Notes:

Ajax with jQuery

The Document Ready Event

$(document).ready(function(){

// jQuery methods go here...

});

* This is to prevent any jQuery code from running before the document is finished loading (is ready).
* It is good practice to wait for the document to be fully loaded and ready before working with it.
* This also allows you to have your JavaScript code before the body of your document, in the head section.
* Shorter method for the document ready method

$(function(){

// jQuery methods go here...

});

* Use the syntax you prefer. We think that the document ready event is easier to understand when reading the code.

Use jquery from online

* <https://code.google.com/apis/libraries/>
* Jquery: <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.2/jquery.min.js"></script>
* Downside of cdn is if the net is not available then jquery is not working properly

Keep in Mind

* DOM is loading from top to bottom
* DOM is the representation of our page
* Write jquery at the bottom of the page just before the ending tag of the body
* So jQuery is written down in bottom after html body for working perfectly
* Or use document.ready method
* In jQuery first target the element from the dom and then apply your desired actions
* We should ensure that one page has only one ID, if a page has more than one ID jQuery will return only the first one

jQuery

* jQuery is a fast, small, and feature-rich JavaScript library.
* Is a free, open JavaScript library
* Simplifies the task of creating highly responsive web pages
* Works across modern browsers
* Abstracts away browser-specific features, allowing us to concentrate on design
* Focuses on simplifying common scripting tasks
* Getting and manipulating page content
* Working with the modern browser event model
* Adding sophisticated effects
* Most modern web development scenarios involve common patterns
* Page loads --- Perform a bunch of page setup
* Event -------- Retrieve content from the page ----- Manipulate or animate the content ------ Put the content back in the page

Benefits of using jQuery

* Leverages your existing knowledge of CSS
* Works with sets of elements
* Performs multiple operations on a set of elements with one line of code (known as statement chaining)
* Hides various browser quirks (so we can concentrate on the end result)
* Is extensible (so you can use third-party-plug-ins to perform specialized tasks, or write your own)

jQuery Syntax

* With jQuery we select (query) HTML elements and perform "actions" on them.
* jQuery uses CSS syntax to select elements.
* Basic syntax is: $(selector).action()
* A $ sign to define/access jQuery
* A (selector) to "query (or find)" HTML elements
* A jQuery action() to be performed on the element(s)

The Document Ready Event

$(document).ready (function () {

// jQuery methods go here...

});

* This is to prevent any jQuery code from running before the document is finished loading (is ready).
* Here are some examples of actions that can fail if methods are run before the document is fully loaded:
* Trying to hide an element that is not created yet
* Trying to get the size of an image that is not loaded yet

The jQuery Philosophy

* The jQuery philosophy is “Write less, do more.” This philosophy can be further brokendown into three concepts:
* Finding some elements (via CSS selectors) and doing something with them (via jQuery methods)
* Chaining multiple jQuery methods on a set of elements
* Using the jQuery wrapper and implicit iteration

Purpose of jQuery

* jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code.
* jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.

jQuery browser compatibly

* Currently compatible with modern versions of all the main browsers in use today

Difference between ID’s and Class

ID's are unique

* Each element can have only one ID
* Each page can have only one element with that ID

Classes are NOT unique

* You can use the same class on multiple elements.
* You can use multiple classes on the same element.

jQuery Way

* Typically, code that you want to execute when the page is loaded is written like this:

function runOnLoad() {

alert(“the page just loaded!”);

}

Window.onload = runOnLoad;

* The on-load event only fires after all of the page content has downloaded, including images
* It’s also harder to add multiple load functions
* jQuery provides a way to run code when the DOM of the page is ready
* This is called the document.ready event

jQuery code:

* The document.ready event is written like this:
* $(“document”).ready( function() {

alert(“the page just loaded!”);

});

* This code will now execute when the DOM has loaded, instead of waiting for all the page content finish downloading.
* Also we can call the document.ready function multiple times, and jQuery will chain together each one to be called in succession.

Overview of jQuery Features

* jQuery’s features break down across 8 major categories:
* Core Functionality
* Implements core jQuery functions as well as commonly used utilities
* Selection and Traversal
* Functions for finding content in documents and navigating among the contents of the document
* Manipulation & CSS
* Functions for editing and changing document content and working with CSS data such as positioning info
* Events
* Simplifies working with the modern DOM events and provides common event helper functions
* Effects
* Functions for creating basic animations and effects such as hiding and showing elements and moving objects around
* AJAX
* Utilities for working with AJAX, such as loading content from pages and dealing with JSON data

jQuery Selectors:

* jQuery selectors start with the dollar sign and parentheses − $().
* The factory function $() makes use of following three building blocks while selecting elements in a given document –

S.N. Selector & Description

1 Tag Name

Represents a tag name available in the DOM. For example $('p') selects all paragraphs <p> in the document.

2 Tag ID

Represents a tag available with the given ID in the DOM. For example $('#some-id') selects the single element in the document that has an ID of some-id.

3 Tag Class

Represents a tag available with the given class in the DOM. For example $('.some-class') selects all elements in the document that have a class of some-class.

JQuery selectors and filters: overview

* jQuery selectors and filters retrieve content from the document so it can be manipulated using other functions (think of this as the “Query” part of the “jQuery”)
* jQuery selectors return an array of objects that match the selection criteria
* jQuery filters operate on a selector to further refine the results array that the selector returns
* This array is not a set of DOM elements
* It is a collection of jQuery objects that provide a large number of predefined functions for further operating on the objects.

Using Basic JQuery Selectors

* CSS-style selectors and filters are based on familiar CSS syntax, and work pretty much the same way as CSS does
* The CSS selectors listed here correspond directly to their CSS counterparts

|  |  |  |
| --- | --- | --- |
| SELECTOR | PURPOSE | |
| tagname | Finds all elements that are named tagname | |
| #identifier | finds all elements with ID of identifier | |
| .className | finds all elements that have class attribute with the value of className | |
| tag.className | Gets elements of type tag that have a class attribute with the value of className | |
| tag#id.className | retrieves the tag element that has an ID of id and a class attribute with the value of className | |
| \* | Finds all of the elements on the page |  |

Using jQuery vs. using the plain browser DOM

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Document</title>

</head>

<body>

<ul id="list1">

<li class="a">item 1</li>

<li class="a">item 2</li>

<li class="b">item 3</li>

<li class="b">item 3</li>

</ul>

<p>This is paragraph 1</p>

<p>This is paragraph 2</p>

<p>This is paragraph 3</p>

<p>This is paragraph 4</p>

</body>

</html>

* Get all<p>tags(DOM):

Document.getElementsByTagName(“p”);

* Get all <p>tags(jQuery):

$(“p”);

* Get the tag with id “list1”(DOM):

Document.getElementById(“list1”);

* Get the tag with id “list1”(jQuery):

$(“#list1”);

* Get all <li> tags with class “a”(jQuery):

$(“li.a”);

* Get all tags with class “b”, but only if they are inside a <ul> (jQuery):

$(“ul .b”);

* The hierarchy and combination selectors allow us to get a little more advanced in selecting page content
* We can select elements based on hierarchical relationships or on a series of common criteria

|  |  |
| --- | --- |
| Selector | Purpose |
| Selector, Selector,…... | Finds all of the specified selectors |
| .class1.class2 | Finds all elements with both .class1 and .class2 applied |
| Parent>child | Finds all child elements that are direct children of elements of type parent |
| ancestor descendant | Finds all descendant elements that are contained within elements of type ancestor |
| Prev+next | Finds all next elements that are next to prev element |
| Prev~siblings | Finds all sibling elements that come after prev and match siblings selector |

* parent > child $("div > p") All <p> elements that are a direct child of a <div> element
* ancestor descendant

- Selects all elements that are descendants of a given ancestor.

- A descendant of an element could be a child, grandchild, great-grandchild, and so on, of that element.

Example - $("div p") All <p> elements that are descendants of a <div> element

* element + next $("div + p") The <p> element that are next to each <div> elements
* element ~ siblings $("div ~ p") All <p> elements that are siblings of a <div> element
* .class,.class $(".intro,.demo") All elements with the class "intro" or "demo"
* el1,el2,el3 $("h1,div,p") All <h1>, <div> and <p> elements
* \* $("\*") All elements

jQuery Filters:

* Filters work with selectors to provide even more fine-grained control over how elements are selected in the document
* jQuery filters fall into six different categories
* Basic
* Provides basic filtering, like getting the first, last and even and odd-numbered items in a returned set
* Content
* Filters a set of elements based on the content, like whether an element contains a particular string
* Visibility
* Filters a set of elements using the visibility setting of each element as a test
* Attribute
* Examines a given attribute on an element to determine whether it should be filtered out
* Child
* Selects elements based upon their relationship with their parent element
* Form
* Provides specialized filters that operate on form elements

Using Basic jQuery Filters

* We can refine a selector by including elements that match certain conditions, like position or index

|  |  |
| --- | --- |
| Filter | Purpose |
| :first | Selects only the first instance of the selector’s returned set |
| :last | Selects only the last instance of the selector’s returned set |
| :even | Selects only even-numbered elements in the selector’s returned set |
| :odd | Selects only odd-numbered elements in the selector’s returned set |
| :eq(n) | Filters out elements that are not positioned at the given index |
| :gt(n) | Includes elements that are greater the given index |
| :lt(n) | Includes elements that are before the given index |
| :header | Selects all header elements (H1, H2, H3, etc.) |
| :not(selector) | Includes elements that do not match the given selector |
| :animated | Selects all elements that are currently being animated in some way |

jQuery Attribute Filters

* We can filter the results of a selector statement based on attribute content

|  |  |
| --- | --- |
| Filter | Purpose |
| [attribute] | Includes elements in the result set if they have the specified attribute |
| [attribute=value] | Includes elements in the result set if they have the specified attribute and it has the given value |
| [attribute!=value] | Includes elements in the result set only if they have the specified attribute and it doesn’t have the given value |
| [attribute^=value] | Includes elements that have the specified attribute and it starts with specified value |
| [attribute$=value] | Includes elements that have the specified attribute and it ends with specified value |
| [attribute\*=value] | Includes elements that have the specified attribute and it contains the specified value |
| [attrFilter1][attriFilterN] | Includes elements that match all of the specified attribute filters |

jQuery Content and Visibility Filters

* We can examine the content of selected elements or their visibility property to determine whether they should be included or excluded from the final set

|  |  |
| --- | --- |
| Content Filter | Purpose |
| :contains(text) | Filters the selection to only include elements that contain the text string |
| :empty | Filters the selection to only include empty elements |
| :has(selector) | Matches element that contain at least one element that has the specified selector |
| :parent | Matches all elements that are parents(i.e. they contain at least one other element, including text) |

|  |  |
| --- | --- |
| Visibility Filter | Purpose |
| :visible | Filters the selection to only include visible elements |
| :hidden | Filters the selection to only include hidden elements |

Child Filters

* We can refine a selector by examining the relationship each element has with its parent element

|  |  |
| --- | --- |
| Filter | Purpose |
| :nth-child(index)  :nth-child(even)  :nth-child(odd)  :nt-child(equation) | Matches elements at index, or even or odd increments, or who match an equation of the form Xn+M (for example, 2n or 3n+1) |
| :first-child | Matches elements who are the first child of their parent |
| :last-child | Matches elements who are the last child of their parent |
| :only-child | Matches elements who are the only child of their parent |

Form Selectors

* We can use form selectors to deal with form elements
* They work like other selectors but start with a colon(:) like a regular filter

|  |  |
| --- | --- |
| Selector | Purpose |
| :input | Finds all input, select, textarea and button elements |
| :text | Finds all text elements |
| :password | Finds all password elements |
| :radio | Finds all radio button elements |
| :checkbox | Finds all checkbox elements |
| :submit | Finds all submit elements |
| :reset | Finds all reset elements |
| :image | Finds all image elements |
| :button | Finds all button elements |
| :file | Finds all file upload elements |

Form Filters

* We can perform additional filtering of form elements, such as whether items are checked, selected, or enabled

|  |  |
| --- | --- |
| Selector | Purpose |
| :enabled | Matches all form elements that are enabled |
| :disabled | Matches all form elements that are disabled |
| :checked | Matches all form elements that are checked (radio buttons and checkboxes |
| :selected | Matches all form elements that are selected |

Traversing Document Information

* We can traverse the information returned from a document easily

|  |  |
| --- | --- |
| Function/Property | Purpose |
| size(), length | The number of elements in the jQuery result set |
| get() | Returns an array of all matched DOM elements. Useful if we need to operate on the DOM elements themselves instead of using built-in jQuery functions |
| get(index) | Access a single matched DOM element at a specified index in the matched set |
| find(expression) | Searches for descendant elements that match the specified expression |
| each(fn) | Execute a function within the context of every matched element |

Manipulating Content: Overview

* Once we’ve used selectors and filters to retrieve web page content, we usually want to do something with it
* Sometimes we want to create new content to dynamically add into the page
* jQuery has functions for creating, copying, deleting, and moving content around, as well as wrapping page content in other content
* jQuery provides cross-browser support for working with CSS, including positioning and sizing information

Creating, Setting and Getting Content

* To create new HTML content, we simply pass a sting containing new HTML to the $() function:

Var newHeader = $(“<h1>My New Header</h1>”);

Var myStr = “<h1>My New Header</h1>”;

Var newHeader = $(myStr);

* In addition to this method, we can use the html() and text() methods to get and set content on

|  |  |
| --- | --- |
| Function | Purpose |
| html() | Returns the HTML content of the first matched element |
| html(newcontent) | Sets the HTML content of every matched element |
| text() | Returns the text content of the first matched element |
| text(newtext) | Sets the text content for all matched elements |

Manipulating Attributes

* To inspect or change the value of attributes on elements, use jQuery’s attr functions

|  |  |
| --- | --- |
| Function | Purpose |
| attr(name) | Accesses property on the first matched element. This method makes it easy to retrieve a property value from the first matched element. If the element does not have an attribute with such a name, undefined is returned |
| attr(properties) | Sets a series of attributes on all matched elements using an object notation syntax. This is the best used for setting large numbers of properties at once  $(“img”).attr({src:”/images/hat.gif”,  title:”jQuery”, alt: “jQuery Logo” }); |
| attr(key,value) | Sets a single property to a value on all matched elements |
| attr(key,fn) | Sets a single property to a computed value, on all matched elements. Instead of supplying a string value, a function is provided that computes the value of the attribute |
| removeAttr(name) | Removes the named attribute from all matched elements |

Inserting Content

* jQuery provides several functions for inserting content into the document, both before and after existing page elements

|  |  |
| --- | --- |
| Function | Purpose |
| append(content) | Appends content to the inside of every matched element |
| appendTo(selector) | Appends all of the matched elements to another, specified, set of elements |
| prepend(content) | Prepends content to the inside of every matched element |
| prependTo(selector) | Prepends all the matched elements to another, specified, set of elements |
| after(content) | Insets content after each of the matched elements |
| before(content) | Inserts content before each of the matched elements |
| insertAfter(selector) | Inserts all of the matched elements after another, specified, set of elements |
| insertBefore(selector) | Inserts all the matched elements before another, specified, set of elements before another, specified, set of elements |

Wrapping, Replacing, Removing Content

* jQuery can wrap existing content in the page, replace content, copy content, and remove it

|  |  |
| --- | --- |
| Function | Purpose |
| wrap(html) | Wraps each matched element with the specified HTML content |
| wrap(element) | Wraps each matched element with the specified element |
| wrapAll(html) | Wraps all the elements in the matched set with the specified HTML content |
| wrapAll(element) | Wraps all elements in the matched set into a single wrapper element |
| wrapInner(html) | Wraps the inner child contents of each matched element (including text nodes) with an HTML structure |
| wrapInner(element) | Wraps inner child contents of each matched element (including text nodes) with a DOM structure |
| replaceWith(content) | Replaces all matched elements with the specified HTML or DOM elements |
| replaceAll(selector) | Replaces the elements matched by the specified selector with the matched elements |
| empty() | Removes all child nodes from the set of matched elements |
| remove() | Removes all matched elements from the DOM |
| clone() | Clone matched DOM elements and selects the clones |
| clone(bool) | Clone the matched DOM elements, and all their event handlers, and select the clones |

Working with CSS Information

* jQuery’s CSS functions provide easy, cross-browser access for setting properties and working with positioning and sizing information
* The css() function allows us to retrieve and set CSS styles for a set of matched elements

|  |  |
| --- | --- |
| Filter | Purpose |
| css(name) | Returns the value for the named CSS property for the first matched element |
| css(properties) | Sets the CSS properties of every matched element using an object-notation syntax:  varcssObj = {  ‘background-color’ : ‘#ddd’,  ‘font-weight’ : ‘ ‘,  ‘color’ : ‘rgb(0,40,244)’ }  $(this).css(cssObj); |
| css(property,value) | Sets a single style property to a value on all matched elements. If an number is provided, it is automatically converted into a pixel value, with the following exceptions:z-index,font-weight,opacity,zoom,and line-height |

Working with CSS classes

* jQuery provides a set of functions for working with CSS classes on page elements
* Classes can be easily added, removed, toggled, and deteched

|  |  |
| --- | --- |
| Functions | Purpose |
| addClass(class) | Adds the specified class(es) to each of the set of matched elements |
| hasClass(class) | Returns true if the specified class is present on at least one of the set of matched elements |
| removeClass(class) | Removes all the specified class(es) from the set of matched elements |
| toggleClass(class) | Adds the specified class if it is not present, removes the specified class if it is present |
| toggleClass(class,switch) | Adds the specified class if the switch is true, removes the specified class if the switch is false |

Working with CSS positioning

* The CSS positioning functions provide cross-browser support for figuring out the positions of elements

|  |  |
| --- | --- |
| CSS Functions | Purpose |
| offset() | Gets the current offset of the first matched element, in pixels, relative to the document |
| offsetParent() | Returns a jQuery collection with the positioned parent of the first matched element |
| position() | Gets the top and left position of an element relative to its offset parent |
| scrollTop() | Gets the scroll top offset of the first matched element |
| ScrollTop(val) | Sets the scroll top offset to the given value on all matched elements |
| scrollLeft() | Gets the scroll left offset of the first matched element |
| scrollLeft(val) | Sets the scroll left offset to the given value on all matched elements |

Working with CSS sizing Information

* To retrieve cross-browser sizing information for elements, use the jQuery size-related CSS functions

|  |  |
| --- | --- |
| CSS Functions | Purpose |
| height() | Gets the current computed, pixel, height of the first matched element |
| height(val) | Sets the CSS height of every matched element |
| width() | Gets the current computed, pixel, width of the first matched element |
| width(val) | Sets the CSS width of every matched element |
| innerHeight() | Gets the inner height (excluding the border and including the padding) for the first matched element |
| innerWidth() | Gets the inner width (excluding the border and including the padding) for the first matched element |
| outerHeight(margin) | Gets the outer height (includes the border and padding by default) for the first matched element. If the margin argument is true, then the margin values are also included |
| outerWidth(margin) | Gets the outer width (includes the border and padding by default) for the first matched element. If the margin argument is true, then the margin values are also included |

JQuery Events: Overview

* Provides mechanism for working with events that is simpler than relying on the DOM
* Abstracts away the differences between browser implementations
* Makes it easy to assign event handlers to groups of elements by using selectors and filters

Binding/Unbinding

* Allows events to be wired up and torn down in a cross-browser way

Unified Event Object

* Provides an event object that exposes the most common properties in a cross-browser way

Convenience Features

* Provides functions that encapsulate common event features and cross-browser helper routines

Associating data with page elements:

* Store arbitrary data associated with the matched elements or return the value at the named data store for the first element in the set of matched elements.

JQuery Statement Chaining

* One of jQuery’s most powerful features is its ability to chain multiple functions together to perform several operations in one line of code

$(selector).fn1().fn2().fn3();

jQuery: nth-child() Selector

* The :nth-child(n) selector matches every element that is the nth child, regardless of type, of its parent.
* n can be a number, a keyword, or a formula.
* Using a formula (an + b). Description: a represents a cycle size, n is a counter (starts at 0), and b is an offset value.

Some examples:

* 1n+0, or simply n, would match every child element.
* 2n+0, or simply 2n, would match child elements 2, 4, 6, 8, etc. You can substitute the keyword even for this expression.
* 2n+1 would match child elements 1, 3, 5, 7, etc. You can substitute the keyword odd for this expression.
* 3n+4 would match child elements 4, 7, 10, 13, etc.
* The values a and b must both be integers, and the index of an element's first child is 1. In other words, this class matches all children whose index fall in the set { an + b; n = 0, 1, 2, ... }.

Examples

Example selectors

* tr:nth-child(2n+1)
* Represents the odd rows of an HTML table.
* tr:nth-child(odd)
* Represents the odd rows of an HTML table.
* tr:nth-child(2n)
* Represents the even rows of an HTML table.
* tr:nth-child(even)
* Represents the even rows of an HTML table.
* span:nth-child(0n+1)
* Represents a span element which is the first child of its parent; this is the same as the :first-child selector.
* span:nth-child(1)
* Equivalent to the above.
* span:nth-child(-n+3)
* Matches if the element is one of the first three children of its parent and also a span.

Remind about jQuery:

* jQuery start the index from 0

jQuery Features:

* Listen for a user to interact with page.
* Create animations in our page
* Communicate with a server without reloading the page.
* HTML document traversal and manipulation.
* Event handling.

Notable Features:

* $ instead of jQuery for the sake of brevity.

$(document).ready()

* Before we can safely use jQuery to do anything to our page, we need to ensure that the page is in a state where it's ready to be manipulated.