JQuery Cheat Sheet PDF

P.S. - Know how to learn JavaScript quickly (https://hackr.io/blog/how-to-learn-javascript).

AJAX

Global Ajax Event Handlers

- **.ajaxComplete(handler)** - Registers an event handler that is to be called when AJAX requests complete.
- **.ajaxError(handler)** - Registers an event handler that is to be called when AJAX requests complete with an error.
- **.ajaxSend(handler)** - Adds a function that is to be executed before an AJAX request is sent.
- **.ajaxStart(handler)** - Registers an event handler that is to be called when the first AJAX request begins.
- **.ajaxStop(handler)** - Registers an event handler that is to be called when all AJAX requests have been completed.
- **.ajaxSuccess(handler)** - Registers an event handler that is to be called when an AJAX request completes successfully.

Helper Functions

- **jQuery.param(obj)/jQuery.param(obj, traditional)** - Creates a serialized representation of an array or a jQuery object suitable to use in a URL query string or an AJAX request. When a jQuery object is passed, it must contain input elements with name/value properties.
- **.serialize()** - Encodes a set of form elements as a string for submission.
- **.serializeArray()** - Encodes a set of form elements as an array of names and values.

Low-Level Interface

- **jQuery.ajax(url [, settings])/jQuery.ajax([settings])** - Performs an AJAX, asynchronous HTTP, request.
- **jQuery.ajaxPrefilter([dataTypes], handler)** - Handles custom AJAX options or modify existing options before each request is sent and the same is processed by *$.ajax()*.
- **jQuery.ajaxSetup(options)** - Sets default values for future AJAX requests.
- **jQuery.ajaxTransport(dataType, handler)** - Creates an object that handles the transmission of AJAX data.

Shorthand Methods

- **jQuery.get(url [, data][, success][, dataType])/jQuery.get([settings])** - Loads data from the server using an HTTP GET request.
- **jQuery.getJSON(url [, data][, success])** - Loads JSON-encoded data from the server using an HTTP GET request.

- **jQuery.getScript(url [, success])** - Loads and executes a JS file from the server using an HTTP GET request.
- **jQuery.post(url [, data][, success][, dataType])/jQuery.post([settings])** - Loads data from the server using an HTTP POST request.
- **.load(url [, data][, complete])** - Loads data from the server and place the returned HTML into the matched elements.

ATTRIBUTES/CSS

Attributes

- **.attr()** - Gets the value of an attribute for the first element among the set of matching elements or sets one or many attributes for every matched element. Its variations are:
- **attr(attributeName)**
- **attr(attributeName, value)/attr(attributes)/attr(attributeName, function)**
- **.prop()** - Gets the value of a property for the first element among the set of matching elements or sets one or many properties for each of the matching elements. Its variations are:
- **.prop(propertyName)**
- **.prop(propertyName, value)/.prop(properties)/.prop(propertyName, function)**
- **.removeAttr(attributeName)** - Removes an attribute from each of the elements among the set of matching elements.
- **.removeProp(propertyName)** - Removes a property from the set of matching elements.
- **.val()/.val(function)/.val(value)** - Gets the present value of the first element among the set of matching elements or sets the value of each matching element.

CSS

- **.addClass(className)/.addClass(function)** - Adds the specified class(es) to every element among the set of matching elements.
- **.css()** - Gets the value of a computed style property for the first element among the set of matching elements or sets one or many CSS properties for each of the matching elements. Its variations are:
- **.css(propertyName)/.css(propertyNames)**
- **.css(propertyName, value)/.css(propertyName, function)/.css(properties)**
- **jQuery.cssHooks** - Offers a way for defining functions for getting and setting particular CSS values. It can also be used for creating new cssHooks for normalizing CSS3 features.
- **jQuery.cssNumber** - It is an object containing all CSS properties usable without a unit.

- ***Note***: The object is used by the *.css()* method to check whether there is a need to add *px* to unitless values.
- **jQuery.escapeSelector(selector)** - Escapes a character that has a special meaning in a CSS selector.
- **.hasClass(className)** - Checks whether any of the matching elements are assigned to the specified class.
- **.removeClass([className])/.removeClass(function)** - Removes one, many, or all classes from each element among the set of the matching elements.
- **.toggleClass()** - Adds or removes one or many classes from every element among the set of matching elements depending on the value of the state argument or the presence of the class. Its variations are:
- **.toggleClass(className)/.toggleClass(className, state)/.toggleClass(function [, state])**
- **toggleClass([state])**

Dimensions

- **.height()/.height(function)/.height(value)** - Returns the present computed height for the first element among the set of matched elements or sets the height of every matching element.
- **.innerHeight()/.innerHeight(function)/.innerHeight(value)** - Returns the current computed inner height for the first element among the set of matching elements or sets the inner height of every matching element.
- **.innerWidth()/.innerWidth(function)/.innerWidth(value)** - Returns the current computed inner width for the first element among the set of matching elements or sets the inner width of each matching element.
- **.outerHeight([includeMargin])/.outerHeight(function)/.outerHeight(value [, includeMargin])** - Returns the present computed outer height for the first element among the set of matching elements or sets the outer height of each matching element.
- **.outerWidth([includeMargin])/.outerWidth(function)/.outerWidth(value [, includeMargin])** - Gets the current computed outer width for the first element among the set of matching elements or sets the outer width of each matching element.
- **.width()/.width(function)/.width(value)** - Returns the present computed width for the first element among the set of matching elements or sets the width of every matching element.

Offset

- **.offset()/.offset(coordinates)/.offset(function)** - Returns the current coordinates of the first element among the set of matching elements or sets the coordinates of each element.
- **.offsetParent()** - Gets the closest positioned ancestor element.

- ***Note***: An element is called positioned if it has a CSS position attribute of absolute, fixed, or relative.
- **.position()** - Gets the current coordinates of the first element among the set of matching elements w.r.t. to the offset parent.
- **.scrollLeft()/.scrollLeft(value)** - Gets the current horizontal position of the scroll bar for the first element among the set of matching elements or sets the horizontal position of the scroll bar for each of the matching elements.
- **.scrollTop()/.scrollTop(value)** - Gets the current vertical position of the scroll bar for the first element among the set of matching elements or sets the vertical position of the scroll bar for each of the matching elements.

Data

- **jQuery.data(element)/jQuery.data(element, key)/jQuery.data(element, key, value)** - Stores the arbitrary data associated with the passed element and/or returns the value that was set.
- **.data()/.date(key)/.data(obj)/.data(key, value)** - Stores arbitrary data associated with the matching elements or returns the value at the named data store for the first element among the set of matching elements.
- **jQuery.hasData(element)** - Checks the specified element for any associated jQuery data.
- **jQuery.removeData(element [, name])** - Removes a specified data from the passed element.
- ***Note***: Low-level method. Using the *.removeData()* method is recommended.
- **.removeData([name])/.removeData([list])** - Removes a specified data from the passed element.

MANIPULATION

Copying

- **.clone([withDataAndEvents])/.clone([withDataAndEvents][, deepWithDataAndEvents])** - Creates a deep copy of the set of the matching elements.

DOM (https://en.wikipedia.org/wiki/Document_Object_Model) Insertion, Around

- **.wrap(function)/.wrap(wrappingElement)** - Wraps an HTML structure around every element among the set of matching elements.
- **.wrapAll(function)/.wrapAll(wrappingElement)** - Wraps an HTML structure around all elements among the set of matching elements.
- **.wrapInner(function)/.wrapInner(wrappingElement)** - Wraps an HTML structure around the content of every element of the set of the matching elements.

DOM Inspection, Inside

- **.append(function)/.append(content [, content])** - Inserts content specified by the passed parameter to the end of every element among the set of matching elements.
- **.appendTo(target)** - Inserts each element of the set of matching elements to the end of the specified target.
- **.html()/.html(function)/.html(htmlString)** - Gets the HTML contents of the first element among the set of matching elements or sets the HTML contents of each of the matching elements.
- **.prepend(content [, content])/.prepend(function)** - Inserts content specified by the passed parameter to the beginning of every element among the set of matching elements.
- **.prependTo(target)** - Inserts each element in the set of matching elements to the beginning of the specified target.
- **.text()/.text(function)/.text(text)** - Gets the combined text contents of every element among the set of matching elements, including the descendants, or sets the text content of every matching element.

DOM Insertion, Outside

- **.after(content [, content])/.after(function)/.after(function-html)** - Inserts specified content after every element among the set of matching elements.
- **.before(content [, content])/.before(function)/.before(function-html)** - Inserts specified content before every element among the set of the matched elements.
- **.insertAfter(target)** - Inserts each element in the set of matching elements after the specified target.
- **.insertBefore(target)** - Inserts each element in the set of matching elements before the specified target.

DOM Removal

- **.detach([selector])** - Removes all the matching elements from the DOM.

Note: The method is similar to the *.remove()* method, with the exception that the former retains all of the jQuery data associated with the removed elements. Hence, it is useful in scenarios requiring reinserting the removed elements into the DOM sometime later.

- **.empty()** - Removes all the child nodes of the matching elements from the DOM.

Note: As per the DOM specification, any text string within an element is considered a child node of that particular element. Hence, the method also removes any text within the set of matching elements.

- **.remove([selector])** - Removes all the matching elements from the DOM.

- **.unwrap()/.unwrap([selector])** - Removes and replaces the parent nodes of all the matching elements from the DOM with the matching elements.

DOM Replacement

- **.replaceAll(target)** - Replaces each of the specified target elements with the set of matching elements.
- **.replaceWith(newContent)/.replaceWith(function)** - Replaces every element among the set of matching elements with the specified content and returns the set of removed elements.

TRAVERSING

Filtering

- **.eq(index)/.eq(indexFromEnd)** - Constructs a new jQuery object from the element specified by the passed index.
- **.filter(selector)/.filter(function)/.filter(elements)/.filter(selection)** - Reduces the set of matching elements to those elements that match the specified selector, jQuery object, element(s), or pass the specified function.
- **.first()** - Constructs a new jQuery object from the first element among the set of the matching elements.
- **.has(selector)/.has(contained)** - Reduces the set of matching elements to those elements that have a descendant element matching the specified selector or the specified DOM element.
- **.is(selector)/.is(function)/.is(selection)/.is(elements)** - Checks the current set of matching elements against a specified selector, element(s), jQuery object, or function and returns true if at least one of the elements matches the specified arguments.
- ***Note***: This method, unlike other filtering methods, doesn’t create a new jQuery object. Thus, it allows testing the contents of a jQuery object without modification. This makes it useful for using inside callbacks.
- **.last()** - Reduces the set of matching elements to the last element of the same.
- **.map(callback)** - Returns a new jQuery object containing the return values resulting by passing every element in the current set of matching elements through a function.
- **.not(selector)/.not(function)/.not(selection)** - Constructs and returns a new jQuery object from a subset of matching elements containing elements that don’t pass the specified selector, element(s), or function’s test.
- **.slice(start [, end])** - Reduces the set of matching elements to a subset specified by a range of indices.

Miscellaneous Traversing

- **.add(selector)/.add(elements)/.add(html)/.add(selection)/.add(selector, context)** - Creates and returns a new jQuery object containing the matching elements and those passed using a selector, element(s), HTML snippet, or jQuery object.
- **.addBack([selector])** - Adds the previous set of elements on the stack to the current set of elements, optionally filtered using a selector.
- **.contents()** - Constructs and returns a new jQuery object containing the child nodes of each element, including text nodes, comment nodes, and HTML elements, among the set of matching elements.
- **.each(function)** - Iterates over a jQuery object and executes the specified function on every matching element.
- **.end()** - Ends the most recent filtering operation on the current set of elements and returns the previous states (values) of the matching elements.

Tree Traversal

- **.children([selector])** - Constructs and returns a new jQuery object containing the child nodes of every element among the set of matching elements.

Note: The method is different from the *.find()* method in the way that the latter can traverse down multiple levels to select grandchildren, great-grandchildren, etc. It is also similar to the *.contents()* method with the exception that the *.contents()* method also selects text nodes, comments nodes, and HTML elements.

- **.closest(selector)/.closest(selector [, context])/.closest(selection)/.closest(element)** - Constructs and returns a new jQuery object containing elements filtered using the specified selector, element(s), or a jQuery object from the set of matching elements and their ancestors in the DOM tree.
- **.find(selector)/.find(element)** - Gets the descendants of every element in the current set of matching elements, filtered using a selector, element(s), or a jQuery object.
- **.next()/.next([selector])** - Gets the immediately following sibling of every element in the set of matching elements, optionally filtered by a selector expression.
- **.nextAll()/.nextAll([selector])** - Gets all the following siblings of every element among the set of matching elements, optionally filtered using a selector.
- **.nextUntil([selector][, filter])/.nextUntil([element][, filter])** - Gets all the following siblings of every element from the set of matching elements and continuing until coming across an element matched by the specified DOM node, selector, or jQuery object.
- **.parent()/.parent([selector])** - Gets the parent node of every element among the set of matching elements, optionally filtered using a selector expression.

- **.parents()/.parents([selector])** - Gets the ancestors of every element among the set of matching elements, optionally filtered by a selector expression.

Note: The difference between the *.parent()* method and *.parents()* method is that the former only traverse a single level up the DOM tree.

- **.parentsUntil([selector][, filter])/.parentsUntil([element][, filter])** - Gets the ancestors of every element from the set of matching elements and continuing until coming across an element matched by the specified DOM node, jQuery object, or selector expression.
- **.prev()/.prev([selector])** - Constructs and returns a new jQuery object containing the immediately preceding sibling, optionally filtered by a selector expression, of every element among the set of matching elements.
- **.prevAll()/.prevAll([selector])** - Gets all the preceding siblings of every element among the set of matching elements, optionally filtered using a selector expression.
- **.prevUntil([selector][, filter])/.prevUntil([element][, filter])** - Continues getting the preceding siblings of every element from the set of matching elements until coming across an element matched by the specified DOM node, jQuery object, or selector expression.
- **.siblings()/.siblings([selector])** - Constructs and returns a jQuery object containing all the siblings of every element belonging to the set of the matching elements, optionally filtered using a selector expression.

Conclusion

With that, we're done with the jQuery Cheat Sheet Part - II. I hope you will find it useful when leveraging jQuery in your code, learning jQuery (https://hackr.io/tutorials/learn-jquery?ref=blog-post), or doing anything requiring quick jQuery reference. Can't find something you were looking for? Try jQuery Cheat Sheet Part - I (https://hackr.io/blog/jquery-cheat-sheet-part-1).

Spotted any misinformation? Let us know via comments so that we can mend it ASAP and better this **jQuery Cheat Sheet**. Thanks in advance!

You can download jQuery Cheat Sheet II from here (https://hackr.io/blog/media/jquery-cheat-sheet-2.pdf).

People are also reading:

- JQuery Interview Questions and Answers (https://hackr.io/blog/jquery-interview-questions)
- Download Bootstrap Cheat Sheet (https://hackr.io/blog/bootstrap-cheat-sheet)
- Download C# Cheat Sheet (https://hackr.io/blog/c-sharp-cheat-sheet)
- Download MySQL Cheat Sheet (https://hackr.io/blog/mysql-cheat-sheet)
- Download Linux Cheat Sheet (https://hackr.io/blog/linux-cheat-sheet)
- Download Git Cheat Sheet (https://hackr.io/blog/git-cheat-sheet)