JQuery

DEESHA

Vishal Shah | M. 98508 10100
DEESHA COMPUTER EDUCATION | VISHUPRASAD APARTMENT, NR. DANDEKAR SHOPPE, VISHRAMBAG, SANGLI. EMAIL : MAIL.DEESHA@GMAIL.COM | PHONE : +91 233 6600371

Introduction

jQuery is a fast and concise JavaScript Library created by John Resig in 2006 with a nice motto − **Write less, do more**.

jQuery simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development.

jQuery is a JavaScript toolkit designed to simplify various tasks by writing less code. Here is the list of important core features supported by jQuery −

- **DOM manipulation** − The jQuery made it easy to select DOM elements, traverse them and modifying their content by using cross-browser open source selector engine called **Sizzle**.
- **Event handling** − The jQuery offers an elegant way to capture a wide variety of events, such as a user clicking on a link, without the need to clutter the HTML code itself with event handlers.
- **AJAX Support** − The jQuery helps you a lot to develop a responsive and feature-rich site using AJAX technology.
- **Animations** − The jQuery comes with plenty of built-in animation effects which you can use in your websites.
- **Lightweight** − The jQuery is very lightweight library - about 19KB in size (Minified and gzipped).
- **Cross Browser Support** − The jQuery has cross-browser support, and works well in IE 6.0+, FF 2.0+, Safari 3.0+, Chrome and Opera 9.0+
- **Latest Technology** − The jQuery supports CSS3 selectors and basic XPath syntax.

Why JQuery?

JQuery is fast, small, and feature-rich JavaScript Library. It makes things like HTML Document Traversal and manipulation, event handling, animation, and Ajax much simpler with an easy to use API that works across a multitude of browsers

Benefits of JQuery

- jQuery is an easy to learn JavaScript library which makes JavaScript programming very easy.
- 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 complicated tasks like AJAX calls and DOM manipulation.
- jQuery will run exactly the same and produce same output in all major browsers
- jQuery is free and very easy to include in your projects: just download its latest version from the jQuery website, or use an online Content Delivery Network
- jQuery is continuously upgraded, maintained and documented by a dedicated community of great developers.
- This ensures high quality and support on the internet.

How to use jQuery?

There are two ways to use jQuery.

- **Local Installation** – You can download jQuery library on your local machine and include it in your HTML code.
- **CDN Based Version** – You can include jQuery library into your HTML code directly from Content Delivery Network (CDN).

Local Installation

- Go to the https://jquery.com/download/ to download the latest version available.
- Now put downloaded **jquery-2.1.3.min.js** file in a directory of your website, e.g. /jquery.

Example

Now you can include *jquery* library in your HTML file as follows –

```
<html>

   <head>
      <title>The jQuery Example</title>
      <script type = "text/javascript"  src = "/jquery/jquery-2.1.3.min.js"></script>

      <script type = "text/javascript">
         $(document).ready(function(){
            document.write("Hello, World!");
         });
      </script>
   </head>

   <body>
      <h1>Hello</h1>
   </body>

</html>
```


CDN Based Version

You can include jQuery library into your HTML code directly from Content Delivery Network (CDN). Google and Microsoft provides content deliver for the latest version.

We are using Google CDN version of the library throughout this tutorial.

Example

Now let us rewrite above example using jQuery library from Google CDN.

```
<html>

   <head>
      <title>The jQuery Example</title>
      <script type = "text/javascript"
         src = "http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type = "text/javascript">
         $(document).ready(function(){
            document.write("Hello, World!");
         });
      </script>
   </head>

   <body>
      <h1>Hello</h1>
   </body>

</html>
```

How to call a jQuery library functions?

As almost everything we do when using jQuery reads or manipulates the document object model (DOM), we need to make sure that we start adding events etc. as soon as the DOM is ready.

If you want an event to work on your page, you should call it inside the $(document).ready() function. Everything inside it will load as soon as the DOM is loaded and before the page contents are loaded.

To do this, we register a ready event for the document as follows –

```
$(document).ready(function() {
   // do stuff when DOM is ready
});
```

```
<html>

   <head>
      <title>The jQuery Example</title>
      <script type = "text/javascript"
         src = "http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type = "text/javascript" language = "javascript">
         $(document).ready(function() {
            $("div").click(function() {alert("Hello, world!");});
         });
      </script>
   </head>
```

```
   <body>
      <div id = "mydiv">
         Click on this to see a dialogue box.
      </div>
   </body>

</html>
```

How to use Custom Scripts?

It is better to write our custom code in the custom JavaScript file : custom.js, as follows –

```
/* Filename: custom.js */
$(document).ready(function() {

   $("div").click(function() {
      alert("Hello, world!");
   });

});
```

Now we can include **custom.js** file in our HTML file as follows –

```
<html>

   <head>
      <title>The jQuery Example</title>
      <script type = "text/javascript"
         src = "http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type = "text/javascript" src = "/jquery/custom.js"></script>
   </head>

   <body>
      <div id = "mydiv">
         Click on this to see a dialogue box.
      </div>
   </body>

</html>
```

jQuery – Selectors

The jQuery library harnesses the power of Cascading Style Sheets (CSS) selectors to let us quickly and easily access elements or groups of elements in the Document Object Model (DOM).

A jQuery Selector is a function which makes use of expressions to find out matching elements from a DOM based on the given criteria. Simply you can say, selectors are used to select one or more HTML elements using jQuery. Once an element is selected then we can perform various operations on that selected element.

The $() factory function

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.

All the above items can be used either on their own or in combination with other selectors. All the jQuery selectors are based on the same principle except some tweaking.

NOTE – The factory function **$()** is a synonym of **jQuery()** function. So in case you are using any other JavaScript library where **$** sign is conflicting with some thing else then you can replace **$** sign by **jQuery** name and you can use function **jQuery()** instead of **$()**.

Example

Following is a simple example which makes use of Tag Selector. This would select all the elements with a tag name **p** and will set their background to "yellow".

```
<html>

   <head>
      <title>The jQuery Example</title>
```

```
      <script type = "text/javascript"
         src = "http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type = "text/javascript" language = "javascript">
         $(document).ready(function() {
            $("p").css("background-color", "yellow");
         });
      </script>
   </head>

   <body>
      <div>
         <p class = "myclass">This is a paragraph.</p>
         <p id = "myid">This is second paragraph.</p>
         <p>This is third paragraph.</p>
      </div>
   </body>

</html>
```

How to use Selectors?

The selectors are very useful and would be required at every step while using jQuery. They get the exact element that you want from your HTML document.

Following table lists down few basic selectors and explains them with examples.

S.N.	Selector & Description
1	**Name** Selects all elements which match with the given element Name.
2	**#ID** Selects a single element which matches with the given ID.
3	**.Class** Selects all elements which match with the given Class.
4	**Universal (*)** Selects all elements available in a DOM.
5	**Multiple Elements E, F, G** Selects the combined results of all the specified selectors E, F or G.

Examples

1. Selecting elements by id in jQuery

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery Select Element by ID</title>
  <script src="https://code.jquery.com/jquery-1.11.3.min.js"></script>
  <script type="text/javascript">
          $(document).ready(function(){
                  // Highlight element with id mark
                  $("#mark").css("background", "yellow");
          });
</script>
</head>
<body>
    <p id="mark">This is a paragraph.</p>
    <p>This is another paragraph.</p>
    <p>This is one more paragraph.</p>
    <p><strong>Note:</strong> The value of the id attribute must be unique in an HTML
document.</p>
</body>
</html>
```


2. Selecting elements by element class in jQuery

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery Select Element by Class</title>
<script src="https://code.jquery.com/jquery-1.11.3.min.js"></script>
  <script type="text/javascript">
          $(document).ready(function(){
              // Highlight elements with class mark
                  $(".mark").css("background", "yellow");
          });
</script>
</head>
<body>
    <p class="mark">This is a paragraph.</p>
    <p class="mark">This is another paragraph.</p>
    <p>This is one more paragraph.</p>
</body>
</html>
```

3. Selecting elements by element name

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery Select Element by Name</title>
<script src="https://code.jquery.com/jquery-1.11.3.min.js"></script>
  <script type="text/javascript">
          $(document).ready(function(){
            // Highlight paragraph elements
                  $("p").css("background", "yellow");
          });
</script>
</head>
<body>
    <h1>This is heading</h1>
    <p>This is a paragraph.</p>
    <p>This is another paragraph.</p>
    <div>This is another block of text.</div>
</body>
</html>
```


4. Selecting elements by attribute in jQuery

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery Select Element by Attribute</title>
<script src="https://code.jquery.com/jquery-1.11.3.min.js"></script>
  <script type="text/javascript">
          $(document).ready(function(){
                  // Highlight paragraph elements
                  $('input[type="text"]').css("background", "yellow");
          });
</script>
</head>
<body>
    <form>
        <label>Name: <input type="text"></label>
        <label>Password: <input type="password"></label>
        <input type="submit" value="Sign In">
    </form>
</body>
</html>
```

5. Selecting elements by compound CSS selector in jQuery

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery Select Element by Compound Selector</title>
<script src="https://code.jquery.com/jquery-1.11.3.min.js"></script>
<script type="text/javascript">
$(document).ready(function(){
    // Highlight only paragraph elements with class mark
    $("p.mark").css("background", "yellow");

    // Highlight only span elements inside the element with ID mark
    $("#mark span").css("background", "yellow");

    // Highlight li elements inside the ul elements
    $("ul li").css("background", "yellow");

    // Highlight li elements only inside the ul element with id mark
    $("ul#mark li").css("background", "red");

    // Highlight li elements inside all the ul element with class mark
    $("ul.mark li").css("background", "green");

    // Highlight all anchor elements with target blank
    $('a[target="_blank"]').css("background", "yellow");
});
</script>
</head>
<body>
    <p>This is a paragraph.</p>
    <p>This is another paragraph.</p>
    <p>This is one more paragraph.</p>
    <ul>
        <li>List item one</li>
        <li>List item two</li>
        <li>List item three</li>
    </ul>
    <ul id="mark">
        <li>List item one</li>
        <li>List item two</li>
        <li>List item three</li>
    </ul>
    <ul class="mark">
        <li>List item one</li>
        <li>List item two</li>
```

```
        <li>List item three</li>
    </ul>
    <p>Go to <a href="#">Home page</a></p>
</body>
</html>
```

6. Selecting elements by jQuery custom selector

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery Custom Selector</title>
<style type="text/css">
    /* Some custom style */
    *{
        padding: 5px;
    }
</style>
<script src="https://code.jquery.com/jquery-1.11.3.min.js"></script>
<script type="text/javascript">
$(document).ready(function(){
    // Highlight table rows appearing at odd places
    $("tr:odd").css("background", "yellow");

    // Highlight table rows appearing at even places
    $("tr:even").css("background", "orange");

    // Highlight first paragraph element
    $("p:first").css("background", "red");

    // Highlight last paragraph element
    $("p:last").css("background", "green");

    // Highlight all input elements with type text inside a form
    $("form :text").css("background", "purple");

    // Highlight all input elements with type password inside a form
    $("form :password").css("background", "blue");

    // Highlight all input elements with type submit inside a form
    $("form :submit").css("background", "violet");
});
</script>
</head>
<body>
    <table border="1">
        <thead>
```

```
            <tr>
                <th>No.</th>
                <th>Name</th>
                <th>Email</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>1</td>
                <td>John Carter</td>
                <td>johncarter@mail.com</td>
            </tr>
            <tr>
                <td>2</td>
                <td>Peter Parker</td>
                <td>peterparker@mail.com</td>
            </tr>
            <tr>
                <td>3</td>
                <td>John Rambo</td>
                <td>johnrambo@mail.com</td>
            </tr>
        </tbody>
    </table>
    <p>This is a paragraph.</p>
    <p>This is another paragraph.</p>
    <p>This is one more paragraph.</p>
    <form>
        <label>Name: <input type="text"></label>
        <label>Password: <input type="password"></label>
        <input type="submit" value="Sign In">
    </form>
</body>
</html>
```


Attribute Selectors

The attribute selectors enable you to select elements based on what attributes the elements have, and even based on attribute values. There are several different attribute selectors which I will cover below.

Has Attribute Selector

The "has attribute" selector enables you to select all elements which have a certain attribute, regardless of that attribute's value.

To specify an attribute selector you write the attribute name inside square brackets in the selector string. Here is a jQuery attribute selector example (has attribute):

```
$('[height]');
```

This example will select all elements that has a `height` attribute.

You can combine the attribute selector with other selectors, e.g. the element name selector selector to select all `div` elements having the attribute `height`. Here is a element name and has attribute selector example:

```
$('div[height]');
```

Attribute Value Equals Selector

The attribute value selector enables you to select elements based on specific attribute values. Here is a jQuery attribute value equals selector example:

```
$('div[height=200]');
```

This example selects all `div` elements with a `height` attribute value of `200`.

You can also enclose the value in quotes, like this:

```
$('div[height="200"]');
```

Attribute Value Not Equals Selector

The attribute value selector also enables you to select elements based on attribute value not being equal to a certain value. This is done by putting a `!` in front of the `=` in the select expression. Here is an example:

```
$('div[height!=200]');
```

This example selects all `div` elements that do not have an attribute value of `200`. This also includes all `div`elements without any `height` attribute at all.

Attribute Value Prefix Selector

The attribute value prefix selector enables you to select elements which contain an attribute with a value that starts with a given value. In other words, where the attribute value contains a certain *prefix*. Here is a jQuery attribute value prefix selector example:

```
$('[href|="http://"]');
```

This example selects all elements with a `href` attribute where the value starts with `http://` .

Attribute Value Contains Selector

The attribute value contains selector enables you to select elements with an attribute where the value contains a certain substring. Here is a jQuery attribute value contains example:

```
$('[href*="/download/"]');
```

This example selects all elements which contains the substring `/download/` in their `href` attribute values.

Attribute Value Ends With Selector

The attribute ends with selector enables you to select elements with an attribute where the value ends with a certain substring. Here is a jQuery attribute value ends with selector example:

```
$('[href$=".pdf"]');
```

This example selects all elements which have an `href` attribute where the value ends with `.pdf`.

Element Visibility Selector

The element visibility selector enables you to select elements based on whether they are visible or not. jQuery contains the following element visibility selectors:

- `:hidden`
- `:visible`

Here is a jQuery visibility selector example:

```
$(':visible');

$(':hidden');
```

The first line selects all visible elements in the page. The second line selects all hidden elements in the page.

Element Order Selectors

The element order selector makes it possible to select element based on the order in which they appear in the HTML page (in the DOM). For instance, you can select the third `div` element in the page, or all `odd` `div`elements etc.

Here is a jQuery element order selector example:

```
$('div:first').text();
```

This example selects the first `div` element in the page. The order is written after the `:` in the jQuery selector parameter.

You can use the following order parameters:

- `:first`
- `:last`
- `:even`
- `:odd`
- `:eq(index)`
- `:gt(index)`
- `:lt(index)`

:first selects the first element of the given kind.

:last selects the last element of the given kind.

:even selects the even elements of the given kind, meaning the elements that have index 0, 2, 4 etc.

:odd selects the odd elements of the given kind, meaning the elements that have index 1, 3, 5 etc.

:eq() selects the element having the given index, starting from 0. For instance, `'div:eq(3)'` selects the 4th`div` element in the page.

:gt() selects all elements with index greater than the given index. For instance, `'div:gt(1)'` selects all `div`elements with indexes greater than 1.

:lt() selects all elements with index less than the given index. For instance, `'div:lt(2)'` selects all `div` elements with indexes less than 2.

Form Field Selectors

jQuery contains a set of form field selectors which makes it easier to select the various different types of form fields from the DOM. In this section I will cover the most common jQuery form field selectors.

:input Selector

The `:input` selector enable you to easily select `input` elements. Here is a jQuery `:input` example:

```
$(":input");
```

This example selects all `input` fields in the page. Notice the `:` before the text `input`. This colon is necessary to not confuse it with a selection based on element name. In this example there might not have been any difference between `input` and `:input`, but for several of the form field selectors, there is a difference.

:text Selector

`:text` selects all input fields of type `text`. It does not select `textarea` elements. Here is a jQuery `:text`selector example:

```
$(":text");       $("input[type=text]")
```

:radio Selector

The `:radio` selects all input fields of type `radio`. That is, all radio button elements. Here is a jQuery `:radio`selector example:

```
$(":radio");
```

:checkbox Selector

The `:checkbox` selects all input fields of type `checkbox`. All checkbox elements, in other words. Here is a jQuery `:checkbox` selector example:

```
$(": checkbox");
```

The `:checked` selects all input fields of type `checkbox` or `radio` which are checked.

:password Selector

The `:password` selector selects all `input` elements of type `password`. Here is a jQuery `:password` selector example:

```
$(":password");
```


:submit Selector

The `:submit` selector selects all `input` elements of type `submit`. That is, all form submit buttons. Here is a jQuery `:submit` selector example:

```
$(":submit");
```

:reset Selector

The `:reset` selector selects all `input` elements of type `reset`. That is, all form reset buttons. Here is a jQuery `:reset` selector example:

```
$(":reset");
```

:file Selector

The :file selector selects all input elements of type file. That is, all file upload fields. **Here is a** jQuery:file selector example:

```
$(":file");
```

:Button Selector

The :button selector selects all button elements and input elements of type button. It does not select inputelements of type submit. Here is a jQuery :button selector example:

```
$(":button");
```

Enabled and Disabled Element Selectors

The `:enabled` selector selects all elements that are enabled. Likewise, the `:disabled` selector selects all elements that are disabled. Here are a jQuery `:enabled` and `:disabled` selector example:

```
$(":enabled");
$(":disabled");
```

Empty Element Selector

The `:empty` element selector selects all elements which are empty, meaning they have no child elements and no text inside its body (attributes are allowed). Here is a jQuery `:empty` selector example:

```
$(":empty");
```

Contains Text Selector

The `:contains` selector is used to select elements which contains a certain text. Here is a jQuery `:contains` example:

```
$(":contains(Hello World)");
```

This example selects all elements which contain the text "Hello World".

Target Element Selector

The `:target` selector selects the element which is the HTML fragment target of the current URL. For instance, if the URL in the browser is:

```
http://myapp.com/somepage.html#someElement
```

Then the `:target` selector will select the element with the id (or `name`) `someElement`, because the HTML fragment part of the URL points is `#someElement` .

Here is a jQuery `:target` selector example:

```
$(":target");
```

All Elements Selector

The `*` (all elements) selector selects all elements in the HTML document. Since you won't often need that, the `*` selector is often combined with other selectors. Here is an example of the element name selector and parent / child selector combined with the `*` selector:

```
$("div>*");
```

This example selects all immediate children of all `div` elements. The `div` part indicates all `div` elements. The `>` part indicates immediate children of what is selected on the left (which is all `div` elements). The `*`means all elements. In total, that means all immediate child elements of all `div` elements.

Parents and Child Selectors

The parent and child selectors enable you to select element based on what parents or ancestors they have, or their order inside their parent elements. You can use the following parent and child selectors to select elements based on their parent - child relationship:

- `:first-child`
- `:last-child`
- `parent>child`

The `:first-child` selector lets you select the first child of a parent element. Here is jQuery `:first-child`example:

```
$('div:first-child');
```

This example selects the first child of all `div` elements.

The `:last-child` selector works the same way as `:first-child` except it selects the last child of the parent.

The `parent>child` selector lets you select children of a specific parent element. Here is a jQuery`parent>child` example:

```
$('div>p');
```

This example will select all `p` elements inside `div` elements. `p` elements outside `div` elements are not selected.

Combining Selectors

You can combine many of the jQuery selectors to create even more powerful selector expressions. To achieve full mastery, you must experiment with combinations yourself, but here I will give you a few examples, so you can get a feeling for how it works.

Here is an example that selects all `p` elements inside `div` elements, of `div` elements having the attribute`height` set to `300`:

```
$('div[height=300]>p');
```

The next example enhances the first example, by adding the criteria, that only `p` elements with the CSS class `go` should be selected:

```
$('div[height=300]>p.go');
```

The third example modifies the previous example, by only selecting the `p` element with the `id` `p_id`:

```
$('div[height=300]>p#p_id');
```

Selector Summary

:last	$("p:last")	The last <p> element
:even	$("tr:even")	All even <tr> elements
:odd	$("tr:odd")	All odd <tr> elements

:first-child	$("p:first-child")	All <p> elements that are the first child of their parent
:first-of-type	$("p:first-of-type")	All <p> elements that are the first <p> element of their parent
:last-child	$("p:last-child")	All <p> elements that are the last child of their parent
:last-of-type	$("p:last-of-type")	All <p> elements that are the last <p> element of their parent
:nth-child(*n*)	$("p:nth-child(2)")	All <p> elements that are the 2nd child of their parent
:nth-last-child(*n*)	$("p:nth-last-child(2)")	All <p> elements that are the 2nd child of their parent, counting from the last child
:nth-of-type(*n*)	$("p:nth-of-type(2)")	All <p> elements that are the 2nd <p> element of their parent
:nth-last-of-type(*n*)	$("p:nth-last-of-type(2)")	All <p> elements that are the 2nd <p> element of their parent, counting from the last child
:only-child	$("p:only-child")	All <p> elements that are the only child of their parent
:only-of-type	$("p:only-of-type")	All <p> elements that are the only child, of its type, of their parent
parent > child	$("div > p")	All <p> elements that are a direct child of a element
parent descendant	$("div p")	All <p> elements that are descendants of a element
element + next	$("div + p")	The <p> element that are next to each elements
element ~ siblings	$("div ~ p")	All <p> elements that are siblings of a element
:eq(*index*)	$("ul li:eq(3)")	The fourth element in a list (index starts at 0)
:gt(*no*)	$("ul li:gt(3)")	List elements with an index greater than 3
:lt(*no*)	$("ul li:lt(3)")	List elements with an index less than 3
:not(*selector*)	$("input:not(:empty)")	All input elements that are not empty
:header	$(":header")	All header elements <h1>, <h2> ...
:animated	$(":animated")	All animated elements
:focus	$(":focus")	The element that currently has focus
:contains(*text*)	$(":contains('Hello')")	All elements which contains the text "Hello"
:has(*selector*)	$("div:has(p)")	All elements that have a <p> element
:empty	$(":empty")	All elements that are empty
:parent	$(":parent")	All elements that are a parent of another element
:hidden	$("p:hidden")	All hidden <p> elements

:visible	$("table:visible")	All visible tables
:root	$(":root")	The document's root element
:lang(*language*)	$("p:lang(de)")	All <p> elements with a lang attribute value starting with "de"
[*attribute*]	$("[href]")	All elements with a href attribute
[*attribute*=*value*]	$("[href='default.htm']")	All elements with a href attribute value equal to "default.htm"
[*attribute*!=*value*]	$("[href!='default.htm']")	All elements with a href attribute value not equal to "default.htm"
[*attribute*$=*value*]	$("[href$='.jpg']")	All elements with a href attribute value ending with ".jpg"
[*attribute*|=*value*]	$("[title|='Tomorrow']")	All elements with a title attribute value equal to 'Tomorrow', or starting with 'Tomorrow' followed by a hyphen
[*attribute*^=*value*]	$("[title^='Tom']")	All elements with a title attribute value starting with "Tom"
[*attribute*~=*value*]	$("[title~='hello']")	All elements with a title attribute value containing the specific word "hello"
[*attribute=*value*]**	$("[title*='hello']")	All elements with a title attribute value containing the word "hello"
:input	$(":input")	All input elements
:text	$(":text")	All input elements with type="text"
:password	$(":password")	All input elements with type="password"
:radio	$(":radio")	All input elements with type="radio"
:checkbox	$(":checkbox")	All input elements with type="checkbox"
:submit	$(":submit")	All input elements with type="submit"
:reset	$(":reset")	All input elements with type="reset"
:button	$(":button")	All input elements with type="button"
:image	$(":image")	All input elements with type="image"
:file	$(":file")	All input elements with type="file"
:enabled	$(":enabled")	All enabled input elements
:disabled	$(":disabled")	All disabled input elements
:selected	$(":selected")	All selected input elements
:checked	$(":checked")	All checked input elements

DOM Traversing

jQuery traversing means "move through", are used to "find" (or select) HTML elements based on their relation to other elements. Start with one selection and move through that selection until you reach the elements you desire.

The image below illustrates a family tree. With jQuery traversing, you can easily move up (ancestors), down (descendants) and sideways (siblings) in the family tree, starting from the selected (current) element. This movement is called traversing - or moving through - the DOM.

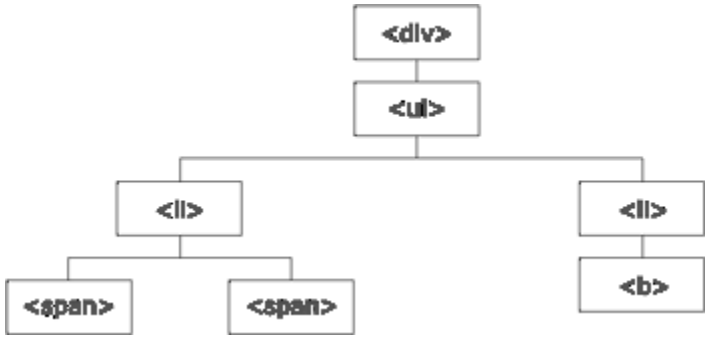

Illustration explained:

- The element is the **parent** of <ul>, and an **ancestor** of everything inside of it
- The <ul> element is the **parent** of both <li> elements, and a **child** of
- The left <li> element is the **parent** of , **child** of <ul> and a **descendant** of
- The element is a **child** of the left <li> and a **descendant** of <ul> and
- The two <li> elements are **siblings** (they share the same parent)
- The right <li> element is the **parent** of <b>, **child** of <ul> and a **descendant** of
- The <b> element is a **child** of the right <li> and a **descendant** of <ul> and

- An ancestor is a parent, grandparent, great-grandparent, and so on.
- A descendant is a child, grandchild, great-grandchild, and so on.
- Siblings share the same parent.

Most of the DOM Traversal Methods do not modify the jQuery object and they are used to filter out elements from a document based on given conditions.

Traversing Up the DOM Tree

Three useful jQuery methods for traversing up the DOM tree are:

- parent()
- parents()

- parentsUntil()

jQuery parent() Method

- The parent() method returns the direct parent element of the selected element.
- This method only traverse a single level up the DOM tree.
- The following example returns the direct parent element of each elements:

Example

```
$(document).ready(function(){
    $("span").parent();
});
```

jQuery parents() Method

The parents() method returns all ancestor elements of the selected element, all the way up to the document's root element (<html>).

```
$(document).ready(function(){
    $("span").parents();
});
```


jQuery parentsUntil() Method

The parentsUntil() method returns all ancestor elements between two given arguments.

```
$(document).ready(function(){
    $("span").parentsUntil("div");
});
```

Traversing Down the DOM Tree

Two useful jQuery methods for traversing down the DOM tree are:

- children()
- find()

jQuery children() Method

The children() method returns all direct children of the selected element.

This method only traverse a single level down the DOM tree.

The following example returns all elements that are direct children of each elements:

Example

```
$(document).ready(function(){
    $("div").children();
});
```

You can also use an optional parameter to filter the search for children.

The following example returns all <p> elements with the class name "first", that are direct children of :

Example

```
$(document).ready(function(){
    $("div").children("p.first");
});
```

jQuery find() Method

The find() method returns descendant elements of the selected element, all the way down to the last descendant.

The following example returns all elements that are descendants of :

Example

```
$(document).ready(function(){
    $("div").find("span");
});
```


Traversing Sideways in The DOM Tree

There are many useful jQuery methods for traversing sideways in the DOM tree:

- siblings()
- next()
- nextAll()
- nextUntil()
- prev()
- prevAll()
- prevUntil()

jQuery siblings() Method

The siblings() method returns all sibling elements of the selected element.

The following example returns all sibling elements of <h2>:

Example

```
$(document).ready(function(){
    $("h2").siblings();
});
```

You can also use an optional parameter to filter the search for siblings.

The following example returns all sibling elements of <h2> that are <p> elements:

Example

```
$(document).ready(function(){
    $("h2").siblings("p");
});
```

jQuery next() Method

The next() method returns the next sibling element of the selected element.

The following example returns the next sibling of <h2>:

Example

```
$(document).ready(function(){
    $("h2").next();
});
```


jQuery nextAll() Method

The nextAll() method returns all next sibling elements of the selected element.

The following example returns all next sibling elements of <h2>:

Example

```
$(document).ready(function(){
    $("h2").nextAll();
});
```

jQuery nextUntil() Method

The nextUntil() method returns all next sibling elements between two given arguments.

The following example returns all sibling elements between a <h2> and a <h6> element:

Example

```
$(document).ready(function(){
    $("h2").nextUntil("h6");
});
```

jQuery prev(), prevAll() & prevUntil() Methods

The prev(), prevAll() and prevUntil() methods work just like the methods above but with reverse functionality: they return previous sibling elements (traverse backwards along sibling elements in the DOM tree, instead of forward).

```
<!DOCTYPE html>
<html>
<head>
<style>
.siblings * {
    display: block;
    border: 2px solid lightgrey;
    color: lightgrey;
    padding: 5px;
    margin: 15px;
}
</style>
<script src="jquery.min.js"></script>
<script>
$(document).ready(function(){
    $('h2').nextUntil('h6').css("color","red");
});
</script>
</head>
<body class="siblings">

<div>div (parent)
  <p>p</p>
  <span>span</span>
  <h2>h2</h2>
  <h3>h3</h3>
  <h4>h4</h4>
  <h5>h5</h5>
  <h6>h6</h6>
  <p>p</p>
</div>

</body>
</html>
```

Find Elements by index

Consider a simple document with the following HTML content –

```
<html>

   <head>
      <title>The JQuery Example</title>
   </head>

   <body>

      <div>
         <ul>
            <li>list item 1</li>
            <li>list item 2</li>
            <li>list item 3</li>
            <li>list item 4</li>
            <li>list item 5</li>
            <li>list item 6</li>
         </ul>
      </div>

   </body>

</html>
```


This will produce following result –

- Above every list has its own index, and can be located directly by using **eq(index)** method as below example.
- Every child element starts its index from zero, thus, *list item 2* would be accessed by using **$("li").eq(1)** and so on.

Example

Following is a simple example which adds the color to second list item.

```
<html>

   <head>
      <title>The JQuery Example</title>
      <script type = "text/javascript"
         src = "http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type = "text/javascript" language = "javascript">
         $(document).ready(function() {
            $("li").eq(2).addClass("selected");
         });
      </script>

      <style>
         .selected { color:red; }
      </style>
   </head>

   <body>

      <div>
         <ul>
            <li>list item 1</li>
            <li>list item 2</li>
            <li>list item 3</li>
            <li>list item 4</li>
            <li>list item 5</li>
            <li>list item 6</li>
         </ul>
      </div>

   </body>

</html>
```


Filtering out Elements

The **filter (selector)** method can be used to filter out all elements from the set of matched elements that do not match the specified selector(s). The *selector* can be written using any selector syntax.

Example

Following is a simple example which applies color to the lists associated with middle class –

```
<html>
```

```
<head>
   <title>The JQuery Example</title>
   <script type = "text/javascript"
      src = "http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

   <script type = "text/javascript" language = "javascript">
      $(document).ready(function() {
         $("li").filter(".middle").addClass("selected");
      });
   </script>

   <style>
      .selected { color:red; }
   </style>
</head>

<body>

   <div>
      <ul>
         <li class = "top">list item 1</li>
         <li class = "top">list item 2</li>
         <li class = "middle">list item 3</li>
         <li class = "middle">list item 4</li>
         <li class = "bottom">list item 5</li>
         <li class = "bottom">list item 6</li>
      </ul>
   </div>

</body>

</html>
```


Locating descendant Elements

The **find (selector)** method can be used to locate all the descendant elements of a particular type of elements. The *selector* can be written using any selector syntax.

Example

Following is an example which selects all the elements available inside different <p> elements –

```
<html>

   <head>
      <title>The JQuery Example</title>
      <script type = "text/javascript"
         src = "http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type = "text/javascript" language = "javascript">
         $(document).ready(function() {
            $("p").find("span").addClass("selected");
         });
```

```
      </script>

      <style>
         .selected { color:red; }
      </style>
   </head>

   <body>
      <p>This is 1st paragraph and <span>THIS IS RED</span></p>
      <p>This is 2nd paragraph and <span>THIS IS ALSO RED</span></p>
   </body>

</html>
```

JQuery DOM Filter Methods

Following table lists down useful methods which you can use to filter out various elements from a list of DOM elements –

S.N.	Method & Description
1	**eq(index)** Reduce the set of matched elements to a single element.
2	**filter(selector)** Removes all elements from the set of matched elements that do not match the specified selector(s).
3	**filter(fn)** Removes all elements from the set of matched elements that do not match the specified function.
4	**is(selector)** Checks the current selection against an expression and returns true, if at least one element of the selection fits the given selector.
5	**map(callback)** Translate a set of elements in the jQuery object into another set of values in a jQuery array (which may, or may not contain elements).
6	**not(selector)** Removes elements matching the specified selector from the set of matched elements.
7	**slice(start, [end])** Selects a subset of the matched elements.

JQuery DOM Traversing Methods

Following table lists down other useful methods which you can use to locate various elements in a DOM –

S.N.	Methods & Description
1	**add(selector)** Adds more elements, matched by the given selector, to the set of matched elements.
2	**andSelf( )** Add the previous selection to the current selection.
3	**children([selector])**

	Get a set of elements containing all of the unique immediate children of each of the matched set of elements.
4	**closest(selector)** Get a set of elements containing the closest parent element that matches the specified selector, the starting element included.
5	**contents( )** Find all the child nodes inside the matched elements (including text nodes), or the content document, if the element is an iframe.
6	**end( )** Revert the most recent 'destructive' operation, changing the set of matched elements to its previous state.
7	**find(selector)** Searches for descendant elements that match the specified selectors.
8	**next([selector])** Get a set of elements containing the unique next siblings of each of the given set of elements.
9	**nextAll([selector])** Find all sibling elements after the current element.
10	**offsetParent( )** Returns a jQuery collection with the positioned parent of the first matched element.
11	**parent([selector])** Get the direct parent of an element. If called on a set of elements, parent returns a set of their unique direct parent elements.
12	**parents([selector])** Get a set of elements containing the unique ancestors of the matched set of elements (except for the root element).
13	**prev([selector])** Get a set of elements containing the unique previous siblings of each of the matched set of elements.
14	**prevAll([selector])** Find all sibling elements in front of the current element.
15	**siblings([selector])** Get a set of elements containing all of the unique siblings of each of the matched set of elements.

$("div"). length: Returns number of selected elements. It is the best way to check selector.

```
<script type="text/javascript">

        $(document).ready(function () {
                alert($("div").length); //output - 3
        });
</script>
<div>Div1</div>
<div>Div2</div>
<div>Div3</div>
```

Getting a specific DOM element

$("div").**get**(2) or $("div")[2] - Returns a 3rd DOM (html) element of the selection

```
<script type="text/javascript">
    $(document).ready(function () {
         alert($($("div").get(2)).html());
    });
</script>
<div>Div1</div>
<div>Div2</div>
<div>Div3</div>
```

Getting a specific jQuery element

$("div").**eq**(2) - Returns a 2nd jQuery element of the selection

Other similar methods: **first(), last(),**

```
<script type="text/javascript">
         $(document).ready(function () {
                  alert($("div").eq(1).html());
                  alert($("div").first().html());
                  alert($("div").last().html());
    });
</script>
<div>Div1</div>
<div>Div2</div>
<div>Div3</div>
```


Each() method: traverses every selected element calling the mentioned function

```
$(document).ready(function () {
         var sum = 0;
         $("div.number").each(function()
         {
                  sum += (+this.innerHTML);
         })
    alert(sum)
});

<div class="number">10</div>
<div class="number">12</div>
<div class="number">15</div>
```

Each() method also passes an indexer

```
$("table tr").each(function (ind) { // ind  - index of the current element
         if (ind % 2)
                  $(this).addClass("odd");
```

```
});
```

Filter method: lets us to specify a criteria and those that match will be returned.

```
$(document).ready(function () {
//To get all <p> elements which have intro as class name
    $("p").filter(".intro").each(function () {
         alert($(this).html())
    });

//To get all <p> elements which don't have intro as class name
   $("p").not(".intro").each(function () {
         alert($(this).html())
   });
});

<h1>Welcome to My Homepage</h1>
<p>My name is Sandeep.</p>
<p class="intro">I live in Hyderabad.</p>
<p class="intro">I love India.</p>
<p>My best friend is Rahul.</p>
```


.children method: Get all the direct children of selected element

```
<script type="text/javascript">
    $(document).ready(function () {
        $("#DemoDiv1").children("span").each(function () {
          //if filter is not provided it returns all children
                alert($(this).html());
        });
    });
</script>
<div id="DemoDiv1">
    <span>span1 </span>
    <p>This is paragraph</p>
    <span>span2 </span>
</div>
```

find method: returns descendant elements of the selected element, all the way down to the last descendant

$("p").find(".header").show(); - Select paragraph and then find elements with class 'header' inside Finding a span inside a div.

```
<script type="text/javascript">
    $(document).ready(function () {
        $("div").find("span").each(function () {
```

```
            alert($(this).html());
        });
    });
</script>
<div>
    <span>This is span1 </span>
    <div>This is div1
        <span>This is span2 </span>
    </div>
    <div>This is div2 </div>
</div>
```

parent method:

```
<script type="text/javascript">
    $(document).ready(function () {
        $("#span1").parent().each(function () {
         alert($(this).html());
        });
    });
</script>
<div id="DemoDiv1">
    <span id="Span1">span1 </span><span>span2 </span>
</div>
```


parents method:

```
 <script>
    $(document).ready(function () {
        $("span").parents().css({ "color": "green", "border": "2px solid red" });
        $("span").parents("ul").css({ "color": "yellow", "border": "2px solid red" });
    });
```

DOM Manipulation

JQuery provides methods to manipulate DOM in efficient way. You do not need to write big code to modify the value of any element's attribute or to extract HTML code from a paragraph or division.

JQuery provides methods such as .attr(), .html(), and .val() which act as getters, retrieving information from DOM elements for later use.

jQuery html()

jQuery html() method is used to change the entire content of the selected elements. It replaces the selected element content with new contents.

Note: It is a very useful function but works in a limited area because of its API documentation. The API documentation of the jQuery html function consists of three method signatures.

The first method signature has no argument, so it just returns the HTML within that element. The remaining two signatures take a single argument: i.e. a string or a function that returns a string.

Syntax:

- $(selector).html() : It is used to return content.
- $(selector).html(content) : It is used to set content.
- $(selector).html(function (index, currentcontent): It is used to set content by calling function.

The jQuery html() method is used either for set the content or return the content of the selected elements.

- **To set content**: When you use this method to set content, it overwrites the content of the all matched elements.
- **To return content**: When you use this method to return content, it returns the content of the first matched element.

Example

Following is an example which makes use of .html() and .text(val) methods. Here .html() retrieves HTML content from the object and then .text(val) method sets value of the object using passed parameter –

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type = "text/javascript"
         src = "http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type = "text/javascript" language = "javascript">
         $(document).ready(function() {
            $("div").click(function () {
               var content = $(this).html();
               $("#result").text( content );
            });
         });
      </script>

      <style>
         #division{ margin:10px;padding:12px; border:2px solid #666; width:60px;}
      </style>
   </head>
   <body>
      <p>Click on the square below:</p>
      <span id = "result"> </span>

      <div id = "division" style = "background-color:blue;">
```

```
        This is Blue Square!!
      </div>
   </body>
</html>
```

jQuery text()

The jQuery text() method is used to set or return the text content of the selected elements.

- **To return content:** When this method is used to return content, it returns the combined text content of all matched elements without the HTML markup.
- **To set content:** When this method is used to set content, it overwrites the content of all matched elements.

Difference between jQuery text() method and jQuery html() method

Sometimes, this confusion is occurred because both of the methods are used to set or return the html content. But, the jQuery text() method is different from html() method.

Following is the main differences:

- The jQuery text() method is used to set or return html content without HTML markup while, html() method is used to set or return the innerHtml (text + HTML markup).
- The jQuery text() method can be used in both XML and HTML document while jQuery html() method can't.

jQuery val()

There are two usage of jQuery val() method.

- It is used to get current value of the first element in the set of matched elements.
- It is used to set the value of every matched element.

Syntax:

$(selector).val() : It is used to get value.

$(selector).val(value) : It is used to set value.

jQuery css()

The jQuery CSS() method is used to get (return)or set style properties or values for selected elements. It facilitates you to get one or more style properties.

jQuery CSS() method provides two ways:

- It is used to get the value of a specified CSS property.
- This property is used to set a specific value for all matched element.

Set multiple CSS properties

css({"propertyname":"value","propertyname":"value",...});

jQuery attr()

The jQuery attr() method is used to set or return attributes and values of the selected elements.

There are two usage of jQuery attr() method.

- **To return attribute value**: This method returns the value of the first matched element.
- **To set attribute value**: This method is used to set one or more attribute/value pairs of the set of matched elements.

jQuery prop()

jQuery prop() method is used for two purpose.

- It is used to **return** the value of a property for the first element in a set of matched elements.
- It is used to **set** one or more property value for a set of matched element.

The jQuery prop() method is generally used to retrieve property values i.e. DOM properties (like tagName, nodeName, defaultChecked) or own custom made properties. This is a very convenient way to set the values of properties, especially the multiple properties.

If you want to retrieve HTML attributes, you should use the attr() method instead.

The removeProp() method is used to remove a property.

```
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>prop demo</title>
  <style>
  p {
    margin: 20px 0 0;
  }
  b {
    color: red;
  }
```

```
  </style>
  <script src="https://code.jquery.com/jquery-1.10.2.js"></script>
</head>
<body>
<input id="check1" type="checkbox" checked="checked">
<label for="check1">Check me</label>
<p></p>
<script>
$( "input" ).change(function() {
  var $input = $( this );
  $( "p" ).html(
    ".attr( \"checked\" ): <b>" + $input.attr( "checked" ) + "</b><br>" +
    ".prop( \"checked\" ): <b>" + $input.prop( "checked" ) + "</b><br>" +
    ".is( \":checked\" ): <b>" + $input.is( ":checked" ) ) + "</b>";
}).change();
</script>
</body>
</html>
```

Difference between jQuery attr() and jQuery prop() method:

This is a very common question because most of the people are confused about where to use prop() method and where attr() method. The differences between them are very important in specific situation.

Following is the exact differences between them:

- The jQuery attr() method is used to retrieve the HTML attribute values while jQuery prop() method is used to retrieve the property values.
- The attr() method changes the attribute of the HTML tag while the prop() method changes a property for the HTML tag as per the DOM tree.
- Properties are generally simpler to deal with than attributes so the jQuery prop() method is mostly used rather than attr() method.

jQuery append()

The jQuery append() method is used to insert specified content as the last child (at the end of) the selected elements in the jQuery collection.

The append () and appendTo () methods are used to perform the same task. The only difference between them is in the syntax.

```
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("#btn1").click(function(){
        $("p").append(" <b>Newly added appended text</b>.");
    });
    $("#btn2").click(function(){
        $("ol").append("<li><b>Newly added appended item</b></li>");
    });
});
</script>
</head>
<body>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<ol>
  <li>Item no.1</li>
  <li>Item no.2</li>
  <li>Item no.3</li>
</ol>
<button id="btn1">Append text</button>
<button id="btn2">Append item</button>
</body>
</html>
```


append(), prepend() Examples

```
// select & append to the beginning
// select & append to the end
$("h1").append("<li>Hello $!</li>");
$("ul").prepend("<li>Hello $!</li>");
// create & append/prepend to selector
$("<span>Hello World!</span>").appendTo("p"); //Insert span element at the end of each p element
$("<span>Hello World!</span>").prependTo("p"); //Insert a <span> element at the beginning of each <p>
element
```

```
$("img").after("Some text after");
$("img").before("Some text before");
```

jQuery addClass()

The addclass() method is used to add one or more class name to the selected element. This method is used only to add one or more class names to the class attributes not to remove the existing class attributes.

If you want to add more than one class separate the class names with spaces.

```
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("p:first").addClass("intro");
    });
});
</script>
<style>
.intro {
    font-size: 200%;
    color: red;
}
</style>
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<button>Add a class name to the first p element</button>
</body>
</html>
```


jQuery toggleClass()

The jQuery toggleCLass() method is used to add or remove one or more classes from the selected elements. This method toggles between adding and removing one or more class name. It checks each element for the specified class names. If the class name is already set, it removes and if the class name is missing, it adds.

In this way, it creates the toggle effect. It also facilitates you to specify to only add or only remove by the use of switch parameter.

```
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
```

```
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("p").toggleClass("main");
    });
});
</script>
<style>
.main {
    font-size: 150%;
    color: red;
}
</style>
</head>
<body>
<button>Toggle class "main" for p elements</button>
<p>Hello! javatpoint.com</p>
<p>This is popular tutorial website.</p>
<p><b>Note:</b> Click repeatedly on the button to see the toggle effect.</p>
</body>
</html>
```

addClass(), removeClass(), hasClass() Example


```
// add and remove class
$("h1, h2, p").addClass("blue");
//add multiple classes
$("#div1").addClass("important blue");
$("p").removeClass("blue").addClass("red");
// add if absent, remove otherwise
$("div").toggleClass("main"); // test for class existence
 if ($("div").hasClass("main")) { //… }
```

jQuery wrap()

jQuery wrap() method is used to wrap specified HTML elements around each selected element. The wrap () function can accept any string or object that could be passed through the $() factory function.

Syntax:
$(selector).wrap(wrappingElement,function(index))

```
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
<script>
$(document).ready(function(){
```

```
    $("button").click(function(){
        $("p").wrap("<div></div>");
    });
});
</script>
<style>
div{background-color: pink;}
</style>
</head>
<body>
<p>Hello Guys!</p>
<p>This is javatpoint.com</p>
<button>Wrap a div element around each p element</button>
</body>
</html>
```

DOM Element Replacement

You can replace a complete DOM element with the specified HTML or DOM elements.

The **replaceWith(content)** method serves this purpose very well.

Here is the syntax for the method –

```
selector.replaceWith( content )
```

Here content is what you want to have instead of original element. This could be HTML or simple text.

Example

Following is an example which would replace division element with "<h1>JQuery is Great </h1>" –

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type = "text/javascript"
         src = "http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type = "text/javascript" language = "javascript">
         $(document).ready(function() {
            $("div").click(function () {
               $(this).replaceWith("<h1>JQuery is Great</h1>");
            });
         });
      </script>

      <style>
         #division{ margin:10px;padding:12px; border:2px solid #666; width:60px;}
      </style>
   </head>

   <body>
      <p>Click on the square below:</p>
```

```
      <span id = "result"> </span>

      <div id = "division" style = "background-color:blue;">
         This is Blue Square!!
      </div>
   </body>
</html>
```

replace() Example

```
$("p:first").replaceWith("Hello world!"); //Replace the first p element with new text
$("<h2>Hello world!</h2>").replaceAll("p"); //Replace all <p> elements with <h2> elements:

Replacing Elements while keeping the content
$("h3").each(function(){
  $(this).replaceWith("<div>" + $(this).html() + "</div>");
});
```

Removing DOM Elements

There may be a situation when you would like to remove one or more DOM elements from the document. JQuery provides two methods to handle the situation.

The **empty( )** method remove all child nodes from the set of matched elements where as the method **remove(expr)** method removes all matched elements from the DOM.

Here is the syntax for the method −

```
selector.remove( [ expr ])
or
selector.empty( )
```

You can pass optional parameter *expr* to filter the set of elements to be removed.

Example

Following is an example where elements are being removed as soon as they are clicked −

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type = "text/javascript"
         src = "http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type = "text/javascript" language = "javascript">
         $(document).ready(function() {
            $("div").click(function () {
               $(this).remove( );
            });
         });
      </script>
```

```
      <style>
         .div{ margin:10px;padding:12px; border:2px solid #666; width:60px;}
      </style>
   </head>

   <body>

      <p>Click on any square below:</p>
      <span id = "result"> </span>
      <div class = "div" style = "background-color:blue;"></div>
      <div class = "div" style = "background-color:green;"></div>
      <div class = "div" style = "background-color:red;"></div>

   </body>
</html>
```

remove() Example


```
// remove all children
$("#mainContent").empty(); //Removes the child elements only from the selected element // remove
selection
$("span.names").remove(); //Removes the selected element (and its child elements)
//Filtering elements to be removed
$("p").remove(".italic"); //removes all <p> elements with class="italic":
```

Effects

jQuery provides a trivially simple interface for doing various kind of amazing effects. jQuery methods allow us to quickly apply commonly used effects with a minimum configuration.

Showing and Hiding elements

The commands for showing and hiding elements are pretty much what we would expect – **show()** to show the elements in a wrapped set and **hide()** to hide them.

Syntax

Here is the simple syntax for **show()** method –

```
[selector].show( speed, [callback] );
```

Here is the description of all the parameters –

- **speed** – A string representing one of the three predefined speeds ("slow", "normal", or "fast") or the number of milliseconds to run the animation (e.g. 1000).
- **callback** – This optional parameter represents a function to be executed whenever the animation completes; executes once for each element animated against.

Following is the simple syntax for **hide()** method –

```
[selector].hide( speed, [callback] );
```

Here is the description of all the parameters –

- **speed** – A string representing one of the three predefined speeds ("slow", "normal", or "fast") or the number of milliseconds to run the animation (e.g. 1000).
- **callback** – This optional parameter represents a function to be executed whenever the animation completes; executes once for each element animated against.

Example

Consider the following HTML file with a small JQuery coding –

```
<html>

   <head>
      <title>The jQuery Example</title>
      <script type = "text/javascript"
         src = "http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type = "text/javascript" language = "javascript">
         $(document).ready(function() {

            $("#show").click(function () {
               $(".mydiv").show( 1000 );
            });

            $("#hide").click(function () {
               $(".mydiv").hide( 1000 );
            });

         });
      </script>

      <style>
         .mydiv{ margin:10px;padding:12px; border:2px solid #666; width:100px; height:100px;}
      </style>
   </head>

   <body>

      <div class = "mydiv">
         This is a SQUARE
      </div>

      <input id = "hide" type = "button" value = "Hide" />
      <input id = "show" type = "button" value = "Show" />

   </body>
```

```
</html>
```

Toggling the elements

jQuery provides methods to toggle the display state of elements between revealed or hidden. If the element is initially displayed, it will be hidden; if hidden, it will be shown.

Syntax

Here is the simple syntax for one of the toggle() methods −

```
[selector]..toggle([speed][, callback]);
```

Here is the description of all the parameters −

- **speed** − A string representing one of the three predefined speeds ("slow", "normal", or "fast") or the number of milliseconds to run the animation (e.g. 1000).
- **callback** − This optional parameter represents a function to be executed whenever the animation completes; executes once for each element animated against.

jQuery slideDown() Method

The jQuery slideDown() method is used to slide down an element.

Syntax:

```
$(selector).slideDown(speed,callback);
```

The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.

The optional callback parameter is a function to be executed after the sliding completes.

The following example demonstrates the slideDown() method:

Example

```
$("#flip").click(function(){
    $("#panel").slideDown();
});
```

jQuery slideUp() Method

The jQuery slideUp() method is used to slide up an element.

Syntax:

```
$(selector).slideUp(speed,callback);
```

The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.

The optional callback parameter is a function to be executed after the sliding completes.

The following example demonstrates the slideUp() method:

Example

```
$("#flip").click(function(){
    $("#panel").slideUp();
});
```

jQuery slideToggle() Method

The jQuery slideToggle() method toggles between the slideDown() and slideUp() methods.

If the elements have been slid down, slideToggle() will slide them up.

If the elements have been slid up, slideToggle() will slide them down.

```
$(selector).slideToggle(speed,callback);
```

The optional speed parameter can take the following values: "slow", "fast", milliseconds.

The optional callback parameter is a function to be executed after the sliding completes.

The following example demonstrates the slideToggle() method:

Example

```
$("#flip").click(function(){
    $("#panel").slideToggle();
});
```

jQuery fadeIn()

jQuery fadeIn() method is used to fade in the element.

Syntax:

```
$(selector).fadein();
$(selector).fadeIn(speed,callback);
$(selector).fadeIn(speed, easing, callback);
```


jQuery fadeOut()

The jQuery fadeOut() method is used to fade out the element.

Syntax:

```
$(selector).fadeOut();
$(selector).fadeOut(speed,callback);
$(selector).fadeOut(speed, easing, callback);
```

jQuery Effect fadeTo() Method

The fadeTo() method gradually changes the opacity, for selected elements, to a specified opacity (fading effect).

Syntax

```
$(selector).fadeTo(speed,opacity,easing,callback)
```

Gradually change the opacity of all <p> elements:

```
$("button").click(function(){
    $("p").fadeTo(1000, 0.4);
});
```

jQuery animate() Method

The animate() method performs a custom animation of a set of CSS properties.

This method changes an element from one state to another with CSS styles. The CSS property value is changed gradually, to create an animated effect.

Only numeric values can be animated (like "margin:30px"). String values cannot be animated (like "background-color:red"), except for the strings "show", "hide" and "toggle". These values allow hiding and showing the animated element.

Tip: Use "+=" or "-=" for relative animations.

```
$("button").click(function(){
    $("#box").animate({height: "300px"});
});
```


Syntax

```
(selector).animate({styles},speed,easing,callback)
```

Parameter	Description
styles	Required. Specifies one or more CSS properties/values to animate. Note: The property names must be camel-cased when used with the animate() method: You will need to write paddingLeft instead of padding-left, marginRight instead of margin-right, and so on. Properties that can be animated: backgroundPositionX, backgroundPositionY, borderWidth, borderBottomWidth, borderLeftWidth, borderRightWidth, borderTopWidth, borderSpacing,margin, marginBottom, marginLeft, marginRight, marginTop, outlineWidth, padding, paddingBottom, paddingLeft, paddingRight, paddingTop, height, width, maxHeight, maxWidth, minHeight, minWidth, fontSize, bottom, left, right, top, letterSpacing, wordSpacing, lineHeight, textIndent Only numeric values can be animated (like "margin:30px"). String values cannot be animated (like "background-color:red"), except for the strings "show", "hide" and "toggle". These values allow hiding and showing the animated element. Tip: Color animations are not included in the core jQuery library. If you want to animate color, you need to download the Color Animations plugin from jQuery.com

speed Optional. Specifies the speed of the animation. Default value is 400 milliseconds
Possible values:
milliseconds (like 100, 1000, 5000, etc)
"slow"
"fast"

easing Optional. Specifies the speed of the element in different points of the animation. Default value is "swing". Possible values:
"swing" - moves slower at the beginning/end, but faster in the middle
"linear" - moves in a constant speed
Tip: More easing functions are available in external plugins.

callback Optional. A function to be executed after the animation completes. To learn more about callback, please read our jQuery Callback chapter

Alternate Syntax

```
(selector).animate({styles},{options})
```

Parameter	Description
styles	Required. Specifies one or more CSS properties/values to animate (See possible values above)
options	Optional. Specifies additional options for the animation. Possible values: **duration** - sets the speed of the animation **easing** - specifies the easing function to use **complete** - specifies a function to be executed after the animation completes **step** - specifies a function to be executed for each step in the animation **progress** - specifies a function to be executed after each step in the animation **queue** - a Boolean value specifying whether or not to place the animation in the effects queue **specialEasing** - a map of one or more CSS properties from the *styles* parameter, and their corresponding easing functions **start** - specifies a function to be executed when the animation begins **done** - specifies a function to be executed when the animation ends **fail** - specifies a function to be executed if the animation fails to complete **always** - specifies a function to be executed if the animation stops without completing

Using animate() with a callback function

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.2/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
```

```
        var div = $("div");
        startAnimation();
        function startAnimation(){
            div.animate({height: 300}, "slow");
            div.animate({width: 300}, "slow");
            div.css("background-color", "blue");
            div.animate({height: 100}, "slow");
            div.animate({width: 100}, "slow", startAnimation);
        }
    });
});
</script>
</head>
<body>

<button>Start Animation</button>

<div style="width:50px;height:50px;position:absolute;left:10px;top:50px;background-color:red;"></div>

</body>
</html>
```

Alternate Syntax Example


```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.2/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("#btn").click(function(){
    $("#box").animate({
      height: "300px",
      width: "300px"
    }, {
      duration: 5000,
      easing: "linear",
      complete: function(){
        $(this).after("<p>Animation is complete!</p>");
      }
    });
  });
});
</script>
</head>
<body>

<button id="btn">Animate height & width</button>

<div id="box" style="background:#98bf21;height:100px;width:100px;margin:6px;"></div>

</body>
</html>
```

ProgressBar Example

```
<!DOCTYPE html>
<html>
<head>
<script src="jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("#btn").click(function(){
    $("#box").animate({
      width: "400px"
    }, {
      duration: 5000,
      easing: "linear",
      step: function(x) {
        $("#demo").text(Math.round(x * 100 / 400)  + "%");
      }
    });
  });
});
</script>
</head>
<body>

<button id="btn">Start Progress Bar</button>

<div style="border:1px solid green;margin:10px;width:400px;">
  <div id="box" style="background:#98bf21;height:50px;width:1px;border:1px solid green;"></div>
</div>

<p id="demo"></p>

</body>
</html>
```

JQuery Effect Methods

You have seen basic concept of jQuery Effects. Following table lists down all the important methods to create different kind of effects –

S.N.	Methods & Description
1	**animate(params, [duration, easing, callback])** A function for making custom animations.
2	**fadeIn(speed, [callback])** Fade in all matched elements by adjusting their opacity and firing an optional callback after completion.
3	**fadeOut(speed, [callback])** Fade out all matched elements by adjusting their opacity to 0, then setting display to "none" and firing an optional callback after completion.
4	**fadeTo(speed, opacity, callback)** Fade the opacity of all matched elements to a specified opacity and firing an optional callback after completion.
5	**hide( )** Hides each of the set of matched elements if they are shown.
6	**hide(speed, [callback])** Hide all matched elements using a graceful animation and firing an optional callback after completion.
7	**show( )** Displays each of the set of matched elements if they are hidden.
8	**show(speed, [callback])** Show all matched elements using a graceful animation and firing an optional callback after completion.
9	**slideDown(speed, [callback])** Reveal all matched elements by adjusting their height and firing an optional callback after completion.
10	**slideToggle(speed, [callback])** Toggle the visibility of all matched elements by adjusting their height and firing an optional callback after completion.
11	**slideUp(speed, [callback])** Hide all matched elements by adjusting their height and firing an optional callback after completion.
12	**stop([clearQueue, gotoEnd])** Stops all the currently running animations on all the specified elements.
13	**toggle( )** Toggle displaying each of the set of matched elements.
14	**toggle(speed, [callback])** Toggle displaying each of the set of matched elements using a graceful animation and firing an optional callback after completion.

15	**toggle(switch)** Toggle displaying each of the set of matched elements based upon the switch (true shows all elements, false hides all elements).
16	**jQuery.fx.off** Globally disable all animations.