

02

OPEN ORIENTED

凹凸实验室

IntersectionObserver

- 你想知道DOM树的某个元素，在可视范围内的时候得到通知
- 延迟加载图片
- ...

现在比较常用的做法是，通过绑定scroll事件，然后在回调函数中调用元素的getBoundingClientRect()获取元素位置来实现这个功能。

```
function isInSight(el) {  
  const bound = el.getBoundingClientRect();  
  const clientHeight = window.innerHeight;  
  return bound.top <= clientHeight;  
}
```

但是调用getBoundingClientRect()方法会触发页面的重排，并且在iframe里面,这个方法是将iframe当做viewport,而不是父窗口。所以有没有更好的实现办法呢？

有,那就是IntersectionObserver。

什么是IntersectionObserver?

IntersectionObserver是用来监听某个元素是否滚动进了浏览器窗口的可视区域(viewport)或者滚动进了它的某个祖先元素的可视区域范围内。

它的用法非常简单

```
let io = new IntersectionObserver(callback, option);
```

callback是可见性变化时的回调函数，option是配置对象(可选参数)。

```
let el = document.querySelector('#example');
```

```
//开始监听  
io.observe(el);  
//停止监听  
io.unobserve(el);  
//关闭全部监听  
io.disconnect();
```


callback的参数是一个数组，每个数组都是一个IntersectionObserverEntry对象。包括以下属性：

| 属性 | 描述 |
|--------------------|---|
| time | 可见性发生变化的时间，单位为毫秒 |
| rootBounds | 与getBoundingClientRect()方法的返回值一样 |
| boundingClientRect | 目标元素的矩形区域的信息 |
| intersectionRect | 目标元素与视口（或根元素）的交叉区域的信息 |
| intersectionRatio | 目标元素的可见比例，即intersectionRect占boundingClientRect的比例，完全可见时为1，完全不可见时小于等于0 |
| target | 被观察的目标元素，是一个 DOM 节点对象 |

callback一般会触发两次，一次是目标刚刚进入可视区域内，另一次是完全离开窗口。

只需要用到intersectionRatio 来判断，元素是否在可视区域内。

当 intersectionRatio > 0 && intersectionRatio<=1的时候，即为在可视区域内。

option有三个参数，root，rootMargin，threshold。

root指定根元素。默认是null(即viewport)，它必须是监听目标元素的祖先元素。

rootMargin，默认是'0px 0px 0px 0px';它可以给根元素添加一个假想的margin。比如：



threshold, 这个属性决定时候触发回调函数。它是一个数组, 默认为[0]。

例如:







```
new IntersectionObserver(  
  callback => { /* ... */ },  
  {  
    threshold: [0, 0.25, 0.5, 0.75, 1]  
  }  
);
```

这就表示目标元素在可见范围0%, 25%, 50%, 75%, 100%时, 会触发回调函数

兼容性

- Chrome 51+
- Android 5+
- Edge 15
- iOS 不支持

但是，WICG提供了一个polyfill

| | | | | | | |
|---|---|--|---|---|---|---|
|  |  |  |  |  |  |  |
| ✓ | ✓ | 6+ | ✓ | 7+ | ✓ | 4.4+ |

- <https://github.com/justjavac/the-front-end-knowledge-you-may-dont-know/issues/10>
- http://www.ruanyifeng.com/blog/2016/11/intersectionobserver_api.html
- <https://developer.mozilla.org/en-US/docs/Web/API/IntersectionObserver/IntersectionObserver>
- <https://github.com/w3c/IntersectionObserver>
- <https://github.com/w3c/IntersectionObserver/tree/master/polyfill>

T H A N K S
FOR YOUR WATCHING



OPEN ORIENTED

凹凸实验室