.prop() vs .attr()

So jQuery 1.6 has the new function `prop()` .

```
$(selector).click(function(){
    //instead of:
    this.getAttribute('style');
    //do i use:
    $(this).prop('style');
    //or:
    $(this).attr('style');
})
```

or in this case do they do the same thing?

And if I *do* have to switch to using `prop()` , all the old `attr()` calls will break if i switch to 1.6?

UPDATE

```
selector = '#id'

$(selector).click(function() {
    //instead of:
    var getAtt = this.getAttribute('style');
    //do i use:
    var thisProp = $(this).prop('style');
    //or:
    var thisAttr = $(this).attr('style');

    console.log(getAtt, thisProp, thisAttr);
});

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.6.0/jquery.min.js"></script>
<div id='id' style="color: red;background: orange;">test</div>
```

Run code snippet Expand snippet

(see also this fiddle: http://jsfiddle.net/maniator/JpUF2/)

The console logs the `getAttribute` as a string, and the `attr` as a string, but the `prop` as a `CSSStyleDeclaration` , Why? And how does that affect my coding in the future?

javascript jquery dom attr prop

edited Dec 25 at 12:11
user2314737
7,373 8 31 54

asked May 3 '11 at 19:33
Neal
94.3k 24 174 244

14 This will be of definite interest: books.google.ca/... – user1385191 May 3 '11 at 19:38

19 @Neal, it's because this change transcends jQuery. The difference between HTML attributes and DOM properties is massive. – user1385191 May 3 '11 at 19:43

6 It makes me sad to see jQuery's reverted the changes. They're heading in the wrong direction. – user1385191 May 13 '11 at 17:35

3 @Neal. Yes, and it just complicates the problem further instead of separating the two methods. – user1385191 May 13 '11 at 17:37

5 @BritishDeveloper the answer is more complicated than simply stating always use x or y because it depends on what you intend to get. Do you want the attribute, or do you want the property? they are two very different things. – Kevin B Jan 3 '14 at 19:27

16 Answers

Update 1 November 2012

My original answer applies specifically to jQuery 1.6. My advice remains the same but jQuery 1.6.1 changed things slightly: in the face of the predicted pile of broken websites, the jQuery team reverted `attr()` to something close to (but not exactly the same as) its old behaviour for Boolean attributes. John Resig also blogged about it. I can see the difficulty they were in but still disagree with the recommendation to prefer `attr()`.

Original answer

If you've only ever used jQuery and not the DOM directly, this could be a confusing change, although it is definitely an improvement conceptually. Not so good for the bazillions of sites using jQuery that will break as a result of this change though.

I'll summarize the main issues:

- You usually want `prop()` rather than `attr()`.
- In the majority of cases, `prop()` does what `attr()` used to do. Replacing calls to `attr()` with `prop()` in your code will generally work.
- Properties are generally simpler to deal with than attributes. An attribute value may only be a string whereas a property can be of any type. For example, the `checked` property is a Boolean, the `style` property is an object with individual properties for each style, the `size` property is a number.
- Where both a property and an attribute with the same name exists, usually updating one will update the other, but this is not the case for certain attributes of inputs, such as `value` and `checked`: for these attributes, the property always represents the current state while the attribute (except in old versions of IE) corresponds to the default value/checkedness of the input (reflected in the `defaultValue` / `defaultChecked` property).
- This change removes some of the layer of magic jQuery stuck in front of attributes and properties, meaning jQuery developers will have to learn a bit about the difference between properties and attributes. This is a good thing.

If you're a jQuery developer and are confused by this whole business about properties and attributes, you need to take a step back and learn a little about it, since jQuery is no longer trying so hard to shield you from this stuff. For the authoritative but somewhat dry word on the subject, there's the specs: DOM4, HTML DOM, DOM Level 2, DOM Level 3. Mozilla's DOM documentation is valid for most modern browsers and is easier to read than the specs, so you may find their DOM reference helpful. There's a section on element properties.

As an example of how properties are simpler to deal with than attributes, consider a checkbox that is initially checked. Here are two possible pieces of valid HTML to do this:

```
<input id="cb" type="checkbox" checked>
<input id="cb" type="checkbox" checked="checked">
```

So, how do you find out if the checkbox is checked with jQuery? Look on Stack Overflow and you'll commonly find the following suggestions:

- `if ( $("#cb").attr("checked") === true ) {...}`
- `if ( $("#cb").attr("checked") == "checked" ) {...}`
- `if ( $("#cb").is(":checked") ) {...}`

This is actually the simplest thing in the world to do with the `checked` Boolean property, which has existed and worked flawlessly in every major scriptable browser since 1995:

```
if (document.getElementById("cb").checked) {...}
```

The property also makes checking or unchecking the checkbox trivial:

```
document.getElementById("cb").checked = false
```

In jQuery 1.6, this unambiguously becomes

```
$("#cb").prop("checked", false)
```

The idea of using the `checked` attribute for scripting a checkbox is unhelpful and unnecessary. The property is what you need.

- It's not obvious what the correct way to check or uncheck the checkbox is using the `checked` attribute
- The attribute value reflects the default rather than the current visible state (except in some older versions of IE, thus making things still harder). The attribute tells you nothing about the whether the checkbox on the page is checked. See http://jsfiddle.net/VktA6/49/.

edited Aug 13 '15 at 23:17
Nathaniel Ford
9,347 13 40 65

answered May 3 '11 at 23:06
Tim Down
198k 42 314 400

8

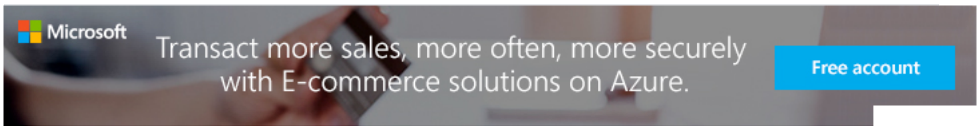
@Neal: If you want to know what properties DOM elements have and how attributes may seed their values, keep these links to hand: The DOM2 HTML specification, the DOM2 spec, and DOM3 spec. The indexes in each case are excellent and link you straight to a thorough description of the property, where its value comes from, etc. – T.J. Crowder May 4 '11 at 13:30

3 @Neal: Re what makes it different: The source and nature of the information. Look at Tim's `value` example, for instance. A function called `attr` that retrieves the *property* `value` rather than the attribute "value" doesn't make much sense (and they can be different). Going forward, `attr` doing attributes and `prop` doing properties will be clearer, though the transition will be painful for some. Don't take this the wrong way, there's nothing like stepping back and reading the specs to get an idea of what properties are. – T.J. Crowder May 4 '11 at 13:46

37 @Neal: A DOM element is an object. Properties are properties of that object, just like any other programming object. Some of those props get their initial values from the attributes in the markup, which are also stored on the DOM object in a *separate* map of attributes. In most cases, writing to a prop only changes the prop, although sadly there are some props that write any changes through to the underlying attr (`value` for instance), but let's try to mostly ignore that. 99% of the time, you want to work with props. If you need to work with an actual attribute, you'll probably know that. – T.J. Crowder May 4 '11 at 13:57

3 @Tim: *"Changing the `value` property of an input doesn't change its `value` attribute in modern browsers"* I didn't think it did, which is why I started to use it as an example, but when I started coding up the example, darned if it didn't get updated on Chrome, Firefox, Opera, IE... - jsbin.com/ahire4 Turns out that's because I was using an `input[type="button"]` for my experiment. It *doesn't* update the attribute on a `input[type="text"]` - jsbin.com/ahire4/2 Talk about convoluted!! Not just the element, but the *type* of it... – T.J. Crowder May 4 '11 at 14:35

5 `$('#chk').checked` **never** worked... That is not proper jQuery at all... – Neal Feb 11 '13 at 14:32


Microsoft
Transact more sales, more often, more securely with E-commerce solutions on Azure.
Free account

I think Tim said it quite well, but let's step back:

A DOM element is an object, a thing in memory. Like most objects in OOP, it has *properties*. It also, separately, has a map of the attributes defined on the element (usually coming from the markup that the browser read to create the element). Some of the element's *properties* get their *initial* values from *attributes* with the same or similar names (`value` gets its initial value from the "value" attribute; `href` gets its initial value from the "href" attribute, but it's not exactly the same value; `className` from the "class" attribute). Other properties get their initial values in other ways: For instance, the `parentNode` property gets its value based on what its parent element is; an element always has a `style` property, whether it has a "style" attribute or not.

Let's consider this anchor in a page at `http://example.com/testing.html` :

```
<a href='foo.html' class='test one' name='fooAnchor' id='fooAnchor'>Hi</a>
```

Some gratuitous ASCII art (and leaving out a lot of stuff):

```
+-------------------------------------------+
| a                                         |
+-------------------------------------------+
| href:       "http://example.com/foo.html" |
| name:       "fooAnchor"                   |
| id:         "fooAnchor"                   |
| className:  "test one"                    |
| attributes:                               |
|    href:  "foo.html"                      |
|    name:  "fooAnchor"                     |
|    id:    "fooAnchor"                     |
|    class: "test one"                      |
+-------------------------------------------+
```

Note that the properties and attributes are distinct.

Now, although they are distinct, because all of this evolved rather than being designed from the ground up, a number of properties write back to the attribute they derived from if you set them. But not all do, and as you can see from `href` above, the mapping is not always a straight "pass the value on", sometimes there's interpretation involved.

When I talk about properties being properties of an object, I'm not speaking in the abstract. Here's some non-jQuery code:

```
var link = document.getElementById('fooAnchor');
alert(link.href);                 // alerts "http://example.com/foo.html"
alert(link.getAttribute("href")); // alerts "foo.html"
```

(Those values are as per most browsers; there's some variation.)

The `link` object is a real thing, and you can see there's a real distinction between accessing a *property* on it, and accessing an *attribute*.

As Tim said, the **vast majority** of the time, we want to be working with properties. Partially that's because their values (even their names) tend to be more consistent across browsers. We mostly only want to work with attributes when there is no property related to it (custom attributes), or when we know that for that particular attribute, the attribute and the property are not 1:1 (as with `href` and "href" above).

The standard properties are laid out in the various DOM specs:

- DOM2 HTML
- DOM2 Core
- DOM3 Core

These specs have excellent indexes and I recommend keeping links to them handy; I use them all the time.

Custom attributes would include, for instance, any `data-xyz` attributes you might put on elements to provide meta-data to your code (now that that's valid as of HTML5, as long as you stick to the `data-` prefix). (Recent versions of jQuery give you access to `data-xyz` elements via the `data` function, but that function does more than that and if you're just dealing with a `data-xyz` attribute, I'd actually use the `attr` function to interact with it.)

The `attr` function used to have some convoluted logic around getting what they thought you wanted, rather than literally getting the attribute. It conflated the concepts. Moving to `prop` and `attr` is meant to de-conflate them. There will be some brief confusion, but hopefully a better understanding of what's really going on going forward.

Some time kicking around the specs above, and experimenting, should help clear some of this up.

Update: jQuery 1.6.1 changes the `attr` function again, in ways that the jQuery team say mean most code that currently uses `attr` will continue to work, even if `prop` would technically be preferred. Details in the jQuery 1.6.1 blog post.

edited Dec 22 '11 at 17:22
RaphaelDDL
3,482 1 20 38

answered May 4 '11 at 14:27
T.J. Crowder
502k 84 813 953

27 +1, well explained. I didn't quite have the patience to step back and explain all that for an SO answer. Maybe time for a blog post... – Tim Down May 4 '11 at 14:35

7 Both you and Tim did a great job on elaborating on the intricacies of attributes/properties. I'll definitely be reading a lot more specs from now on. – user1385191 May 4 '11 at 19:34

Wow. that new 1.6.1 update really nullifies this question a bit.. (but not much) but that link basically answers it now – Neal May 11 '11 at 4:52

It didn't nullify it for me, I just fixed a bug using jquery1.7 where I was setting .attr('class', 'bla') and it wasn't working where .prop('className', 'bla') worked – PandaWood Mar 22 '12 at 2:08

1 @T.J.Crowder re: `.data` - it'll *read* the default value of the attribute, if present, but if you write to it, it'll change a hidden *property* of the element, and not update the attribute. – Alnitak Jul 23 '12 at 14:48

This change has been a long time coming for jQuery. For years, they've been content with a function named `attr()` that mostly retrieved DOM properties, not the result you'd expect from the name. The segregation of `attr()` and `prop()` should help alleviate some of the confusion between HTML attributes and DOM properties. `$.fn.prop()` grabs the specified DOM property, while `$.fn.attr()` grabs the specified HTML attribute.

To fully understand how they work, here's an extended explanation on the difference between HTML attributes and DOM properties.:

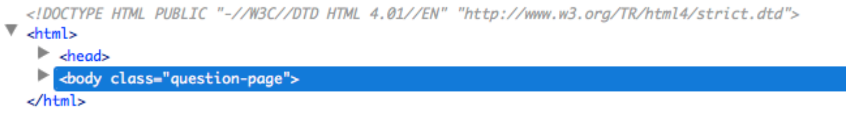
HTML Attributes

Syntax:

```
<body onload="foo()">
```

Purpose: Allows markup to have data associated with it for events, rendering, and other purposes.

Visualization:



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
▼ <html>
    ▶ <head>
    ▶ <body class="question-page">
  </html>
```

The class attribute is shown here on the body. It's accessible through the following code:

```
var attr;
attr = document.body.getAttribute("class");
//IE 8 Quirks and below
attr = document.body.getAttribute("className");
```

Attributes are returned in string form and can be inconsistent from browser to browser. However, they can be vital in some situations. As exemplified above, IE 8 Quirks Mode (and below) expects the name of a DOM property in get/set/removeAttribute instead of the attribute name. This is one of many reasons why it's important to know the difference.

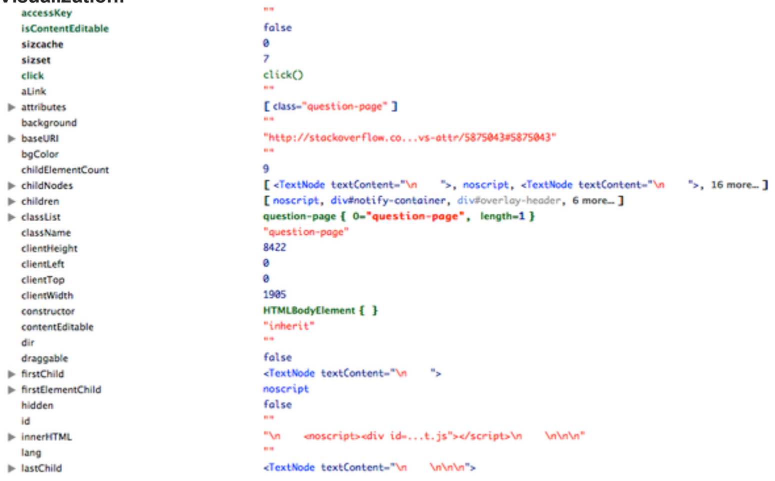
DOM Properties

Syntax:

```
document.body.onload = foo;
```

Purpose: Gives access to properties that belong to element nodes. These properties are similar to attributes, but are only accessible through JavaScript. This is an important difference that helps clarify the role of DOM properties. **Please note that attributes are completely different from properties**, as this event handler assignment is useless and won't receive the event (body doesn't have an onload event, only an onload attribute).

Visualization:



```
  accessKey               ""
  isContentEditable       false
  sizcache                0
  sizset                  7
  click                   click()
  aLink                   ""
▶ attributes              [ class="question-page" ]
  background              ""
▶ baseURI                 "http://stackoverflow.co...vs-attr/5875043#5875043"
  bgColor                 ""
  childElementCount       9
▶ childNodes              [ <TextNode textContent="\n    ">, noscript, <TextNode textContent="\n    ">, 16 more… ]
▶ children                [ noscript, div#notify-container, div#overlay-header, 6 more… ]
▶ classList               question-page { 0="question-page", length=1 }
  className               "question-page"
  clientHeight            8422
  clientLeft              0
  clientTop               0
  clientWidth             1905
  constructor             HTMLBodyElement {  }
  contentEditable         "inherit"
  dir                     ""
  draggable               false
▶ firstChild              <TextNode textContent="\n    ">
▶ firstElementChild       noscript
  hidden                  false
  id                      ""
▶ innerHTML               "\n    <noscript><div id=...t.js"></script>\n    \n\n\n"
  lang                    ""
▶ lastChild               <TextNode textContent="\n    \n\n\n">
```

Here, you'll notice a list of properties under the "DOM" tab in Firebug. These are DOM properties. You'll immediately notice quite a few of them, as you'll have used them before without knowing it. Their values are what you'll be receiving through JavaScript.

Documentation

- JavaScript: The Definitive Guide by David Flanagan
- HTML Attributes, Mozilla Dev Center
- DOM Element Properties, Mozilla Dev Center

Example

HTML: `<textarea id="test" value="foo"></textarea>`

JavaScript: `alert($('#test').attr('value'));`

In earlier versions of jQuery, this returns an empty string. In 1.6, it returns the proper value, `foo`.

Without having glanced at the new code for either function, I can say with confidence that the confusion has more to do with the difference between HTML attributes and DOM properties, than with the code itself. Hopefully, this cleared some things up for you.

-Matt

edited Dec 25 '14 at 22:24
Boann
28.1k 8 62 98

answered May 3 '11 at 20:05
user1385191

2 @Matt this explains nothing about `prop()` in jQuery.... – Neal May 3 '11 at 21:02

6 `$.prop()` gets DOM properties, `$.attr()` gets HTML attributes. I'm trying to bridge the gap psychologically so you can understand the difference between the two. – user1385191 May 3 '11 at 21:06

7 Boy, I am confused now too. So `$('#test').prop('value')` does not return anything? Nor does `.attr('checked')` for a checkbox? But it used to? Now you'd have to change it to `prop('checked')`? I don't understand the need for this distinction - why is it important to differentiate between HTML attributes and DOM properties? What is the common use-case that made this change *"a long time coming"*? What is *wrong* with abstracting the distinction between the two away, since it seems like their use-cases mostly overlap? – BlueRaja - Danny Pflughoeft May 3 '11 at 22:11

3 @BlueRaja: because there is a serious underlying difference between the two concepts which, if you brush it under the carpet like jQuery used to, results in unexpected failures. `value` is one of the most obvious cases, since the `value` property will give you the current value of a field, but the `value` attribute will give you the original value that was declared in the `value="..."` attribute, which is actually the `defaultValue` property. (Though this particular case gets confused again by bugs in IE<9.) – bobince May 3 '11 at 23:04

4 @BlueRaja: The answer to that is even more complicated. :-) Setting `attr('value', ...)` in jQuery <1.6 sets the property, so the current value changes and the default value doesn't. In 1.6 it sets the attribute, so in theory the default value changes and the current value doesn't. However, there are (more) browser inconsistencies over what exactly setting the `value` attribute does. In IE it sets the current value as well; in some browsers it only sets the current value if the current value has not already been set before. [cries] – bobince May 6 '11 at 19:16

A property is in the DOM; an attribute is in the HTML that is parsed into the DOM.

Further detail

If you change an attribute, the change will be reflected in the DOM (sometimes with a different name).

Example: Changing the `class` attribute of a tag will change the `className` property of that tag in the DOM. If you have no attribute on a tag, you still have the corresponding DOM property with an empty or a default value.

Example: While your tag has no `class` attribute, the DOM property `className` does exist with a empty string value.

edit

If you change the one, the other will be changed by a controller, and vice versa. This controller is not in jQuery, but in the browser's native code.

edited Nov 15 at 7:09
jwpfox
2,201 5 23 29

answered Sep 27 '11 at 16:55
HerrSerker
13.8k 5 39 69

13 @yunzen it is a clear, concise, and simple **correct** answer. You definitely deserve it :-) – Neal Nov 5 '12 at 16:51

1 @Luke the DOM is the inner technical representation, the model. The HTML attributes are an outer representation, a view. If you change the one, the other will be changed by a controller, and vice versa. – HerrSerker Aug 8 '13 at 5:50

1 @Luke Look at this: jsfiddle.net/Pms3W – HerrSerker Aug 8 '13 at 7:30

1 "If you change the one, the other will be changed by a controller, and vice versa. This controller is not in jQuery, but in the browsers' native code." This is **not** true for most attributes/properties. For the majority it's only a one-way translation where an attribute determines the *initial* value of the corresponding property. The top two answers explain this in detail. – VINCENT Apr 18 '15 at 19:16

1 This is real correct answer. Nothing complicated. I don't need a master's thesis or care how it's coded in the background. Just what is the difference, int a nutshell, and this answer explains that. – user3621633 Sep 8 '15 at 20:20

It's just the distinction between HTML attributes and DOM objects that causes a confusion. For those that are comfortable acting on the DOM elements native properties such a `this.src` `this.value` `this.checked` etc, `.prop` is a very warm welcome to the family. For others, it's just an added layer of confusion. Let's clear that up.

The easiest way to see the difference between `.attr` and `.prop` is the following example:

```
<input blah="hello">
```

1. **`$('input').attr('blah')`** : returns `'hello'` as expected. No suprises here.
2. **`$('input').prop('blah')`** : returns `undefined` -- because it's trying to do `[HTMLInputElement].blah` -- and no such property on that DOM object exists. It only exists in the scope as an attribute of that element i.e. `[HTMLInputElement].getAttribute('blah')`

Now we change a few things like so:

```
$('input').attr('blah', 'apple');
$('input').prop('blah', 'pear');
```

1. **`$('input').attr('blah')`** : returns `'apple'` eh? Why not "pear" as this was set last on that element. Because the property was changed on the input attribute, not the DOM input element itself -- they basically almost work independently of each other.
2. **`$('input').prop('blah')`** : returns `'pear'`

The thing you really need to be careful with is just **do not mix the usage of these for the same property throughout your application** for the above reason.

See a fiddle demonstrating the difference: http://jsfiddle.net/garreh/uLQXc/

`.attr` vs `.prop` :

Round 1: style

```
<input style="font:arial;"/>
```

- `.attr('style')` -- returns inline styles for the matched element i.e. `"font:arial;"`
- `.prop('style')` -- returns an style declaration object i.e. `CSSStyleDeclaration`

Round 2: value

```
<input value="hello" type="text"/>
```

```
$('input').prop('value', 'i changed the value');
```

- `.attr('value')` -- returns `'hello'` *
- `.prop('value')` -- returns `'i changed the value'`

* Note: jQuery for this reason has a `.val()` method, which internally is equivalent to `.prop('value')`

edited May 19 '11 at 14:19 answered May 19 '11 at 10:18
Gary Green
13.4k 6 32 61

@Neal becaue it gives reference to the structure of the jquery functions better. "$('input').prop('blah'): returns undefined -- because it's trying to do [HTMLInputElement].blah -- and no such property on that DOM object exists. It only exists in the scope as an attribute of that element i.e. [HTMLInputElement].getAttribute('blah')" – Uğur Gümüşhan Jul 25 '12 at 5:24

2 Seems different from the doc, api.jquery.com/prop: "The .prop() method should be used to set disabled and checked instead of the .attr() method. The .val() method should be used for getting and setting value." – swan Dec 11 '12 at 10:59

I do not understand why when class or style property is changed attribute is changed as well jsfiddle.net/featon/Jbw6m/3 ? – Damian May 11 '13 at 16:45

TL;DR

Use `prop()` over `attr()` in the majority of cases.

A *property* is the current state of the input element. An *attribute* is the default value.

A property can contain things of different types. An attribute can only contain strings

edited Mar 9 '15 at 16:21
isherwood
31k 6 58 90

answered Jul 16 '14 at 9:45
agjmills
491 5 10

1 if possible to come with few easy sample of what u r saying and also show when to use `prop() and when to go for attr()`. waiting for answer :) – Mou May 28 '15 at 12:31

dirty checkedness is an example where the difference is **observable**.

To see it, run the following snippet and:

- click the button. Both checkboxes got checked.
- uncheck both checkboxes.
- click the button again. Only the `prop` checkbox got checked. BANG!

```
$('button').on('click', function() {
  $('#attr').attr('checked', 'checked')
  $('#prop').prop('checked', true)
})
```

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.1/jquery.min.js"></script>
<label>attr <input id="attr" type="checkbox"></label>
<label>prop <input id="prop" type="checkbox"></label>
<button type="button">Set checked attr and prop.</button>
```

Run code snippet Expand snippet

For some attributes like `disabled` on `button`, adding or removing the content attribute `disabled="disabled"` always toggles the property (called IDL attribute in HTML5) because http://www.w3.org/TR/html5/forms.html#attr-fe-disabled says:

> The disabled IDL attribute must reflect the disabled content attribute.

so you might get away with it, although it is ugly since it modifies HTML without need.

For other attributes like `checked="checked"` on `input type="checkbox"`, things break, because once you click on it, it becomes dirty, and then adding or removing the `checked="checked"` content attribute **does not toggle checkedness anymore**.

This is why you should use mostly `.prop`, as it affects the effective property directly, instead of relying on complex side-effects.

edited Mar 16 '15 at 10:04 answered Jul 6 '14 at 11:43
Ciro Santilli 烏坎事件 2016六四事件 法轮功
59.8k 11 247 189

For completeness also try these variations: 1) Before clicking the button check and then uncheck the first checkbox 2) (For this one you will need to change the snippet) Give the first input a `checked` attribute: `checked="checked"` – VINCENT Apr 18 '15 at 19:23

All is in the doc :

> The difference between attributes and properties can be important in specific situations. Before jQuery 1.6, the .attr() method sometimes took property values into account when retrieving some attributes, which could cause inconsistent behavior. As of jQuery 1.6, the .prop() method provides a way to explicitly retrieve property values, while .attr() only retrieves attributes.

So use prop !

answered May 3 '11 at 19:35
Arnaud F.
4,544 5 27 69

1 So that mean when i switch to 1.6, alot of things will break?? – Neal May 3 '11 at 19:36

2 +1 for RTFM. A quick note about attributes versus properties would be a good addition though. – Jason McCreary May 3 '11 at 19:37

5 I seem to recall . `$.attr()` retrieving properties most of the time I worked with it, not "sometimes". The confusion caused by it is still resonant today. Sadly, I'm having a lot of trouble finding a *definitive* read on HTML attributes vs. DOM properties, so I might write an answer here in a bit. – user1385191 May 3 '11 at 19:39

2 @Arnaud-f Not true. All code previous to 1.6 that uses attr('checked') to check checkboxes will now be broken. – CaptSaltyJack May 3 '11 at 20:39

2 @Matt McDonald: What kind of "...trouble finding a *definitive* read on HTML attributes vs. DOM properties..." do you mean? Properties (reflected and otherwise) are described in the DOM2 HTML specification; you may also need to refer to the DOM2 and DOM3 specs. – T.J. Crowder May 4 '11 at 13:23

---

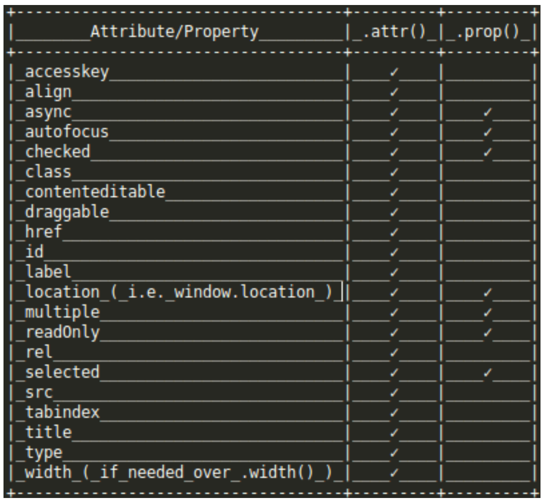
attributes are in your HTML *text document/file* (== imagine this is the result of your html markup parsed), whereas
properties are in HTML *DOM tree* (== basically an actual property of some object in JS sense).

Importantly, many of them are synced (if you update `class` property, `class` attribute in html will also be updated; and otherwise). **But** some attributes may be synced to unexpected properties - eg, **attribute** `checked` corresponds to **property** `defaultChecked` , so that

- manually checking a checkbox will change `.prop('checked')` value, but will not change `.attr('checked')` and `.prop('defaultChecked')` values
- setting `$('#input').prop('defaultChecked', true)` will also change `.attr('checked')` , but this will not be visible on an element.

> **Rule of thumb is**: `.prop()` method should be used for boolean attributes/properties and for properties which do not exist in html (such as window.location). All other attributes (ones you can see in the html) can and should continue to be manipulated with the `.attr()` method. (http://blog.jquery.com/2011/05/10/jquery-1-6-1-rc-1-released/)

And here is a table that shows where `.prop()` is preferred (even though `.attr()` can still be used).



Attribute/Property	.attr()	.prop()
accesskey	✓	
align	✓	
async	✓	✓
autofocus	✓	✓
checked	✓	✓
class	✓	
contenteditable	✓	
draggable	✓	
href	✓	
id	✓	
label	✓	
location (i.e. window.location)	✓	✓
multiple	✓	✓
readOnly	✓	✓
rel	✓	
selected	✓	✓
src	✓	
tabindex	✓	
title	✓	
type	✓	
width (if needed over .width())	✓	

Why would you sometimes want to use .prop() instead of .attr() where latter is officially adviced?

1. `.prop()` can return any type - string, integer, boolean; while `.attr()` always returns a string.
2. `.prop()` is said to be about 2.5 times faster than `.attr()` .

edited Jun 12 '15 at 7:09 answered Jun 12 '15 at 5:56
lakesare
2,681 1 15 27

something has been changed, like these: `$('#myImageId').prop('src', 'http://example.com/img.png')` , `var class = $('.myDivClass').prop('class')` , or `$('#myTextBox').prop('type', 'button')` . And so on... – Kevin Nov 16 '15 at 15:32

@Mr.Wolf, sorry? – lakesare Nov 16 '15 at 16:06

@Mr.Wolf, sorry, I stil don't get what you mean. what has 'changed'? syntax has always been like like that. – lakesare Nov 21 '15 at 10:52

I think the table in your answer is unnecessary. Because `.prop()` works fine with all of the properties. You don't wanna mark all properties in `.prop()` column, do you? – Kevin Nov 22 '15 at 5:29

@Mr.Wolf, I think it is necessary, because, as already stated, it 'shows where `.prop()` is preferred (even though `.attr()` can still be used)' – lakesare Nov 22 '15 at 5:43

Usually you'll want to use properties. Use attributes only for:

1. Getting a custom HTML attribute (since it's not synced with a DOM property).
2. Getting a HTML attribute that doesn't sync with a DOM property, e.g. get the "original value" of a standard HTML attribute, like `<input value="abc">`.

answered Feb 2 '14 at 20:36
naor
639 9 9

.attr() :-

- Get the value of an **attribute** for the first element in the set of matched elements.
- gives you the value of element as it was defines in the html on page load

.prop() :-

- Get the value of a **property** for the first element in the set of matched elements.
- gives the updated values of elements which is modified via javascript/jquery

answered Dec 8 '15 at 9:14
Kgn-web
817 1 8 19

`attributes` -> HTML

`properties` -> DOM

answered Dec 26 '14 at 13:28
NkS
556 6 22

6 I know what you want to say, but HTML is a markup language, and the DOM a representation created out of the HTML. A DOMElement has both *attributes* and *properties*. – t.niese Dec 26 '14 at 13:52

@t.niese, `attirbutes` also one of the property of `DOM` – NkS Dec 27 '14 at 5:00

The short answer is simple. – Nabi K.A.Z. Jan 5 at 21:04

Gently reminder about using prop(), example:

```
if ($("#checkbox1").prop('checked')) {
        isDelete = 1;
    } else {
        isDelete = 0;
    }
```

Function above is used to check if checkbox1 is checked or not, if checked: return 1; if not: return 0. Function prop() used here as a GET function.

```
if ($("#checkbox1").prop('checked',true)) {
        isDelete = 1;
    } else {
        isDelete = 0;
    }
```

Function above is used to set checkbox1 to be checked and ALWAYS return 1. Now function prop() used as a SET function.

Don't mess up.

P/S: When I'm checking Image ***src*** property. If the ***src*** is empty, **prop** return the current url of the page (wrong), and **attr** return empty string (right).

edited Oct 20 '15 at 12:45 answered Oct 17 '15 at 16:12
user2657778
45 9

You're wrong in this example: `<img src="smiley.gif" alt="Smiley face" width="42" height="42" onclick="alert($(this).prop('src'))">` . It should work and return the image location. – Kevin Nov 16 '15 at 15:37

If the src is EMPTY – user2657778 Nov 18 '15 at 11:25

Before jQuery 1.6 , the attr() method ***sometimes*** took property values into account when retrieving attributes, this caused rather inconsistent behavior.

The introduction of the prop() method provides a way to explicitly retrieve property values, while .attr() retrieves attributes.

The Docs:

jQuery.attr() Get the value of an attribute for the first element in the set of matched elements.

jQuery.prop() Get the value of a property for the first element in the set of matched elements.

answered Oct 25 '15 at 0:05
PetrusR3x
1,807 1 6 16

> 1) A property is in the DOM; an attribute is in the HTML that is parsed into the DOM.
>
> 2) $(elem).attr("checked") (1.6.1+) "checked" (String) Will change with checkbox state
>
> 3) $(elem).attr("checked") (pre-1.6) true (Boolean) Changed with checkbox state

- Mostly we want to use for DOM object rather then custom attribute like `data-img, data-xyz` .
- Also some of difference when accessing `checkbox` value and `href` with `attr()` and `prop()` as thing change with DOM output with `prop()` as full link from `origin` and `Boolean` value for checkbox `(pre-1.6)`
- We can only access DOM elements with `prop` other then it gives **`undefined`**

Show code snippet

answered Dec 18 '15 at 9:28
Parth Trivedi
2,863 6 30

Gary Hole answer is very relevant to solve the problem if the code is written in such way

```
obj.prop("style","border:1px red solid;")
```

Since the prop function return `CSSStyleDeclaration` object, above code will not working properly in some browser(tested with `IE8 with Chrome Frame Plugin` in my case).

Thus changing it into following code

```
obj.prop("style").cssText = "border:1px red solid;"
```

solved the problem.

edited Oct 15 '15 at 10:30 answered Oct 15 '15 at 8:38
zawhtut
4,821 3 33 57

---

protected by Neal Jan 5 '12 at 15:23

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?