Mutation Observer

1、概述

Mutation Observer（变动观察器）是监视DOM变动的接口。当DOM对象树发生任何变动时，Mutation Observer会得到通知。

要概念上，它很接近事件。可以理解为，当DOM发生变动会触发Mutation Observer事件。但是，它与事件有一个本质不同：事件是同步触发，也就是说DOM发生变动立刻会触发相应的事件；Mutation Observer则是异步触发，DOM发生变动以后，并不会马上触发，而是要等到当前所有DOM操作都结束后才触发。

这样设计是为了应付DOM变动频繁的情况。举例来说，如果在文档中连续插入1000个段落（p元素），会连续触发1000个插入事件，执行每个事件的回调函数，这很可能造成浏览器的卡顿；而Mutation Observer完全不同，只在1000个段落都插入结束后才会触发，而且只触发一次。

注：在控制台可看到log

Mutation Observer有以下特点：

它等待所有脚本任务完成后，才会运行，即采用异步方式

它把DOM变动记录封装成一个数组进行处理，而不是一条条地个别处理DOM变动。

它即可以观察发生在DOM节点的所有变动，也可以观察某一类变动

目前，Firefox(14+)、Chrome(26+)、Opera(15+)、IE(11+)和Safari(6.1+)支持这个API。Safari 6.0和Chrome 18-25使用这个API的时候，需要加上WebKit前缀（WebKitMutationObserver）。可以使用下面的表达式检查浏览器是否支持这个API。

```
var MutationObserver = window.MutationObserver || window.WebKitMutationObserver ||
window.MozMutationObserver;var mutationObserverSupport = !!MutationObserver;
```

2、使用方法

首先，使用MutationObserver构造函数，新建一个实例，同时指定这个实例的回调函数。

```
var observer = new MutationObserver(callback);
```

2.1 observer方法

observer方法指定所要观察的DOM元素，以及要观察的特定变动。

```
var article = document.querySelector('article');
var options = {
'childList': true,
'arrtibutes': true
};
observer.observer(article, options);
```

上面代码首先指定，所要观察的DOM元素提article，然后指定所要观察的变动是子元素的变动和属性变动。最后，将这两个限定条件作为参数，传入observer对象的observer方法。

MutationObserver所观察的DOM变动（即上面代码的option对象），包含以下类型：

childList：子元素的变动

attributes：属性的变动

characterData：节点内容或节点文本的变动

subtree：所有下属节点（包括子节点和子节点的子节点）的变动

想要观察哪一种变动类型，就在option对象中指定它的值为true。需要注意的是，不能单独观察subtree变动，必须同时指定childList、attributes和characterData中的一种或多种。

除了变动类型，option对象还可以设定以下属性：

attributeOldValue：值为true或者为false。如果为true，则表示需要记录变动前的属性值。

characterDataOldValue：值为true或者为false。如果为true，则表示需要记录变动前的数据值。

attributesFilter：值为一个数组，表示需要观察的特定属性（比如['class', 'str']）。

2.2 disconnect方法和takeRecord方法

disconnect方法用来停止观察。发生相应变动时，不再调用回调函数。

```
observer.disconnect();
```

takeRecord方法用来清除变动记录，即不再处理未处理的变动。

```
observer.takeRecord
```

2.3 MutationRecord对象

DOM对象每次发生变化，就会生成一条变动记录。这个变动记录对应一个MutationRecord对象，该对象包含了与变动相关的所有信息。Mutation Observer进行处理的一个个变动对象所组成的数组。

MutationRecord对象包含了DOM的相关信息，有如下属性：

type:观察的变动类型（attribute、characterData或者childList）。

target:发生变动的DOM对象。

addedNodes:新增的DOM对象。

removeNodes:删除的DOM对象。

previousSibling:前一个同级的DOM对象，如果没有则返回null。

nextSibling:下一个同级的DOM对象，如果没有就返回null。

attributeName:发生变动的属性。如果设置了attributeFilter，则只返回预先指定的属性。

oldValue:变动前的值。这个属性只对attribute和characterData变动有效，如果发生childList变动，则返回null。

3、实例

3.1 子元素的变动

下面的例子说明如果读取变动记录。

```
var callback = function(records) {
    records.map(function(record) {
        console.log('Mutation type: ' + record.type);
        console.log('Mutation target: ' + record.target);
    });
};
var mo = new MutationObserver(callback);var option = {
'childList': true,
'subtree': true
};

mo.observer(document.body, option);
```

上面代码的观察器，观察body元素的所有下级元素（childList表示观察子元素，subtree表示观察子元素的下级元素）的变动。回调函数会在控制台显示所有变动的类型和目标元素。

3.2、属性的变动

下面的例子说明如何追踪属性的变动。

```
var callback = function(records) {
    records.map(function(record) {
        console.log('Previous attribute value: ' + record.oldValue);
    });
};
var mo = new MutationObserver(callback);
var element = document.getElementById('#my_element');
var option = {
'attribute': true,
'attributeOldValue': true
};
mo.observer(element, option);
```

上面代码先设定追踪属性变动（'attributes': true），然后设定记录变动前的值。实际发生变动时，会将变动前的值显示在控制台。