

Non hai bisogno di jQuery

Il mondo del Frontend si evolve rapidamente oggi, i browsers moderni hanno già implementato un'ampia gamma di DOM/BOM API soddisfacenti. Non dobbiamo imparare jQuery dalle fondamenta per la manipolazione del DOM o di eventi. Nel frattempo, grazie al prevalere di librerie per il frontend come React, Angular o Vue, manipolare il DOM direttamente diventa un anti-pattern, di conseguenza jQuery non è mai stato meno importante. Questo progetto sommarizza la maggior parte dei metodi e implementazioni alternative a jQuery, con il supporto di IE 10+.

Tabella contenuti

1. [Query Selector](#)
2. [CSS & Style](#)
3. [Manipolazione DOM](#)
4. [Ajax](#)
5. [Eventi](#)
6. [Utilities](#)
7. [Alternative](#)
8. [Traduzioni](#)
9. [Supporto Browsers](#)

Query Selector

Al posto di comuni selettori come class, id o attributi possiamo usare `document.querySelector` o `document.querySelectorAll` per sostituzioni. La differenza risiede in:

- `document.querySelector` restituisce il primo elemento combaciante
- `document.querySelectorAll` restituisce tutti gli elementi combacianti della NodeList. Può essere convertito in Array usando `[].slice.call(document.querySelectorAll(selector)) || []`;
- Se nessun elemento combacia, jQuery restituirebbe `[]` lì dove il DOM API ritornerà `null`. Prestate attenzione al Null Pointer Exception. Potete anche usare `||` per settare valori di default se non trovato, come `document.querySelectorAll(selector) || []`

*Notare: `document.querySelector` e `document.querySelectorAll` sono abbastanza **SLOW**, provate ad usare `getElementById`, `document.getElementsByClassName` o `document.getElementsByTagName` se volete avere un bonus in termini di performance.*

- [1.0](#) Query da selettore

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

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

- [1.1](#) Query da classe

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

// Nativo
```

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

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

- [1.2](#) Query da id

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

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

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

- [1.3](#) Query da attributo

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

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

- [1.4](#) Trovare qualcosa.

- Trovare nodes

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

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

- Trovare body

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

```
// Nativo
document.body;
```

- Trovare Attributi

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

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

- Trovare attributo data

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

// Nativo
// using getAttribute
el.getAttribute('data-foo');
// potete usare `dataset` solo se supportate IE 11+
el.dataset['foo'];
```

- [1.5](#) Fratelli/Precedento/Successivo Elemento

- Elementi fratelli

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

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

- Elementi precedenti

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

// Nativo
el.previousElementSibling;
```

- Elementi successivi

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

// Nativo
el.nextElementSibling;
```

- [1.6](#) Il piu' vicino

Restituisce il primo elementi combaciante il selettore fornito, attraversando dall'elemento corrente fino al document .

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

// Nativo - Solo ultimo, NO IE
el.closest(selector);
```

```
// Nativo - 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](#) Fino a parenti

Ottiene il parente di ogni elemento nel set corrente di elementi combiacianti, fino a ma non incluso, l'elemento combiaciante il selettore, DOM node, o jQuery object.

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

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

  // il match parte dal parente
  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');

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

- [1.9](#) Iframe Contents

`$('iframe').contents()` restituisce `contentDocument` per questo specifico iframe

- Iframe contenuti

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

// Nativo
iframe.contentDocument;
```

- Iframe Query

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

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

[↑ back to top](#)

CSS & Style

- [2.1](#) CSS

- Ottenere style

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

// Nativo
// NOTA: Bug conosciuto, restituirà 'auto' se il valore di style è 'auto'
const win = el.ownerDocument.defaultView;
// null significa che non restituirà lo pseudo style
win.getComputedStyle(el, null).color;
```

- **Settare style**

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

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

- **Ottenere/Settare Styles**

Nota che se volete settare styles multipli in una sola volta, potete riferire [setStyles](#) metodo in oui-dom-utils package.

- **Aggiungere classe**

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

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

- **Rimouvere class**

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

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

- **has class**

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

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

- **Toggle class**

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

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

- [2.2](#) Width & Height

Width e Height sono teoricamente identici, prendendo Height come esempio:

- Window height

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

// senza scrollbar, si comporta come jQuery
window.document.documentElement.clientHeight;

// con scrollbar
window.innerHeight;
```

- Document height

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

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

- Element height

```
// 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;
}

// preciso a intero (quando `border-box`, e' `height`; quando `content-
box`, e' `height + padding + border`)
el.clientHeight;

// preciso a decimale (quando `border-box`, e' `height`; quando
`content-box`, e' `height + padding + border`)
el.getBoundingClientRect().height;
```

- [2.3](#) Position & Offset

- Position

```
// 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](#) Scroll Top

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

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

[↑ back to top](#)

Manipolazione DOM

- [3.1](#) Remove

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

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

- [3.2](#) Text

- Get text

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

// Nativo
el.textContent;
```

- Set text


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

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

- [3.3](#) HTML

- Ottenere HTML

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

// Nativo
el.innerHTML;
```

- Settare HTML

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

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

- [3.4](#) Append

appendere elemento figlio dopo l'ultimo elemento figlio del genitore

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

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

- [3.5](#) Prepend

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

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

- [3.6](#) insertBefore

Inserire un nuovo node dopo l'elemento selezionato

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

```
// Nativo
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);

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

- [3.8](#) is

Restituisce `true` se combacia con l'elemento selezionato

```
// jQuery - Notare `is` funziona anche con `function` o `elements` non di
importanza qui
$el.is(selector);

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

[↑ back to top](#)

Ajax

Sostituire con [fetch](#) and [fetch-jsonp](#)

[↑ back to top](#)

Eventi

Per una completa sostituzione con namespace e delegation, riferire a <https://github.com/oneuijs/oui-dom-events>

- [5.1](#) Bind un evento con on

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

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

- [5.2](#) Unbind an event with off

```
// jQuery
$.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);
```

[↑ back to top](#)

Utilities

- [6.1 isArray](#)

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

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

- [6.2 Trim](#)

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

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

- [6.3 Object Assign](#)

Extend, usa object.assign polyfill <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);
```

[↑ back to top](#)




Alternative

- [Forse non hai bisogno di jQuery](#) - Esempi di come creare eventi comuni, elementi, ajax etc usando puramente javascript.
- [npm-dom](#) e [webmodules](#) - Organizzazione dove puoi trovare moduli per il DOM individuale su NPM

Traduzioni

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

Supporto Browsers

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

Licenza

MIT