You Don't Need jQuery

Frontend environments evolve rapidly nowadays, modern browsers have already implemented a great deal of DOM/BOM APIs which are good enough. We don't have to learn jQuery from scratch for DOM manipulation or events. In the meantime, thanks to the prevailment of frontend libraries such as React, Angular and Vue, manipulating DOM directly becomes anti-pattern, jQuery has never been less important. This project summarizes most of the jQuery method alternatives in native implementation, with IE 10+ support.

Table of Contents

Query Selector

In place of common selectors like class, id or attribute we can use `document.querySelector` or `document.querySelectorAll` for substitution. The differences lie in:

- `document.querySelector` returns the first matched element
- `document.querySelectorAll` returns all matched elements as NodeList. It can be converted to Array using `[].slice.call(document.querySelectorAll(selector) || []);`
- If no elements matched, jQuery would return `[]` whereas the DOM API will return `null`. Pay attention to Null Pointer Exception. You can also use `||` to set default value if not found, like `document.querySelectorAll(selector) || []`

> *Notice: `document.querySelector` and `document.querySelectorAll` are quite **SLOW**, try to use `getElementById`, `document.getElementsByClassName` or `document.getElementsByTagName` if you want to get a performance bonus.*

- 1.0 Query by selector

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

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

- 1.1 Query by class

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

// Native
document.querySelectorAll('.class');

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

- 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 Query in descendents

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

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

- 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

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

// Native
el.nextElementSibling;
```

- 1.6 Closest

  Return the first matched element by provided selector, traversing from current element to document.

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

// Native - Only latest, NO IE
el.closest(selector);

// Native - IE10+
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

  Get the ancestors of each element in the current set of matched elements, up to but not including the element matched by the selector, DOM node, or jQuery object.

```
// 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

  `$('iframe').contents()` returns `contentDocument` for this specific iframe

  - Iframe contents

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

    // Native
    iframe.contentDocument;
    ```

  - Iframe Query

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

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

- 1.10 Get body

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

// Native
document.body;
```

- 1.11 Attribute getter and setter
  - Get an attribute

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

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

  - Set an attribute

```
// jQuery, note that this works in memory without change the DOM
$el.attr('foo', 'bar');

// Native
el.setAttribute('foo', 'bar');
```

  - Get a `data-` attribute

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

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

↑ back to top

CSS & Style

- 2.1 CSS

- Get style

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

// Native
// NOTE: Known bug, will return 'auto' if style value is 'auto'
const win = el.ownerDocument.defaultView;
// null means not return pseudo styles
win.getComputedStyle(el, null).color;
```

- Set style

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

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

- Get/Set Styles

  Note that if you want to set multiple styles once, you could refer to setStyles method in oui-dom-utils package.

- 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 and Height are theoretically identical, take Height as example:

  - Window height

    ```
    // window height
    $(window).height();
    // without scrollbar, behaves like jQuery
    window.document.documentElement.clientHeight;
    // with scrollbar
    window.innerHeight;
    ```

  - Document height

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

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

  - Element height

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

    // Native
    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;
    }
    // accurate to integer (when `border-box`, it's `height`; when
    `content-box`, it's `height + padding + border`)
    el.clientHeight;
    // accurate to decimal (when `border-box`, it's `height`; when
    `content-box`, it's `height + padding + border`)
    el.getBoundingClientRect().height;
    ```

- 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;
```

↑ back to top

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 child element after the last child of parent element

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

// Native
el.insertAdjacentHTML("beforeend","<div id='container'>hello</div>");
```

- 3.5 Prepend

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

// Native
el.insertAdjacentHTML("afterbegin","<div id='container'>hello</div>");
```

- 3.6 insertBefore

  Insert a new node before the selected elements

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

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

- 3.7 insertAfter

  Insert a new node after the selected elements

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

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

- 3.8 is

  Return `true` if it matches the query selector

```
// jQuery - Notice `is` also work with `function` or `elements` which is not
concerned here
$el.is(selector);

// Native
el.matches(selector);
```

- 3.9 clone

  Create a deep copy of that element

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

// Native
el.cloneNode();

// For Deep clone , set param as `true`
```

- 3.10 empty

  Remove all child nodes

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

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

- 3.11 wrap

  Wrap an HTML structure around each element

```
// jQuery
$('.inner').wrap('<div class="wrapper"></div>');

// Native
[].slice.call(document.querySelectorAll('.inner')).forEach(function(el){
  var wrapper = document.createElement('div');
  wrapper.className = 'wrapper';
  el.parentNode.insertBefore(wrapper, el);
  el.parentNode.removeChild(el);
  wrapper.appendChild(el);
});
```

- 3.12 unwrap

  Remove the parents of the set of matched elements from the DOM

```
// jQuery
$('.inner').unwrap();

// Native
[].slice.call(document.querySelectorAll('.inner')).forEach(function(el){
  [].slice.call(el.childNodes).forEach(function(child){
    el.parentNode.insertBefore(child, el);
  });
  el.parentNode.removeChild(el);
});
```

  - 3.13 replaceWith

  Replace each element in the set of matched elements with the provided new content

```
// jQuery
$('.inner').replaceWith('<div class="outer"></div>');

// Native
[].slice.call(document.querySelectorAll('.inner')).forEach(function(el){
  var outer = document.createElement('div');
```

```
  outer.className = 'outer';
  el.parentNode.insertBefore(outer, el);
  el.parentNode.removeChild(el);
});
```

↑ back to top

Ajax

Fetch API is the new standard to replace XMLHttpRequest to do ajax. It works on Chrome and Firefox, you can use polyfills to make it work on legacy browsers.

Try github/fetch on IE9+ or fetch-ie8 on IE8+, fetch-jsonp to make JSONP requests.

↑ back to top

Events

For a complete replacement with namespace and delegation, refer to 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);
```

[↑ back to top](#)

Utilities

Most of utilities are found by native API. Others advanced functions could be choosed better utilities library focus on consistency and performance. Recommend lodash to replace.

- 6.1 Basic utilities
  - isArray

  Determine whether the argument is an array.

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

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

  - isWindow

  Determine whether the argument is a window.

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

// Native
function isWindow(obj) {
  return obj != null && obj === obj.window;
}
```

  - inArray

  Search for a specified value within an array and return its index (or -1 if not found).

```
// jQuery
$.inArray(item, array);

// Native
Array.indexOf(item);
```

  - isNumbic

  Determines whether its argument is a number. Use `typeof` to decide type. if necessary to use library, sometimes `typeof` isn't accurate.

```
// jQuery
$.isNumbic(item);

// Native
function isNumbic(item) {
```

```
  return typeof value === 'number';
}
```

- isFunction

Determine if the argument passed is a JavaScript function object.

```
// jQuery
$.isFunction(item);

// Native
function isFunction(item) {
  return typeof value === 'function';
}
```

- isEmptyObject

Check to see if an object is empty (contains no enumerable properties).

```
// jQuery
$.isEmptyObject(obj);

// Native
function isEmptyObject(obj) {
  for (let key in obj) {
    return false;
  }
  return true;
}
```

- isPlanObject

Check to see if an object is a plain object (created using “{}” or “new Object”).

```
// jQuery
$.isPlanObject(obj);

// Native
function isPlainObject(obj) {
  if (typeof (obj) !== 'object' || obj.nodeType || obj != null && obj ===
obj.window) {
    return false;
  }

  if (obj.constructor &&
      !{}.hasOwnPropert.call(obj.constructor.prototype, 'isPrototypeOf')) {
    return false;
  }

  return true;
}
```

- extend

Merge the contents of two or more objects together into the first object. object.assign is ES6 API, and you could use polyfill also.

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

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

- trim

Remove the whitespace from the beginning and end of a string.

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

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

- map

Translate all items in an array or object to new array of items.

```
// jQuery
$.map(array, function(value, index) {
});

// Native
array.map(function(value, index) {
});
```

- each

A generic iterator function, which can be used to seamlessly iterate over both objects and arrays.

```
// jQuery
$.each(array, function(value, index) {
});

// Native
array.forEach(function(value, index) {
});
```

- grep

Finds the elements of an array which satisfy a filter function.

```
// jQuery
$.grep(array, function(value, index) {
});
```

```
// Native
array.filter(function(value, index) {
});
```

- type

Determine the internal JavaScript [[Class]] of an object.

```
// jQuery
$.type(obj);

// Native
Object.prototype.toString.call(obj).replace(/^\[object (.+)\]$/,
'$1').toLowerCase();
```

- merge

Merge the contents of two arrays together into the first array.

```
// jQuery
$.merge(array1, array2);

// Native
// But concat function don't remove duplicate items.
function merge() {
  return Array.prototype.concat.apply([], arguments)
}
```

- now

Return a number representing the current time.

```
// jQuery
$.now();

// Native
Date.now();
```

- proxy

Takes a function and returns a new one that will always have a particular context.

```
// jQuery
$.proxy(fn, context);

// Native
fn.bind(context);
```

- makeArray

Convert an array-like object into a true JavaScript array.

```
// jQuery
$.makeArray(array);

// Native
[].slice.call(array);
```

- 6.2 DOM utilities
    - unique

    Sorts an array of DOM elements, in place, with the duplicates removed. Note that this only works on arrays of DOM elements, not strings or numbers.

    Sizzle's API
    - contains

    Check to see if a DOM element is a descendant of another DOM element.

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

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

- 6.3 Globaleval

```
// jQuery
$.globaleval(code);

// Native
function Globaleval(code) {
  let script = document.createElement('script');
  script.text = code;

  document.head.appendChild(script).parentNode.removeChild(script);
}

// Use eval, but context of eval is current, context of $.Globaleval is
global.
eval(code);
```

- 6.4 parse
    - parseHTML

    Parses a string into an array of DOM nodes.

```
// jQuery
$.parseHTML(htmlString);

// Native
```

```
function parseHTML(string) {
  const tmp = document.implementation.createHTMLDocument();
  tmp.body.innerHTML = string;
  return tmp.body.children;
}
```

- parseJSON

Takes a well-formed JSON string and returns the resulting JavaScript value.

```
// jQuery
$.parseJSON(str);

// Native
JSON.parse(str);
```

[↑ back to top]

Promises

A promise represents the eventual result of an asynchronous operation. jQuery has its own way to handle promises. Native JavaScript implements a thin and minimal API to handle promises according to the Promises/A+ specification.

- 7.1 done, fail, always

  `done` is called when promise is resolved, `fail` is called when promise is rejected, `always` is called when promise is either resolved or rejected.

  ```
  // jQuery
  $promise.done(doneCallback).fail(failCallback).always(alwaysCallback)

  // Native
  promise.then(doneCallback, failCallback).then(alwaysCallback, alwaysCallback)
  ```

- 7.2 when

  `when` is used to handle multiple promises. It will resolve when all promises are resolved, and reject if either one is rejected.

  ```
  // jQuery
  $.when($promise1, $promise2).done((promise1Result, promise2Result) => {})

  // Native
  Promise.all([$promise1, $promise2]).then([promise1Result, promise2Result] =>
  {});
  ```

- 7.3 Deferred

  Deferred is a way to create promises.

```
// jQuery
function asyncFunc() {
  var d = new $.Deferred();
  setTimeout(function() {
    if(true) {
      d.resolve('some_value_compute_asynchronously');
    } else {
      d.reject('failed');
    }
  }, 1000);
  return d.promise();
}

// Native
function asyncFunc() {
  return new Promise((resolve, reject) => {
    setTimeout(function() {
      if (true) {
        resolve('some_value_compute_asynchronously');
      } else {
        reject('failed');
      }
    }, 1000);
  });
}
```

[↑ back to top]

Animation

- 8.1 Show & Hide

```
// jQuery
$el.show();
$el.hide();

// Native
// More detail about show method, please refer to
https://github.com/oneuijs/oui-dom-utils/blob/master/src/index.js#L363
el.style.display = ''|'inline'|'inline-block'|'inline-table'|'block';
el.style.display = 'none';
```

- 8.2 Toggle

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

// Native
if (el.ownerDocument.defaultView.getComputedStyle(el, null).display ===
```

```
'none') {
  el.style.display = ''|'inline'|'inline-block'|'inline-table'|'block';
}
else {
  el.style.display = 'none';
}
```

- 8.3 FadeIn & FadeOut

```
// jQuery
$el.fadeIn(3000);
$el.fadeOut(3000);

// Native
el.style.transition = 'opacity 3s';
// fadeIn
el.style.opacity = '1';
// fadeOut
el.style.opacity = '0';
```

- 8.4 FadeTo

```
// jQuery
$el.fadeTo('slow',0.15);
// Native
el.style.transition = 'opacity 3s'; // assume 'slow' equals 3 seconds
el.style.opacity = '0.15';
```

- 8.5 FadeToggle

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

// Native
el.style.transition = 'opacity 3s';
let { opacity } = el.ownerDocument.defaultView.getComputedStyle(el, null);
if (opacity === '1') {
  el.style.opacity = '0';
}
else {
  el.style.opacity = '1';
}
```

- 8.6 SlideUp & SlideDown

```
// jQuery
$el.slideUp();
$el.slideDown();

// Native
```

```
let originHeight = '100px';
el.style.transition = 'height 3s';
// slideUp
el.style.height = '0px';
// slideDown
el.style.height = originHeight;
```

- 8.7 SlideToggle

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

// Native
let originHeight = '100px';
el.style.transition = 'height 3s';
let { height } = el.ownerDocument.defaultView.getComputedStyle(el, null);
if (parseInt(height, 10) === 0) {
  el.style.height = originHeight;
}
else {
 el.style.height = '0px';
}
```

- 8.8 Animate

```
// jQuery
$el.animate({params}, speed);

// Native
el.style.transition = 'all' + speed;
Object.keys(params).forEach(function(key) {
  el.style[key] = params[key];
})
```

Alternatives

- You Might Not Need jQuery - Examples of how to do common event, element, ajax etc with plain javascript.
- npm-dom and webmodules - Organizations you can find individual DOM modules on NPM

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

MIT