**Part I**

[[edit](http://docs.jquery.com/action/edit/Tutorials:How_to_Get_Anything_You_Want_2?section=2)]

**Introduction**

With jQuery, you can get to just about anything on a web page. In this entry, I’ll show you a few ways you can use jQuery’s versatile selector expressions to traverse to an element — or group of elements — in the [DOM](http://docs.jquery.com/DOM).

I’ve put together a short paragraph and an ordered list scattered with inline elements such as <em>, <code>, and links. It’s all wrapped in a DIV with an ID of "jqdt," which allows me to focus the demonstration to one area of this page.

Each button below the "jqdt" DIV will toggle a yellow background on and off for the parts of the DIV described.

[[edit](http://docs.jquery.com/action/edit/Tutorials:How_to_Get_Anything_You_Want_2?section=3)]

**Demo**

This is a *paragraph* of **text** with class=”goofy.” It has an [external link](http://www.englishrules.com/), some $(code), and a [#dt-link3 same-page link](#dt-link3_same-page_link).

1. list item 1 with dummy link to [silly.pdf](http://docs.jquery.com/action/edit/Silly.pdf)
2. *list* ***item*** *2* with class=”**goofy**“
3. list item 3
4. **list item 4** with silly link to [silly.pdf silly.pdf](http://docs.jquery.com/action/edit/Silly.pdf_silly.pdf)

* $('li:eq(0)') gets the first list item
* $('li:even') gets all odd-numbered list items (because, in javascript, numbering starts with 0, not 1).
* $('li:lt(3)') gets the first 3 list items. “lt” stands for “less than,” and it starts counting at 0, not 1.
* $('li:not(.goofy)') gets list items 1, 2, and 4, because they’re not “goofy.”
* $('p a[href\*=#]') gets any links that are inside a paragraph and have an “href” attribute containing “#” anywhere. There are problems with trying to identify same-page links this way. I’ll write about this in an upcoming entry. Note the space between the “p” and the “a”; that means that the “a” is a *descendant* of the “p.”
* $('code, li.goofy') gets all code elements *and* any list item with a class of “goofy.”
* $('ol .goofy > strong') gets all strong elements that are children of any element with a class of “goofy” as long as the class somewhere inside (i.e. a descendent) of an ordered list. Notice that the word “item” is not highlighted because, even though it’s inside “.goofy,” it’s not a direct child. Only “goofy” is a direct child of “.goofy.” Maybe we should call it “goofy jr.”
* $('li + li > a[href$=pdf]') gets all links ending in “pdf” that are children of any list item that has another list item as its previous sibling. It won’t get the first list item’s silly.pdf because that list item has no other list items before it.
* $('span:hidden') gets any span element that is hidden.

Note: The selectors used for the toggle buttons are identical to the ones shown next to each button, except that they are preceded by $('#jqdt').find to target the highlighting.

jQuery’s [selector expressions](http://docs.jquery.com/Selectors) cover the full range of CSS 1-3, along with some custom, jQuery-only expressions.

In the follow-up to this tutorial, I’ll explore jQuery methods such as .filter(), prev(), and siblings() that complement the above selector expressions to give you full DOM-traversing power!

[[edit](http://docs.jquery.com/action/edit/Tutorials:How_to_Get_Anything_You_Want_2?section=4)]

**Part II**

[[edit](http://docs.jquery.com/action/edit/Tutorials:How_to_Get_Anything_You_Want_2?section=5)]

**Introduction**

Now that we have looked at traversing the DOM with jQuery’s selector expressions to get any elements in the document, I’m going to focus on jQuery methods that provide even more ways to get elements. Some of these methods have a nearly identical counterpart among the selector expressions, but for the most part, the two ways of getting elements complement each other.

Once again, I’ve put together a short paragraph and an ordered list scattered with inline elements such as <em>, <code>, and links. This time it’s wrapped in a DIV with an ID of "jqdt2," which allows me to focus the demonstration to one area of this page.

Each button below the "jqdt2" DIV will toggle a background color on and off for the parts of the DIV described.

[[edit](http://docs.jquery.com/action/edit/Tutorials:How_to_Get_Anything_You_Want_2?section=6)]

**Demo**

This is a *paragraph* of **text** with class=”goofy.” It has an [external link](http://docs.jquery.com/www.englishrules.com), some $(code), and a [same-page link](#dt-link3).

1. list item 1 with dummy link to [silly.pdf](http://docs.jquery.com/dummy.pdf)
2. *list* ***item*** *2* with class=”**goofy**“
3. list item 3 with class=”funny”
4. **list item 4** with silly link to [silly.pdf](http://docs.jquery.com/silly.pdf)
5. list item 5 with class=”**goofy**“

Click a link to toggle the selector's highlighting

* $('#jqdt2').find('li').slice(0,1)

gets the first list item within the element with an id of "jqdt2."

* $('#jqdt2').find('li').slice(0,3)

gets the first 3 list items.

* $('#jqdt2').find('li.funny').siblings()

gets list items 1, 2, 4, and 5 because they're the siblings of "funny."

* $('#jqdt2').find('li').not(':gt(2)').filter(':contains(silly)')

gets all list items that contain the string "silly" as long as they are not greater than the third list item. So, even though list item 4 has the word "silly" in it, it won't be selected because it is greater than the third.

* $('#jqdt2').find('li').filter('[class=goofy]')

first gets all list items inside "jqdt2" and then filters them so that it ultimately only gets the list items that have a class of "goofy."

* $('#jqdt2')  
  .find('li.funny')  
    .css('backgroundColor','#0ff')  
  .end()  
  .find('p > strong')  
    .css('backgroundColor','#ff0')   
  With this example you can see that jQuery's chaining does not have to occur on a single line, but can be extended to multiple lines for better readability. I've also exposed the CSS backgroundColor setting that occurs in all of these examples so that you can see how .end() works. Any time you use one of jQuery's DOM traversing methods, you can back out of it after you have done what you want with it. Here, the first .find() applies CSS to list items with a class of "funny"; the .end() backs us out of that set of jQuery objects and up to the opening $('#jqdt2'); the final .find() method gets a completely separate group of DOM nodes and applies an aqua background color to it. Also, note that I use the DOM property "backgroundColor" rather than the CSS property "background-color." While either one *should* work with jQuery, CSS properties that contain a hyphen will break in some builds. Therefore, I recommend "camel-casing" those properties, just to be safe.
* $('li.goofy')  
  .parents('#jqdt2')  
  .children('p')  
  .next()  
  .find('a')  
  .parent()  
  This final example is a ridiculously circuitous traversal sequence that probably isn't practical in itself, but I wanted to show a couple more of the methods at your disposal. Unlike all the other examples, this one doesn't start by narrowing the possible DOM elements to those within an element with an id of "jqdt2." Instead, it starts with all list items with a class of goofy, and *then* gets any **ancestor** with an id of "jqdt2" (since IDs should be unique, there should be only one). Next, it gets all **children** of "jqdt2" that are paragraphs and moves on to the **next** sibling, which in this case is the ordered list (<ol>). Finally, it looks for links within that ordered list and gets the **parent** of each. So, by the end of all this skipping around the DOM, we should have selected list items 1 and 4.

For a complete list of DOM traversal methods, check out the [Traversing](http://docs.jquery.com/Traversing) section of the API. In addition to the ones covered in this entry, you’ll find:

* .is()
* .prev(): just like next(), but in the opposite direction
* .parents(): As of jQuery 1.1, .ancestors() is no longer supported. Please use .parents() instead.
* .add(): allows you to add another group of jQuery objects
* and many more!

#### jQuery('#id')

* id

An ID to search for, specified via the id attribute of an element.

Description: Selects a single element with the given id attribute.

For id selectors, jQuery uses the JavaScript function document.getElementById(), which is extremely efficient. When another selector is attached to the id selector, such as h2#pageTitle, jQuery performs an additional check before identifying the element as a match.

As always, remember that as a developer, your time is typically the most valuable resource. Do not focus on optimization of selector speed unless it is clear that performance needs to be improved.

Each id value must be used only once within a document. If more than one element has been assigned the same ID, queries that use that ID will only select the first matched element in the DOM. This behavior should not be relied on, however; a document with more than one element using the same ID is invalid.

If the id contains characters like periods or colons you have to [escape those characters with backslashes](#How_do_I_select_an_element_by_an_ID_that_has_characters_used_in_CSS_notation.3F).

### Examples:

#### Example: Finds the element with the id "myDiv".

<!DOCTYPE html>  
<html>  
<head>  
  <style>  
  div {  
    width: 90px;  
    height: 90px;  
    float:left;  
    padding: 5px;  
    margin: 5px;  
    background-color: #EEEEEE;  
  }  
  </style>  
  <script src="http://code.jquery.com/jquery-latest.js"></script>  
</head>  
<body>  
  <div id="notMe"><p>id="notMe"</p></div>  
  
  <div id="myDiv">id="myDiv"</div>  
<script>$("#myDiv").css("border","3px solid red");</script>  
  
</body>  
</html>

#### Demo:

#### Example: Finds the element with the id "myID.entry[1]". See how certain characters must be escaped with backslashes.

<!DOCTYPE html>  
<html>  
<head>  
  <style>  
  div {  
    width: 300px;  
    float:left;  
    padding: 2px;  
    margin: 3px;  
    background-color: #EEEEEE;  
  }  
  </style>  
  <script src="http://code.jquery.com/jquery-latest.js"></script>  
</head>  
<body>  
  <div id="myID.entry[0]">id="myID.entry[0]"</div>  
  
  <div id="myID.entry[1]">id="myID.entry[1]"</div>  
  <div id="myID.entry[2]">id="myID.entry[2]"</div>  
<script>$("#myID\\.entry\\[1\\]").css("border","3px solid red");</script>  
  
</body>  
</html>

#### Demo: