

- Used by a large percentage (> 70% as of , 2020) of public facing web sites.
- Raison d'etre was hiding browser incompatibilities re. DOM and event model.
- Less necessary now that browsers and JavaScript implementations are more standards compliant.
- Newer applications using newer frameworks will typically not use jQuery.
- Nevertheless, it remains a very popular legacy framework.

# Replaces DOM Tedium

From **jqia**:

```
let checkedValue;
let elements =
  document.getElementsByTagName('input');
for (let i = 0; i < elements.length; i++) {
  if (elements[i].type === 'radio' &&
      elements[i].name === 'some-radio-group' &&
      elements[i].checked) {
    checkedValue = elements[i].value;
    break;
  }
}

let checkedValue =
jQuery('input:radio[name="some-radio-group"]:checked').val()
```

- jQuery is a function which takes up to 2 arguments with different behavior based on form and type of arguments.
- jQuery often aliased as \$.

# On Load Handler

Need to wait for page to load before operating on DOM.  
`window.onload()` handler provided by DOM:

```
window.onload = function() { //waits for page display,  
    ... //and loading external assets  
} //only single onload() handler
```

replaced by jQuery's `$(document).ready()` handler:

```
$(function() { //waits for page DOM to be built;  
    ... //external assets may still be loading  
}); //can have multiple ready handlers.
```

Most jQuery functions return the object they operated on which allows method chaining. Hence

```
obj.fn1().fn2();
```

instead of:

```
obj.fn1();  
obj.fn2();
```

- `$('p.chemical')` returns all p elements having class chemical.
- `$('#h2o.chemical')` returns all elements having id h2o and class chemical.
- `$('p .chemical')` returns all elements with class chemical which are descendants of p elements.
- `$('p > .chemical')` returns all elements with class chemical which are children of p elements.
- `$('p + .chemical')` returns all elements with class chemical which are immediately preceded by a p sibling element.
- `$('p ~ .chemical')` returns all elements with class chemical which are preceded by a p sibling element.

# Attribute Selectors

- `$('p[data-val]')` matches elements of type `p` which have a `data-val` attribute with any value.
- `$('.chemical[data-atomic-wt="92"]')` matches elements having class `chemical` having a `data-atomic-wt` attribute having value `"92"`.
- `$('.chemical[data-name^="A"]')` matches elements having class `chemical` having a `data-name` attribute having value starting with `A`.
- `$('.chemical[data-name$="ium"]')` matches elements having class `chemical` having a `data-name` attribute having value ending with `ium`.
- `$('.chemical[data-atomic-wt!="92"]')` matches elements having class `chemical` having a `data-atomic-wt` attribute having value not equal to `"92"` or lacks `data-atomic-wt` attribute completely.

# Position and Child Filters

Position filters: `:first`, `:last`, `:even`, `:odd`, `:eq(n)`, `:gt(n)`, `:lt(n)`. Indexes start at 0.

- `$('p:first')` selects first paragraph.
- `$('li:odd')` selects all li elements at odd indexes.
- `$('li:gt(4):odd')` selects all li elements at odd indexes greater than 4.
- `$('ul:first-child')` selects first child of ul.
- `$('ul:last-child')` selects last child of ul.
- `$('ul:nth-child(3)')` selects third child of ul (index starts at 1).
- `$('ul:nth-child(3n+1)')` selects first, fourth, ... child of ul.



`$('input[name="widget"]:checked')` selects input controls having name `widget` with state checked.

`$('input:disabled')` selects input controls which are disabled.

`$('input:focus')` selects input controls which are focused.

`$('#:selected')` selects option elements which are selected.

# Accessing and Updating Elements

- `addClass(names)` / `removeClass(names)` / `toggleClass(names)` will add/remove/toggle classes given by spaced-delimited names.
- `css(name)` / `css(name, value)` will return value css property name of first matched / update css property name of all matched to value.
- `html()` / `html(value)` will access matched elements inner HTML / update inner HTML of matched elements to value.
- `val()` / `val(value)` will return value of first matched form control / update all matched form control values to value.
- `attr(name)` / `attr(name, value)` will return value of attribute name of first matched element / update attribute name of all matched elements to value. If value is a function return value is new value; it is called with index and current value with `this` set to element.

# Accessing and Updating Elements Continued

- `append(content)` / `prepend(content)` will append / prepend argument content to content of all matched elements. If content is a function return value is added content; it is called with index and existing element context with `this` set to element.
- `before(content)` / `after(content)` will insert content as sibling before / after all matched elements. The content argument has the same semantics as for `append()`.
- `remove()` will remove all matched elements from DOM. Can be called with optional parameter selector as `remove(selector)` which allows filtering on elements to be removed.
- `detach()` is like `remove()` except that it retains events and data associated with elements so that the elements can be inserted back. Preferred way to make changes in background.

`on(eventType, handler)`

**eventType** Includes blur, change, click, focus, keydown, keyup, keypress, mouse{down,enter,leave,move,out,over,up}, select, submit.

**handler** Function which handles event. First argument is the event which is a cleaned up version of the DOM event which provides event properties like the (x,y) position, target, etc. Has methods like `preventDefault()`, `stopPropagation()`.  
this accesses the DOM element to which the handler was bound.

`trigger(eventType)` Trigger event `eventType` on matched elements.

`off(eventType)` Remove all event handlers for `eventType` from matched elements.

jquery-play

**Example** which illustrates use of input fields with a jQuery based script.

Example illustrates following concepts:

- Data-driven programming.
- The use of Immediately Invoked Function Expressions (IIFEs) for encapsulation.
- The use of jQuery.

## *jQuery Web Site*

[jqia] Bear Bibeault, Yehuda Katz, and Aurelio De Rosa, *jQuery in Action*, Third Edition, Manning, 2015.