jQuery

* + 1. [**jQuery**](https://jquery.com/)

**2.3.5.1 About**

jQuery is a DOM manipulation library. Although it is a powerful tool, it does add to the "weight" of a project, so be aware of what circumstances exist in which [**You Might Not Need jQuery**](http://youmightnotneedjquery.com/).

**2.3.5.2 Syntax**:

// when a user clicks the button with ID "trigger":

$('#trigger').click(function() {

//change the body's background to yellow

$('body').css('background', 'yellow');

//fade out all "img" elements over 3 seconds

$('img').fadeOut(3000, function() {

// remove "img" elements from page when fadeOut is done:

$(this).remove();

});

});

* **NOTE**: The **$** sign is defined as a function in jQuery, but otherwise has no unique characteristic in plain JS.
* **NOTE**: In order to access a specific DOM element (such as the first <div> on a page) after, you must access it via $('div')[0];. This will allow you to work with the specific DOM element instead of the whole jQuery object containing all <div> elements.

**2.3.5.3 Selecting with jQuery**

* Selecting with jQuery is very similar to querySelectorAll() in the sense that we provide a CSS style selector and jQuery will return ALL matching elements:

$('selectorGoesHere');

* Selecting can be done on HTML tags, classes, and IDs, and specific selections can be performed in the same manner as in CSS (e.g., all <a> tags inside of <li> tags):

$('li a');

* **TIP**: See here for a list of all [**jQuery Selectors**](https://api.jquery.com/category/selectors/).
* **Manipulating Style**
* The **.css()** method is jQuery's interface to styling:

$('selector').css('property', 'value');

· For example, to place a border around an element with the ID "special":

$('#special').css('border', '2px solid red');

* In addition to individual properties, an entire **JavaScript Object** containing multiple properties can be passed into an element by setting a variable:

var styles = {

color: 'red',

background: 'pink',

border: '2px solid purple'

};

$('#special').css(styles);

**2.3.5.4 Common** [**Methods**](https://api.jquery.com/)

* **text()**

· (1) Get the combined text contents of eachelement in the set of matched elements, including their descendants, or (2) set the text contents of the matched elements. (Basically, the jQuery equivalent of "textContent".)

* A value passed into text() will update the text content, e.g.:

$('h1').text('New Text');

* **NOTE**:All text within the elements will be returned in one string.
* **html()**

· (1) Get the HTML contents of the firstelement in the set of matched elements or (2) set the HTML contents of every matched element. (Similar to "innerHTML".)

* If you have a set of three <li> tags, saying $('li').html() will return the inner HTML of the first <li>, but if you say $('li').html('<strong>New Text</strong>'), then that will change the inner HTML of all three.
* **NOTE**: To get the value for each element individually, use a **Looping Construct** such as jQuery's [**.each()**](https://api.jquery.com/each/) or [**.map()**](http://api.jquery.com/jquery.map/) methods.
* **NOTE**: To set the contents of just the **first** <li>, you can use CSS pseudo-selector ":first-of-type", or jQuery's ":first" shorthand selector, or jQuery's .first() method. Technically, the former is more resource efficient (due to being built-in by JS) and is preferred.
* **NOTE**: To set the contents of the **last** <li>, you can use CSS pseudo-selector ":last-of-type" or jQuery's .last() method, e.g.:

$('li').last().css('border', '1px solid red');

* **attr()**

· (1) Get the value of an attribute for the firstelement in the set of matched elements or (2) set one or more attributes for every matched element.

* **Syntax**:
* To "get" an attribute's value (return the value), simply pass through the name of the attribute:

$('selector').attr('attributeName')

* However, to "set" an attribute's value, you must pass through both the name of the attribute and the new value:

$('selector').attr('attributeName', 'value')

* **NOTE:** To set multiple attributes at once, pass through an object containing the attributes:

$('#photo').attr({

alt: 'text',

title: 'text'

});

* **val()**

· (1) Get the current value of the first element in the set of matched elements or (2) set the value of every matched element. This allows you to extract the value from an **Input** in the same way that .value() does in JavaScript.

* **BUT NOTE**: Although most useful for inputs, this method actually works on ALL elements that have a value attribute (e.g., checkboxes, dropdown menus).

· Like the methods above, .val() uses the **Getter/Setter** **Dynamic**. If run without an argument, it will return the value. If run with an argument passed through, it will set that argument as the new value.

* **NOTE**: While it's not common that you would want to change the value of an input (this is usually the user's job), this method is useful for clearing the value in the input box after the user has entered the input, i.e.:

$(this).val('');

* **addClass()**
* **removeClass()**
* **toggle()**

· These work in the same manner as their vanilla JavaScript counterparts.

* **NOTE**: "add" and "remove" can be used in tandem like so:

$('p').removeClass('firstClass').addClass('secondClass');

* **ready()**

· The .ready() method offers a way to run JS code as soon as the page's DOM becomes safe to manipulate:

$(document).ready(function() {

// code to run after DOM has been loaded

});

* **NOTE**: As of jQuery 3.0, the above syntax has been **Deprecated**, and the [**Recommended Syntax**](https://api.jquery.com/ready/) has been abbreviated as follows:

$(function() {

// code to run after DOM has been loaded

});

* **IMPORTANT**: jQuery's .ready() method differs from JavaScript's built-in window.onload() method. The latter fires when **ALL** content on the page has loaded (including the DOM, asynchronous JS, frames, and images), whereas the former executes after just the **DOM** has loaded (i.e., the function will execute after all HTML tags and scripts have loaded; it will not wait for the images and frames to fully load). Thus, if it is more important that you execute a function prior to your images being loaded, use the .ready() method. Conversely, if it is more important that your images load first before a less-essential function, then use window.onload().
* **Common** [**Events**](https://api.jquery.com/category/events/)

**click()**

· Bind an event handler to the "click" JS event, or trigger that event on an element.

**o Syntax**:

$('button').click(function() {

alert('You clicked a button.');

});

**o THIS NOTE**: If you want to use "this" in jQuery (to, for example, specify that the .click() event changes the background color of the particular button pressed among a collection of buttons), you have to wrap "this" in jQuery language (i.e., $(this)), because you are applying a jQuery method to a jQuery object, not a vanilla JS object, e.g.:

$('button').click(function() {

$(this).css('background', 'green');

});

**2.3.5.4 keypress()**

· Bind an event handler to the "keypress" JS event, or trigger that event on an element.

**o NOTE**: .keypress() focuses on which character is entered, rather than focusing on the key's physical state. If you want to trigger an event specifically when the key is pressed down, use **.keydown()** instead. Conversely, use **.keyup()** to trigger the even when the key is released. This distinction is important for inputs where a user may press the Shift key to capitalize his name. If you use .keydown(), the browser will treat the down-pressed "Shift" as its own entry and the lowercase "a" as a subsequent entry.

· When working with a text **Input** and you want to have the user's data be taken when the user hits the Enter key, use the following syntax that works with the **Event Object**:

$('input[type="text"]').keypress(function(event ) {

if (event.which === 13) {

//run code when Enter key is pressed

}

});

* Here, the "event" object (call it whatever you want) contains all information about the .keypress() event. (**NOTE**: The information is always logged, but never captured as a variable unless you include the argument.) One property in this information is called "**Which**", and that corresponds to the unique key code that jQuery looks at to know "which" key is pressed. Here, the Enter key has a "which" value of 13.

2.3.5.5 Refer to [**JavaScript Char Codes**](https://www.cambiaresearch.com/articles/15/javascript-char-codes-key-codes) for a table for all key codes.

* **on()**

· Attach an event handler function for one or more events to the selected elements. Behaves in the same manner as ".addEventListener()" by letting you specify the type of event to listen for, e.g.:

$('li').on('mouseenter', function() {

$(this).css('fontWeight', 'bold');

});

$('li').on('mouseout', function() {

$(this).css('fontWeight', 'normal');

});

* **VERY IMPORTANT NOTE**: In you are using click listeners, why use .on() rather than just .click()? Because .click only adds click listeners to objects that are **on the page at that time**, whereas .on() will add listeners to **all FUTURE elements if** [**Event Delegation**](https://learn.jquery.com/events/event-delegation/) **is used**.

**2.3.5.6 Common** [**Effects**](https://api.jquery.com/category/effects/)

* **Fading**
* **fadeOut()** hides the matched elements by fading them to transparent.
* **NOTE**: You can modify the rate at which the element fades by including an argument stating the desired time (in ms) to complete the fade effect (the default is 400ms, and you can also use "slow" or "fast" to operate as 600ms or 200ms, respectively):

$('li').fadeOut(800);

* **NOTE**: You can modify the "easing" of the fade effect by specifying a second (or first if rate is not specified) argument of either "linear" or "swing", with the latter being the default:

$('li').fadeOut(200, 'linear');

* **NOTE**: Finally, you can specify a function as a third (or second if easing not specified, or first if neither rate nor easing are specified) argument to trigger when the animation is complete:

$('li').fadeOut('fast', function() {

// run your code

});

* **IMPORTANT**: When an element is faded out, it is not removed from the DOM. It's "display" property is merely set to "none". If you want to remove the element, then use a callback function in tandem with the .remove() method:

$('li').fadeOut(function() {

$(this).remove();

});

* **fadeIn()** displays the matched elements by fading them to opaque. It operates exactly the same way as .fadeOut() (including the ability to use all three arguments), except in reverse (by removing "display: none" from the element).
* **fadeToggle()** display or hide the matched elements by animating their opacity. It operates the same as above (like using .toggle()).

**2.3.5.7 Sliding**

· Operates the same as fading, except by sliding the element up (and hiding it) or down (and displaying it) by animating the element's height (rather than opacity). The three effects are .slideUp(), .slideDown(), and .slideToggle().

* **Other Methods**
* **stopPropagation()**

· If a parent element has an event listener (e.g., a click listener), and its child element has its own event listener, modern browsers will first trigger the child element's event listener, then the parent's event listener, and will continue to "bubble" up the chain until it reaches the <html> element itself. (This is a phenomenon known as **Event Bubbling**.) This can be problematic when you only want a child element's event listener to fire without triggering any of its parent's event listeners. jQuery allows you to prevent this behavior by its .stopPropagation() method:

$('p').click(function(event) {

event.stopPropagation(); // run code for child element only

});

* **NOTE**: Similar to the .which() method used in the .keypress() method above, .stopPropagation() requires the use an "event" object that can be called whatever you want.
* **ALSO NOTE**: Although .stopPropagation() is a jQuery method in this particular context, it should be noted that .stopPropagation() is also a [**built-in method in plain JS**](https://developer.mozilla.org/en-US/docs/Web/API/Event/stopPropagation) (as are various other jQuery methods).
* **parent()**

· Get the immediate parent of each element in the current set of matched elements. The method optionally accepts a selector expression. If the selector is supplied, the elements will be filtered accordingly. For example, if you want to use the .remove() method to remove an <li> when a <span> within the <li> is clicked (as opposed to just removing the <span>):

$('span').on('click', function() {

$(this).parent().remove();

});

* **Append()**

· Insert content to the end of each element in the set of matched elements. For example, to add an <li> to the end of all <ul> tags on the page:

$('ul').append(<li>Text</li>);