

Tú no necesitas jQuery

El desarrollo Frontend evoluciona día a día, y los navegadores modernos ya han implementado nativamente APIs para trabajar con DOM/BOM, las cuales son muy buenas, por lo que definitivamente no es necesario aprender jQuery desde cero para manipular el DOM. En la actualidad, gracias al surgimiento de librerías frontend como React, Angular y Vue, manipular el DOM es contrario a los patrones establecidos, y jQuery se ha vuelto menos importante. Este proyecto resume la mayoría de métodos alternativos a jQuery, pero de forma nativa con soporte IE 10+.

Tabla de Contenidos

1. [Query Selector](#)
2. [CSS & Estilo](#)
3. [Manipulación DOM](#)
4. [Ajax](#)
5. [Eventos](#)
6. [Utilidades](#)
7. [Traducción](#)
8. [Soporte de Navegadores](#)

Query Selector

En lugar de los selectores comunes como clase, id o atributos podemos usar `document.querySelector` o `document.querySelectorAll` como alternativas. Las diferencias radican en:

- `document.querySelector` devuelve el primer elemento que cumpla con la condición
- `document.querySelectorAll` devuelve todos los elementos que cumplen con la condición en forma de `NodeList`. Puede ser convertido a Array usando

```
[].slice.call(document.querySelectorAll(selector) || []);
```
- Si ningún elemento cumple con la condición, jQuery retornaría `[]` mientras la API DOM retornaría `null`. Nótese el `NullPointerException`. Se puede usar `||` para establecer el valor por defecto al no encontrar elementos, como en `document.querySelectorAll(selector) || []`

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

- [1.0](#) Buscar por selector

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

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

- [1.1](#) Buscar por Clase

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

```
// Nativo
document.querySelectorAll('.class');

// Forma alternativa
document.getElementsByClassName('class');
```

- [1.2](#) Buscar por id

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

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

// Forma alternativa
document.getElementById('id');
```

- [1.3](#) Buscar por atributo

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

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

- [1.4](#) Buscar

- Buscar nodos

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

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

- Buscar "body"

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

// Nativo
document.body;
```

- Buscar Atributo

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

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

- Buscar atributo "data"

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

// Nativo
// Usando getAttribute
el.getAttribute('data-foo');
// También puedes utilizar `dataset` desde IE 11+
el.dataset['foo'];
```

- [1.5](#) Elementos Hermanos/Previos/Siguientes

- Elementos hermanos

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

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

- Elementos previos

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

// Nativo
el.previousElementSibling;
```

- Elementos siguientes

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

// Nativo
el.nextElementSibling;
```

- [1.6](#) Closest

Retorna el elemento más cercano que coincida con la condición, partiendo desde el nodo actual hasta document.

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

```
// Nativo
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

Obtiene los ancestros de cada elemento en el set actual de elementos que cumplan con la condición, sin incluir el actual

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

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

  // Partir desde el elemento padre
  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](#) Formularios

- Input/Textarea

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

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

- Obtener el índice de e.currentTarget en `.radio`

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

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

- [1.9](#) Contenidos de Iframe

`$('iframe').contents()` devuelve `contentDocument` para este iframe específico

- Contenidos de Iframe

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

// Nativo
iframe.contentDocument;
```

- Buscar dentro de un Iframe

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

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

[↑ volver al inicio](#)

CSS & Estilo

- [2.1](#) CSS

- Obtener Estilo

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

// Nativo
// NOTA: Bug conocido, retornará 'auto' si el valor de estilo es
'auto'
const win = el.ownerDocument.defaultView;
// null significa que no tiene pseudo estilos
win.getComputedStyle(el, null).color;
```

- Establecer style

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

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

- Obtener/Establecer Estilos

Nótese que si se desea establecer múltiples estilos a la vez, se puede utilizar el método [setStyles](#) en el paquete oui-dom-utils.

- Agregar clase

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

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

- Quitar Clase

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

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

- Consultar si tiene clase

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

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

- Toggle class

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

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

- [2.2](#) Width & Height

Ancho y Alto son teóricamente idénticos. Usaremos el Alto como ejemplo:

- Alto de Ventana

```
// alto de ventana
$(window).height();

// Sin scrollbar, se comporta como jQuery
window.document.documentElement.clientHeight;

// Con scrollbar
window.innerHeight;
```

- Alto de Documento

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

// Nativo
document.documentElement.scrollHeight;
```

- Alto de Elemento

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

// Nativo
function getHeight(el) {
    const styles = this.getComputedStyle(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;
}

// Precisión de integer (when `border-box`, it's `height`; when
`content-box`, it's `height + padding + border`)
el.clientHeight;

// Precisión de decimal (when `border-box`, it's `height`; when
`content-box`, it's `height + padding + border`)
el.getBoundingClientRect().height;
```

- [2.3](#) Posición & Offset

- Posición

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

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

- Offset

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

// Nativo
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](#) Posición del Scroll Vertical

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

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

[↑ volver al inicio](#)

Manipulación DOM

- [3.1](#) Remove

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

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

- [3.2](#) Text

- Obtener Texto

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

// Nativo
el.textContent;
```

- Establecer Texto


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

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

- [3.3](#) HTML

- Obtener HTML

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

// Nativo
el.innerHTML;
```

- Establecer HTML

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

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

- [3.4](#) Append

Añadir elemento hijo después del último hijo del elemento padre

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

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

- [3.5](#) Prepend

Añadir elemento hijo después del último hijo del elemento padre

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

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

- [3.6](#) insertBefore

Insertar un nuevo nodo antes del primero de los elementos seleccionados

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

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

- [3.7](#) insertAfter

Insertar un nuevo nodo después de los elementos seleccionados

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

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

[↑ volver al inicio](#)

Ajax

Reemplazar con [fetch](#) y [fetch-jsonp](#) + [Fetch API](#) es el nuevo estándar que reemplaza a XMLHttpRequest para efectuar peticiones AJAX. Funciona en Chrome y Firefox, como también es posible usar un polyfill en otros navegadores. + Es una buena alternativa utilizar [github/fetch](#) en IE9+ o [fetch-ie8](#) en IE8+, [fetch-jsonp](#) para efectuar peticiones JSONP. [↑ volver al inicio](#)

Eventos

Para un reemplazo completo con namespace y delegación, utilizar <https://github.com/oneuijs/oui-dom-events>

- [5.1](#) Asignar un evento con "on"

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

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

- [5.2](#) Desasignar un evento con "off"

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

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

- [5.3](#) Trigger

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

// Nativo
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);
```

[↑ volver al inicio](#)

Utilidades

- [6.1](#) isArray

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

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

- [6.2](#) Trim

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

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

- [6.3](#) Object Assign

Utilizar polyfill para object.assign <https://github.com/ljharb/object.assign>

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

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

- [6.4](#) Contains

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



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

[↑ volver al inicio](#)

Traducción

- [한국어](#)
- [简体中文](#)
- [Bahasa Melayu](#)
- [Bahasa Indonesia](#)
- [Português\(PT-BR\)](#)
- [Tiếng Việt Nam](#)
- [Español](#)
- [Русский](#)
- [Türkçe](#)

Soporte de Navegadores

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

Licencia

MIT