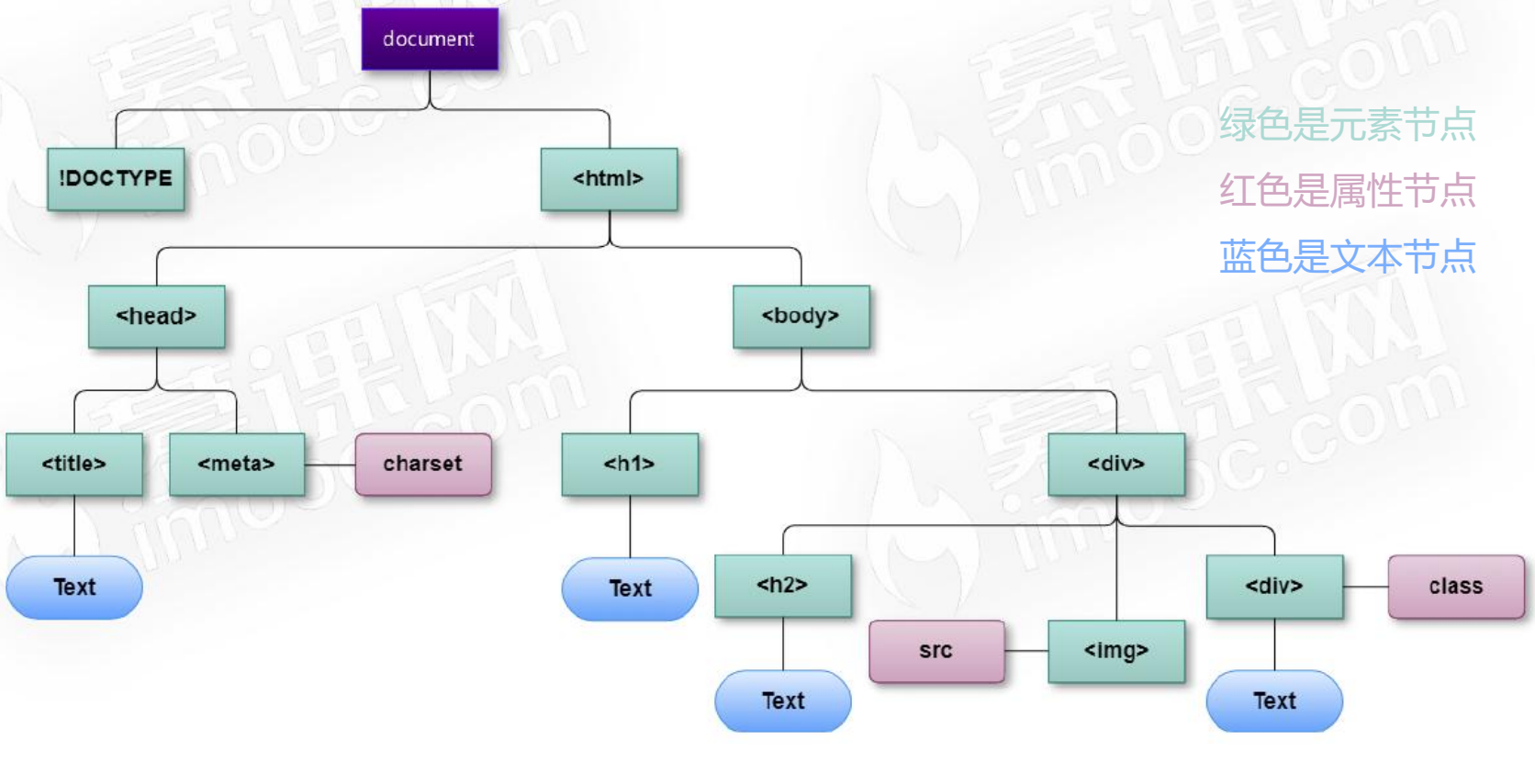
- DOM（Document Object Model，文档对象模型）是JavaScript操作HTML文档的接口，使文档操作变得非常优雅、简便

- DOM最大的特点就是将文档表示为节点树

DOM节点树

```
<!DOCTYPE  html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Document</title>
    </head>
    <body>
        <h1>慕课网</h1>
        <div>
            <h2>程序员的梦工厂 </h2>
            <img src="logo.png">
            <div class="box">
                慕课专栏
            </div>
        </div>
    </body>
</html>
```

DOM节点树



绿色是元素节点

红色是属性节点

蓝色是文本节点

nodeType常用属性值

- 节点的nodeType属性可以显示这个节点具体的类型

nodeType值	节点类型
1	元素节点，例如 <p> 和
3	文字节点
8	注释节点
9	document节点
10	DTD节点

访问元素节点

访问元素节点

- 所谓“访问”元素节点，就是指“得到”、“获取”页面上的元素节点
- 对节点进行操作，第一步就是要得到它
- 访问元素节点主要依靠document对象

认识document对象

- document对象是DOM中最重要的东西，几乎所有DOM的功能都封装在了document对象中
- document对象也表示整个HTML文档，它是DOM节点树的根
- document对象的nodeType属性值是9

访问元素节点的常用方法

方法	功能	兼容性
document.getElementById()	通过id得到元素	IE6
document.getElementsByTagName()	通过标签名得到元素数组	IE6
document.getElementsByClassName()	通过类名得到元素数组	IE9
document.querySelector()	通过选择器得到元素	IE8部分兼容、IE9完全兼容
document.querySelectorAll()	通过选择器得到元素数组	IE8部分兼容、IE9完全兼容

getElementById()

- document.getElementById()功能是通过id得到元素节点

HTML代码：

```
<div id="box">我是一个盒子</div>
<p id="para">我是一个段落</p>
```

JS代码：

```
var box = document.getElementById('box');
var para = document.getElementById('para');
```

参数就是元素节点的id，注意不要写#号

注意事项

- 如果页面上有相同id的元素，则只能得到第一个
- 不管元素藏的位置有多深，都能通过id把它找到

延迟运行

- 在测试DOM代码时，通常JS代码一定要写到HTML节点的后面，否则JS无法找到相应HTML节点

- 可以使用window.onload = function(){} 事件，使页面加载完毕后，再执行指定的代码

getElementsByTagName()

- getElementsByTagName()方法的功能是通过标签名得到节点数组

HTML代码：

```
<p>我是段落</p>
<p>我是段落</p>
<p>我是段落</p>
<p>我是段落</p>
```

JS代码：

```
var ps = document.getElementsByTagName('p');
```

数组

注意事项

- 数组方便遍历，从而可以批量操控元素节点
- 即使页面上只有一个指定标签名的节点，也将得到长度为1的数组
- 任何一个节点元素也可以调用getElementsByTagName()方法，从而得到其内部的某种类的元素节点

get ElementsB yClassName()

- getElementsByClassName()方法的功能是通过类名得到节点数组

HTML代码：

```
<div class="spec">我是盒子</div>
<div>我是盒子</div>
<div class="spec">我是盒子</div>
<div class="spec">我是盒子</div>
```

JS代码：

```
var spec_divs = document.getElementsByClassName('spec');
```

注意不要写.号

注意事项

- getElementsByClassName()方法从IE9开始兼容
- 某个节点元素也可以调用getElementsByClassName()方法，从而得到其内部的某类名的元素节点

querySelector()

- querySelector()方法的功能是通过选择器得到元素

HTML代码：

```
<div id="box1">
    <p>我是段落</p>
    <p class="spec">我是段落</p>
    <p>我是段落</p>
</div>
```

JS代码：

```
var the_p = document.querySelector('#box1 .spec');
```

选择器

注意事项

- querySelector()方法只能得到页面上一个元素，如果有多个元素符合条件，则只能得到第一个元素
- querySelector()方法从IE8开始兼容，但从IE9开始支持CSS3的选择器，如:nth-child()、:[src^='dog']等CSS3选择器形式都支持良好

querySelectorAll()

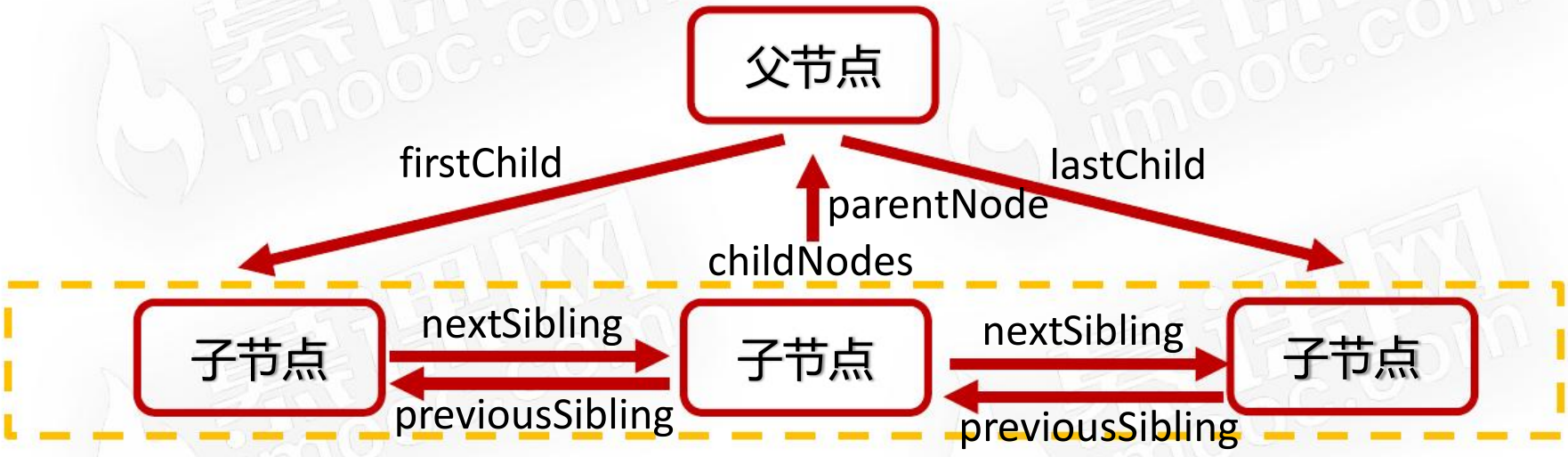
- querySelectorAll()方法的功能是通过选择器得到元素数组
- 即使页面上只有一个符合选择器的节点，也将得到长度为1的数组

访问元素节点的常用方法

方法	功能	兼容性
document.getElementById()	通过id得到元素	IE6
document.getElementsByTagName()	通过标签名得到元素数组	IE6
document.getElementsByClassName()	通过类名得到元素数组	IE9
document.querySelector()	通过选择器得到元素	IE8部分兼容、IE9完全兼容
document.querySelectorAll()	通过选择器得到元素数组	IE8部分兼容、IE9完全兼容

节点的关系

节点的关系



节点的关系

关系	考虑所有节点
子节点	childNodes
父节点	parentNode
第一个子节点	firstChild
最后一个子节点	lastChild
前一个兄弟节点	previousSibling
后一个兄弟节点	nextSibling

注意：文本节点也属于节点

- DOM中，文本节点也属于节点，在使用节点的关系时一定要注意

- 在标准的W3C规范中，空白文本节点也应该算作节点，但是在IE8及以前的浏览器中会有一定的兼容问题，它们不把空文本节点当做节点

排除文本节点的干扰

- 从IE9开始支持一些“只考虑元素节点”的属性

关系	考虑所有节点	只考虑元素节点
子节点	childNodes	children
父节点	parentNode	同
第一个子节点	firstChild	firstElementChild
最后一个子节点	lastChild	lastElementChild
前一个兄弟节点	previousSibling	previousElementSibling
后一个兄弟节点	nextSibling	nextElementSibling

书写常见的节点关系函数

- 书写IE6也能兼容的“寻找所有元素子节点”函数
- 书写IE6也能兼容的“寻找前一个元素兄弟节点”函数
- 如何编写函数，获得某元素的所有的兄弟节点?

节点操作

如何改变元素节点中的内容

- 改变元素节点中的内容可以使用两个相关属性：

  ① innerHTML ② innerText

- innerHTML属性能以HTML语法设置节点中的内容

- innerText属性只能以纯文本的形式设置节点中的内容

如何改变元素节点的CSS样式

- 改变元素节点的CSS样式需要使用这样的语句：

CSS属性要写成“驼峰”形式

```
oBox.style.backgroundColor = 'red';
oBox.style.backgroundImage = 'url(images/1.jpg)';
oBox.style.fontSize = '32px';
```

CSS属性值要设置成完整形式

注意写单位

如何改变元素节点的HTML属性

- 标准W3C属性，如src、href等等，只需要直接打点进行更改即可

```
oImg.src  = 'images/2.jpg';
```

如何改变元素节点的HTML属性

- 不符合W3C标准的属性，要使用setAttribute()和getAttribute()来设置、读取

```
oBox.setAttribute('data-n', 10);

var n = oBox.getAttribute('data-n');
alert(n);
```

节点的创建、移除和克隆

节点的创建

- document.createElement() 方法用于创建一个指定tag name的HTML元素

```
var oDiv  = document.createElement('div');
```

“孤儿节点”

- 新创建出的节点是“孤儿节点”，这意味着它并没有被挂载到DOM树上，我们无法看见它

- 必须继续使用appendChild()或insertBefore()方法将孤儿节点插入到DOM树上

appen dChil d()

- 任何已经在DOM树上的节点，都可以调用appendChild()方法，它可以将孤儿节点挂载到它的内部，成为它的最后一个子节点

```
父节点.appendChild(孤儿节点);
```

insertBefore()

- 任何已经在DOM树上的节点，都可以调用insertBefore()方法，它可以将孤儿节点挂载到它的内部，成为它的“标杆子节点”之前的节点

```
父节点.insertBefore(孤儿节点, 标杆节点);
```

小练习

- 请动态创建出一个20行12列的表格

- 请制作九九乘法表（可以在刚才代码基础上改动升级）

移动节点

- 如果将已经挂载到DOM树上的节点成为appendChild()或者insertBefore()的参数，这个节点将会被移动

```
新父节点.appendChild(已经有父亲的节点);
新父节点.insertBefore(已经有父亲的节点, 标杆子节点);
```

- 这意味着一个节点不能同时位于DOM树的两个位置

删除节点

- removeChild() 方法从DOM中删除一个子节点

```
父节点.removeChild(要删除子节点);
```

- 节点不能主动删除自己，必须由父节点删除它

克隆节点

- cloneNode() 方法可以克隆节点，克隆出的节点是“孤儿节点”

```
var 孤儿节点 = 老节点.cloneNode();
```

```
var 孤儿节点 = 老节点.cloneNode(true);
```

- 参数是一个布尔值，表示是否采用深度克隆：如果为true，则该节点的所有后代节点也都会被克隆，如果为false，则只克隆该节点本身