

before(),after(),prepend(),append()等新DOM方法简介

这篇文章发布于 2017年09月28日, 星期四, 01:07, 归类于 [JS API](#)。阅读 18605 次, 今日 3 次 [16 条评论](#)

by zhangxinxu from <http://www.zhangxinxu.com/wordpress/?p=6443>

本文可全文转载, 但需得到原作者书面许可, 同时保留原作者和出处, 摘要引流则随意。

一、DOM API也在不断升级

DOM

Living Standard — Last Updated 14 September 2017

web前端标准一直在不断升级, 比方说, 说了很多年的HTML5、CSS3, 以及天天见的ES6。

然后, 似乎就没有然后了。实际上, 除了HTML5/CSS3/ES6+, 关于DOM标准也是在不断升级进步, 而且浏览器也在悄悄地进行跟进与支持。

然而, 这种跟进与支持呢非常的低调与隐讳; 加上行业的话语权大部分集中在JS工程师身上, 同时DOM这种东西比较“低级”撑不起场面, 因此各种什么技术大会啊, 分享会议啊绝对不会讲这玩意儿的。

于是都叫他注意力很容易被带走, 而没有意识到在DOM操作这一块, 现在实际上已经非常厉害了。

二、before(),after(),prepend(),append()等新增DOM API

接下来要介绍这些新增的DOM API方法, 都比较新, 其设计目的是可以像jQuery那样, 使用非常简单的api, 便利的操作dom元素。

这些api包括: `before()`, `after()`, `replaceWith()`, `append()`, `prepend()`, 好, 下面一个一个介绍。

1. DOM API之before()

这里的 `before()` 是个 `ChildNode` 方法, 也就是节点方法。节点方法对应于元素方法。区别在于, 节点可以直接是文本节点, 甚至注释等。但是, 元素必须要有HTML标签。

因此, `before()` 的参数既可以是DOM元素, 也可以是DOM节点, 甚至可以直接字符内容, 咦, 感觉活脱脱的jQuery的 `before()` API嘛? 没错, 真的很类似, 并且语义也是一样的, 表示当前节点的前面是XXX。

语法如下:

```
void ChildNode.before((节点或字符串)... 其它更多节点);
```

从语法可以看出before()方法支持多个参数, 并且参数可以是节点对象, 也可以是字符串对象。

至于具体细节, 我们可以通过一个一个例子来实际验证。

先来看一个最简单例子, 已知HTML如下:

```

```

如果我們希望在圖片 `` 前面插入寫文字，直接可以：

```
document.getElementById('img').before('美女: ');
```

效果會是这样：



如果我們插入的是一段HTML字符串，那效果又是怎样的呢？如下：

```
document.getElementById('img').before('<strong>美女: </strong>');
```

結果，當當當當，HTML被轉義成了安全的普通文本顯示了，如下：



可以看出原生DOM的 `before()` API和jQuery中的 `before()` API還是有差別的，在jQuery中，`before()` 的參數值是作為html字符處理的，但是這裡的 `before()` 是作為text字符處理的。

如果我們想要在圖片前面插入HTML內容，可以使用DOM節點方式插入，例如：

```
var eleTitle = document.createElement('h4');  
eleTitle.innerHTML = '美女: ';  
document.getElementById('img').before(eleTitle);
```

則會表現如下：



可能有人要疑問了，我非要在圖片前面插入HTML字符內容怎麼辦？

可以使用兼容性更好的 `insertAdjacentHTML()` DOM方法，使用示意如下：

```
document.getElementById('img').insertAdjacentHTML('beforebegin', '<strong>美女: </strong>')  
;
```

語法如下：

```
element.insertAdjacentHTML(position, html);
```

非本文重点就不展开了。

元素DOM的 `before()` API还有一个很棒的特性，那就是可以同时插入多个节点内容，例如：

```
document.getElementById('img').before('1. 美女', ' ', '2. 美女');
```

效果如下截图所示：



demo演示

以上所有出现的 `before()` 效果都可以狠狠地点击[这里](#)体验：[DOM before\(\)节点API方法demo](#)

兼容性

`before()` API Chrome54+，Firefox49+才支持，还算比较新，IE edge目前还未支持，具体参见下面截图（实时兼容数据）：



对于团队或公司内部的一些项目，管理平台或者工具之类的web页面我们可以放心大胆使用 `before()` 等API，如果是面向用户对兼容性有要求的项目呢？

很简单，加一段polyfill JS代码就可以了，如下（参考自MDN）：

```
// 源自 https://github.com/jserz/js_piece/blob/master/DOM/ChildNode/before()/before().md
(function (arr) {
  arr.forEach(function (item) {
    if (item.hasOwnProperty('before')) {
      return;
    }
    Object.defineProperty(item, 'before', {
      configurable: true,
      enumerable: true,
      writable: true,
      value: function before() {
        var argArr = Array.prototype.slice.call(arguments),
            docFrag = document.createDocumentFragment();

        argArr.forEach(function (argItem) {
          var isNode = argItem instanceof Node;
          docFrag.appendChild(isNode ? argItem : document.createTextNode(String(argItem)))
        });

        this.parentNode.insertBefore(docFrag, this);
      }
    });
  });
})([Element.prototype, CharacterData.prototype, DocumentType.prototype]);
```

注意，上面的polyfill 并不支持IE8及其以下浏览器。也就是 `before()` API只能在至少兼容到IE9浏览器的项目使用。

和`insertBefore()`比较

`insertBefore()` 作为老牌传统的API，优势在于兼容性好。不足之处，其语法着实很奇怪，A元素插到B元素前面，需要父元素 `parentNode.insertBefore(newNode, referenceNode)`，小辈之间的打闹闹牵扯到父辈，显然事情就会麻烦。至少这么多年下来 `insertBefore` 的参数究竟是新插入节点在前还是先插入节点在后，我都没有准确记清楚。

但是，`before()` API就不一样了，语法仅涉及到插入节点和相对节点，非常好记，不容易出错，而且API名称更短。

2. DOM API之after()

`after()` 和 `before()` 的语法特性兼容性都是一一对应的，差别就在于语义上，一个是在前面插入，一个是在后面插入。

由于语法类似，因此，就不一个一个示意了，若想直观体验 `after()` 的特性表现，您可以狠狠的点击[这里：DOM after\(\)节点API方法demo](#)

然后下面的JS代码是对 `after()` API的polyfill，同样地，至少IE9+浏览器。

```
//源自 https://github.com/jserz/js_piece/blob/master/DOM/ChildNode/after()/after().md
(function (arr) {
  arr.forEach(function (item) {
    if (item.hasOwnProperty('after')) {
      return;
    }
    Object.defineProperty(item, 'after', {
      configurable: true,
      enumerable: true,
      writable: true,
      value: function after() {
        var argArr = Array.prototype.slice.call(arguments),
            docFrag = document.createDocumentFragment();

        argArr.forEach(function (argItem) {
          var isNode = argItem instanceof Node;
          docFrag.appendChild(isNode ? argItem : document.createTextNode(String(argItem)))
        });

        this.parentNode.insertBefore(docFrag, this.nextSibling);
      }
    });
  });
})([Element.prototype, CharacterData.prototype, DocumentType.prototype]);
```

3. DOM API之replaceWith()

其语法如下：

```
ChildNode.replaceWith((节点或字符串)... 更多节点);
```

表示替换当前节点为其他节点内容。

举个例子，把页面上所有HTML注释都替换成可显示的普通文本节点。

如下JS代码：

```
var treewalker = document.createTreeWalker(document.body, NodeFilter.SHOW_COMMENT);
```

```
while (treewalker.nextNode()) {
    var commentNode = treewalker.currentNode;
    // 替换注释节点为文本节点
    commentNode.replaceWith(commentNode.data);
}
```

例如页面上有这么一段注释：

点击某按钮执行上面的JS代码，结果这段注释内容变成普通文本显示在页面上了，效果如下截图：

眼见为实，您可以狠狠的点击这里：[DOM replaceWith\(\)节点API方法demo](#)

对我们开发有没有什么启示呢？比方说页面模板可以放在注释中.....

同样，如果对兼容性又要求，可以使用下面的JS polyfill（参考自MDN）：

```
function ReplaceWith(Ele) {
    var parent = this.parentNode,
        i = arguments.length,
        firstIsNode = +(parent && typeof Ele === 'object');
    if (!parent) return;

    while (i-- > firstIsNode){
        if (parent && typeof arguments[i] !== 'object'){
            arguments[i] = document.createTextNode(arguments[i]);
        } if (!parent && arguments[i].parentNode){
            arguments[i].parentNode.removeChild(arguments[i]);
            continue;
        }
        parent.insertBefore(this.previousSibling, arguments[i]);
    }
    if (firstIsNode) parent.replaceChild(this, Ele);
}

if (!Element.prototype.replaceWith) {
    Element.prototype.replaceWith = ReplaceWith;
}

if (!CharacterData.prototype.replaceWith) {
    CharacterData.prototype.replaceWith = ReplaceWith;
}

if (!DocumentType.prototype.replaceWith) {
    CharacterData.prototype.replaceWith = ReplaceWith;
}
```

4. DOM API之prepend()

其语法如下：

```
ParentNode.prepend((节点或字符串)... 更多节点);
```

表示在当前节点的最前面插入其它节点内容（作为子节点）。其意思和jQuery中的 `prepend()` API是一样的，对jQuery熟悉的人学习这几个API都是分分钟的事情。

参数值特性什么的和 `before()`，`after()` 等方法类似，就不重复展开。

若有兴趣，您可以狠狠地点击这里：[DOM prepend\(\)节点API方法demo](#)

对demo因为上的按钮乱点一通之后，会发现所有插入内容都在最前面显示：

在以前要想在元素节点的最前面插入内容，要么使用 `insertBefore()` 找 `firstChild`，要么使用 `insertAdjacentHTML()` 或者 `insertAdjacentElement()` 方法，都很啰嗦。

`prepend()`这个api要更简单和直接。

兼容性和 `before()` 一模一样，对于IE9+支持项目，可以辅助下面的polyfill：

```
// 源自: https://github.com/jserz/js_piece/blob/master/DOM/ParentNode/prepend()/prepend().md
(function (arr) {
  arr.forEach(function (item) {
    if (item.hasOwnProperty('prepend')) {
      return;
    }
    Object.defineProperty(item, 'prepend', {
      configurable: true,
      enumerable: true,
      writable: true,
      value: function prepend() {
        var argArr = Array.prototype.slice.call(arguments),
            docFrag = document.createDocumentFragment();

        argArr.forEach(function (argItem) {
          var isNode = argItem instanceof Node;
          docFrag.appendChild(isNode ? argItem : document.createTextNode(String(argItem)))
        });

        this.insertBefore(docFrag, this.firstChild);
      }
    });
  });
})([Element.prototype, Document.prototype, DocumentFragment.prototype]);
```

5. DOM API之append()

其语法如下：

```
ParentNode.append((节点或字符串)... 更多节点);
```

表示在当前节点的最后面插入其它节点内容（作为子节点）。其意思和jQuery中的 `append()` API是一样的，细节上就是不支持html字符串直接显示的差别。

您可以狠狠地点击这里：[DOM append\(\)节点API方法demo](#)

所有点击按钮插入内容都在图片后面，也就是容器里面的最后显示：

polyfill如下：

```
// 源自: https://github.com/jserz/js_piece/blob/master/DOM/ParentNode/append()/append().md
```

```
(function (arr) {
  arr.forEach(function (item) {
    if (item.hasOwnProperty('append')) {
      return;
    }
    Object.defineProperty(item, 'append', {
      configurable: true,
      enumerable: true,
      writable: true,
      value: function append() {
        var argArr = Array.prototype.slice.call(arguments),
            docFrag = document.createDocumentFragment();

        argArr.forEach(function (argItem) {
          var isNode = argItem instanceof Node;
          docFrag.appendChild(isNode ? argItem : document.createTextNode(String(argItem)))
        });

        this.appendChild(docFrag);
      }
    });
  });
})([Element.prototype, Document.prototype, DocumentFragment.prototype]);
```

三、结束语

我们为什么使用jQuery，一是DOM API的便利性，二是内部帮我们搞定了很多兼容性问题，三是友好地扩展和插件机制。

如今，原生的DOM API借鉴jQuery的优点，也整出了很多简单遍历的API方法；如果我们再粘贴一些poly fill JS搞定兼容性问题；再配合ES5很多数据处理方法；DOM leave 3的事件处理等。

基本上就没有需要使用jQuery的理由了，省了资源加载，提高了代码性能，还装了回哔哩哔哩。

所以，基本上，已经不可逆地，在不久将来，不出几年，行业会兴起原生DOM API，原生JS开发前端应用的小风尚，jQuery会越来越不被人提起，完成其历史使命，日后可以领取个终身成就奖。

另外，还有类jQuery的API本文没有介绍，例如 `Node.remove()`，这个API出现相对要早一些，和前面介绍的不是一波出来的，因此没放到一起介绍。

作用如其名，移除当前节点。

好，以上就是本文的全部内容，都是浅薄的知识点。

感谢阅读，欢迎交流！

《CSS世界》签名版独家发售，包邮，可指定寄语，点击显示购买码

（本篇完） // 想要打赏？点击[这里](#)。有话要说？点击[这里](#)。



猜你喜欢

- jquery之append与insertBefore使用实例
- 小tip: DOM appendHTML实现及insertAdjacentHTML
- CSS content内容生成技术以及应用
- CSS3 box-shadow实现纸张的曲线投影效果
- 几种纯CSS(CSS3)下的纸张效果实现展示
- 小tip: 使用CSS(Unicode字符)让inline水平元素换行
- 我是如何在实际项目中使用before/after伪元素的
- CSS counter计数器(content目录序号自动递增)详解
- CSS之before, after伪元素特性表现两则
- CSS content换行实现字符点点点loading效果
- 最近整的MooTools库下Mbox弹框插件

分享到:        { 0 }

标签: :before, after, API, append, createTreeWalker, dom, insertBefore, jQuery, prepend(), replaceWith()

发表评论（目前16条评论）

名称 (必须)

邮件地址(不会被公开) (必须)

网站

提交评论

1. 小书匠说道:
2018年06月26日 04:10
原来大神也纠结insertBefore啊.
[回复](#)



2. 妮妮说道:
2017年11月28日 15:54
嫂夫人着实好看
[回复](#)



3.

123说道:

2017年11月17日 09:28

写得很好

回复


4.

skye说道:

2017年11月3日 11:55

呃，兼容性那里好像写反了，截图上before() API的兼容性显示是：Firefox54+，Chrome49+才支持。文字的描述刚好反了。

回复


5.

一只程序猿说道:

2017年10月16日 16:39

为什么美女图不是张含韵，老夫我看不下去了，以后不看你的博客了

回复

easy-blue说道:

2017年10月31日 18:00

那可是嫂夫人

回复


6.

爱吃西红柿的兔子说道:

2017年10月13日 12:30

小师傅讲的不错

回复


7.

Web前端之家说道:

2017年10月13日 11:13

讲的比较深入啊。

回复


8.

tcdona说道:

2017年10月6日 09:59

求张含韵回归+1
哈哈哈 感谢分享

回复


9.

Plus.Zhang说道:

2017年09月29日 10:45

张含韵呢? ? ? ?

回复


10.

上官说道:

2017年09月29日 09:27

看了这么久旭爷您的博客，必须表示，美女我只服张韶涵，ㄟ~ㄟ~ㄟ23333~~~~

回复



11.

王如果说道:
2017年09月28日 12:25

鑫旭，你变了，现在美女照片不再是张含韵了！！

[回复](#)


12.

甜虾说道:
2017年09月28日 11:34

为什么美女不是张含韵。。。

[回复](#)


13.

topfe说道:
2017年09月28日 11:18

分析的不错，技术前瞻

[回复](#)


14.

zio说道:
2017年09月28日 09:29

文章很赞，有点小瑕疵，简单遍历有笔误，便利

[回复](#)


15.

sjzxc说道:
2017年09月28日 08:51

根据项目的实际情况，比如一些内部系统，我个人特别主张完全使用原生方法来实现功能。所以，看到有些同学不管什么项目都要用上jquery，陷入jquery的舒适区无法自拔，我个人觉得很无语。

[回复](#)



最新文章

- » [常见的CSS图形绘制合集](#)
- » [粉丝群第1期CSS小测点评与答疑](#)
- » [分享三个纯CSS实现26个英文字母的案例](#)
- » [小tips: 纯CSS实现打字动画效果](#)
- » [CSS/CSS3 box-decoration-break属性简介](#)
- » [CSS :placeholder-shown伪类实现Material Design占位符交互效果](#)
- » [从天猫某活动视频不必要的3次请求说起](#)
- » [CSS vector-effect与SVG stroke描边缩放](#)
- » [CSS ::backdrop伪元素是干嘛用的?](#)
- » [周知: CSS -webkit-伪元素选择器不再导致整行无效](#)

今日热门

- » [常见的CSS图形绘制合集](#) ⁽¹⁹⁰⁾
- » [未来必热: SVG Sprite技术介绍](#) ⁽¹¹⁹⁾
- » [粉丝群第1期CSS小测点评与答疑](#) ⁽¹¹⁵⁾
- » [HTML5终极备忘大全 \(图片版+文字版\)](#) ⁽⁹³⁾
- » [让所有浏览器支持HTML5 video视频标签](#) ⁽⁸⁶⁾
- » [Selectivizr-让IE6~8支持CSS3伪类和属性选择器](#) ⁽⁸²⁾
- » [CSS3下的147个颜色名称及对应颜色值](#) ⁽⁷⁹⁾
- » [视区相关单位vw, vh..简介以及可实际应用场景](#) ⁽⁷⁶⁾
- » [写给自己看的display: flex布局教程](#) ⁽⁷⁶⁾
- » [小tips: 纯CSS实现打字动画效果](#) ⁽⁷⁶⁾



今年热议

- » 《CSS世界》女主角诚寻靠谱一起奋斗之人 (76)
- » 不借助Echarts等图形框架原生JS快速实现折线图效果 (64)
- » 看，for..in和for..of在那里吵架！ (60)
- » 是时候好好安利下LuLu UI框架了！ (47)
- » 原来浏览器原生支持JS Base64编码解码 (35)
- » 妙法攻略：渐变虚框及边框滚动动画的纯CSS实现 (33)
- » 炫酷H5中序列图片视频化播放的高性能实现 (31)
- » CSS scroll-behavior和JS scrollIntoView让页面滚动平滑 (30)
- » windows系统下批量删除OS X系统.DS_Store文件 (26)
- » 写给自己看的display: flex布局教程 (26)

猜你喜欢

- jquery之append与insertBefore使用实例
- 小tip: DOM appendHTML实现及insertAdjacentHTML
- CSS content内容生成技术以及应用
- CSS3 box-shadow实现纸张的曲线投影效果
- 几种纯CSS(CSS3)下的纸张效果实现展示
- 小tip: 使用CSS(Unicode字符)让inline水平元素换行
- 我是如何在实际项目中使用before/after伪元素的
- CSS counter计数器(content目录序号自动递增)详解
- CSS之before, after伪元素特性表现两则
- CSS content换行实现字符点点loading效果
- 最近整的MooTools库下Mbox弹框插件