You Don't Need jQuery

前端发展很快，现代浏览器原生 API 已经足够好用。我们并不需要为了操作 DOM、Event 等再学习一下 jQuery 的 API。同时由于 React、Angular、Vue 等框架的流行，直接操作 DOM 不再是好的模式，jQuery 使用场景大大减少。本项目总结了大部分 jQuery API 替代的方法，暂时只支持 IE10+ 以上浏览器。

目录

1. Query Selector
2. CSS & Style
3. DOM Manipulation
4. Ajax
5. Events
6. Utilities
7. Alternatives
8. Translations
9. Browser Support

Query Selector

常用的 class、id、属性 选择器都可以使用 `document.querySelector` 或 `document.querySelectorAll` 替代。区别是

- `document.querySelector` 返回第一个匹配的 Element
- `document.querySelectorAll` 返回所有匹配的 Element 组成的 NodeList。它可以通过 `[].slice.call()` 把它转成 Array
- 如果匹配不到任何 Element，jQuery 返回空数组 `[]`，但 `document.querySelector` 返回 `null`，注意空指针异常。当找不到时，也可以使用 `||` 设置默认的值，如 `document.querySelectorAll(selector) || []`

> *注意：`document.querySelector` 和 `document.querySelectorAll` 性能很**差**。如果想提高性能，尽量使用 `document.getElementById`、`document.getElementsByClassName` 或 `document.getElementsByTagName`。*

- 1.0 Query by selector

```
// jQuery
$('selector');

// Native
document.querySelectorAll('selector');
```

- 1.1 Query by class

```
// jQuery
$('.css');

// Native
document.querySelectorAll('.css');
```

```
// or
document.getElementsByClassName('css');
```

- 1.2 Query by id

```
// jQuery
$('#id');

// Native
document.querySelector('#id');

// or
document.getElementById('id');
```

- 1.3 Query by attribute

```
// jQuery
$('a[target=_blank]');

// Native
document.querySelectorAll('a[target=_blank]');
```

- 1.4 Find sth.
  - Find nodes

```
// jQuery
$el.find('li');

// Native
el.querySelectorAll('li');
```

  - Find body

```
// jQuery
$('body');

// Native
document.body;
```

  - Find Attribute

```
// jQuery
$el.attr('foo');

// Native
e.getAttribute('foo');
```

  - Find data attribute

```
// jQuery
$el.data('foo');

// Native
// using getAttribute
el.getAttribute('data-foo');
// you can also use `dataset` if only need to support IE 11+
el.dataset['foo'];
```

- 1.5 Sibling/Previous/Next Elements
  - Sibling elements

```
// jQuery
$el.siblings();

// Native
[].filter.call(el.parentNode.children, function(child) {
  return child !== el;
});
```

  - Previous elements

```
// jQuery
$el.prev();

// Native
el.previousElementSibling;
```

  - Next elements

```
// next
$el.next();
el.nextElementSibling;
```

- 1.6 Closest

Closest 获得匹配选择器的第一个祖先元素，从当前元素开始沿 DOM 树向上。

```
// jQuery
$el.closest(queryString);

// Native
function closest(el, selector) {
  const matchesSelector = el.matches || el.webkitMatchesSelector ||
el.mozMatchesSelector || el.msMatchesSelector;

  while (el) {
    if (matchesSelector.call(el, selector)) {
```

```
      return el;
    } else {
      el = el.parentElement;
    }
  }
  return null;
}
```

- 1.7 Parents Until

  获取当前每一个匹配元素集的祖先，不包括匹配元素的本身。

```
// jQuery
$el.parentsUntil(selector, filter);

// Native
function parentsUntil(el, selector, filter) {
  const result = [];
  const matchesSelector = el.matches || el.webkitMatchesSelector ||
el.mozMatchesSelector || el.msMatchesSelector;

  // match start from parent
  el = el.parentElement;
  while (el && !matchesSelector.call(el, selector)) {
    if (!filter) {
      result.push(el);
    } else {
      if (matchesSelector.call(el, filter)) {
        result.push(el);
      }
    }
    el = el.parentElement;
  }
  return result;
}
```

- 1.8 Form
  - Input/Textarea

```
// jQuery
$('#my-input').val();

// Native
document.querySelector('#my-input').value;
```

  - Get index of e.currentTarget between `.radio`

```
// jQuery
$(e.currentTarget).index('.radio');
```

```
// Native
[].indexOf.call(document.querySelectAll('.radio'), e.currentTarget);
```

- 1.9 Iframe Contents

  jQuery 对象的 iframe `contents()` 返回的是 iframe 内的 `document`

  - Iframe contents

    ```
    // jQuery
    $iframe.contents();

    // Native
    iframe.contentDocument;
    ```

  - Iframe Query

    ```
    // jQuery
    $iframe.contents().find('.css');

    // Native
    iframe.contentDocument.querySelectorAll('.css');
    ```

⬆ 回到顶部

CSS & Style

- 2.1 CSS

  - Get style

    ```
    // jQuery
    $el.css("color");

    // Native
    // 注意：此处为了解决当 style 值为 auto 时，返回 auto 的问题
    const win = el.ownerDocument.defaultView;
    // null 的意思是不返回伪类元素
    win.getComputedStyle(el, null).color;
    ```

  - Set style

    ```
    // jQuery
    $el.css({ color: "#ff0011" });

    // Native
    el.style.color = '#ff0011';
    ```

  - Get/Set Styles

注意，如果想一次设置多个 style，可以参考 oui-dom-utils 中 setStyles 方法

- Add class

```
// jQuery
$el.addClass(className);

// Native
el.classList.add(className);
```

- Remove class

```
// jQuery
$el.removeClass(className);

// Native
el.classList.remove(className);
```

- has class

```
// jQuery
$el.hasClass(className);

// Native
el.classList.contains(className);
```

- Toggle class

```
// jQuery
$el.toggleClass(className);

// Native
el.classList.toggle(className);
```

- 2.2 Width & Height

  Width 与 Height 获取方法相同，下面以 Height 为例：

  - Window height

```
// jQuery
$(window).height();

// Native
// 不含 scrollbar，与 jQuery 行为一致
window.document.documentElement.clientHeight;
// 含 scrollbar
window.innerHeight;
```

- Document height

```
// jQuery
$(document).height();

// Native
document.documentElement.scrollHeight;
```

- Element height

```
// jQuery
$el.height();

// Native
// 与 jQuery 一致（一直为 content 区域的高度）
function getHeight(el) {
  const styles = this.getComputedStyles(el);
  const height = el.offsetHeight;
  const borderTopWidth = parseFloat(styles.borderTopWidth);
  const borderBottomWidth = parseFloat(styles.borderBottomWidth);
  const paddingTop = parseFloat(styles.paddingTop);
  const paddingBottom = parseFloat(styles.paddingBottom);
  return height - borderBottomWidth - borderTopWidth - paddingTop -
paddingBottom;
}
// 精确到整数（border-box 时为 height 值，content-box 时为 height +
padding + border 值）
el.clientHeight;
// 精确到小数（border-box 时为 height 值，content-box 时为 height +
padding + border 值）
el.getBoundingClientRect().height;
```

- Iframe height

  $iframe .contents() 方法返回 iframe 的 contentDocument

```
// jQuery
$('iframe').contents().height();

// Native
iframe.contentDocument.documentElement.scrollHeight;
```

- 2.3 Position & Offset
  - Position

```
// jQuery
$el.position();
```

```
// Native
{ left: el.offsetLeft, top: el.offsetTop }
```

- Offset

```
// jQuery
$el.offset();

// Native
function getOffset (el) {
  const box = el.getBoundingClientRect();

  return {
    top: box.top + window.pageYOffset -
document.documentElement.clientTop,
    left: box.left + window.pageXOffset -
document.documentElement.clientLeft
  }
}
```

- 2.4 Scroll Top

```
// jQuery
$(window).scrollTop();

// Native
(document.documentElement && document.documentElement.scrollTop) ||
document.body.scrollTop;
```

↑ 回到顶部

DOM Manipulation

- 3.1 Remove

```
// jQuery
$el.remove();

// Native
el.parentNode.removeChild(el);
```

- 3.2 Text

- Get text

```
// jQuery
$el.text();
```

```
// Native
el.textContent;
```

- Set text

```
// jQuery
$el.text(string);

// Native
el.textContent = string;
```

- 3.3 HTML
  - Get HTML

```
// jQuery
$el.html();

// Native
el.innerHTML;
```

  - Set HTML

```
// jQuery
$el.html(htmlString);

// Native
el.innerHTML = htmlString;
```

- 3.4 Append

  Append 插入到子节点的末尾

```
// jQuery
$el.append("<div id='container'>hello</div>");

// Native
let newEl = document.createElement('div');
newEl.setAttribute('id', 'container');
newEl.innerHTML = 'hello';
el.appendChild(newEl);
```

- 3.5 Prepend

```
// jQuery
$el.prepend("<div id='container'>hello</div>");

// Native
let newEl = document.createElement('div');
```

```
newEl.setAttribute('id', 'container');
newEl.innerHTML = 'hello';
el.insertBefore(newEl, el.firstChild);
```

- 3.6 insertBefore

  在选中元素前插入新节点

```
// jQuery
$newEl.insertBefore(queryString);

// Native
const target = document.querySelector(queryString);
target.parentNode.insertBefore(newEl, target);
```

- 3.7 insertAfter

  在选中元素后插入新节点

```
// jQuery
$newEl.insertAfter(queryString);

// Native
const target = document.querySelector(queryString);
target.parentNode.insertBefore(newEl, target.nextSibling);
```

↑ 回到顶部

Ajax

用 fetch 和 fetch-jsonp 替代

↑ 回到顶部

Events

完整地替代命名空间和事件代理，链接到 https://github.com/oneuijs/oui-dom-events

- 5.1 Bind an event with on

```
// jQuery
$el.on(eventName, eventHandler);

// Native
el.addEventListener(eventName, eventHandler);
```

- 5.2 Unbind an event with off

```
// jQuery
$el.off(eventName, eventHandler);
```

```
// Native
el.removeEventListener(eventName, eventHandler);
```

- 5.3 Trigger

```
// jQuery
$(el).trigger('custom-event', {key1: 'data'});

// Native
if (window.CustomEvent) {
  const event = new CustomEvent('custom-event', {detail: {key1: 'data'}});
} else {
  const event = document.createEvent('CustomEvent');
  event.initCustomEvent('custom-event', true, true, {key1: 'data'});
}

el.dispatchEvent(event);
```

↑ 回到顶部

Utilities

- 6.1 isArray

```
// jQuery
$.isArray(range);

// Native
Array.isArray(range);
```

- 6.2 Trim

```
// jQuery
$.trim(string);

// Native
string.trim();
```

- 6.3 Object Assign

  继承，使用 object.assign polyfill https://github.com/ljharb/object.assign

```
// jQuery
$.extend({}, defaultOpts, opts);

// Native
Object.assign({}, defaultOpts, opts);
```

- 6.4 Contains

```
// jQuery
$.contains(el, child);

// Native
el !== child && el.contains(child);
```

⬆ 回到顶部

Alternatives

- 你可能不需要 jQuery (You Might Not Need jQuery) - 如何使用原生 JavaScript 实现通用事件，元素，ajax 等用法。
- npm-dom 以及 webmodules - 在 NPM 上提供独立 DOM 模块的组织

Translations

- 한국어
- 简体中文
- Bahasa Melayu
- Bahasa Indonesia
- Português(PT-BR)
- Tiếng Việt Nam
- Español
- Русский
- Türkçe
- Italian

Browser Support

Chrome	Firefox	IE	Opera	Safari
Latest ✓	Latest ✓	10+ ✓	Latest ✓	6.1+ ✓

License