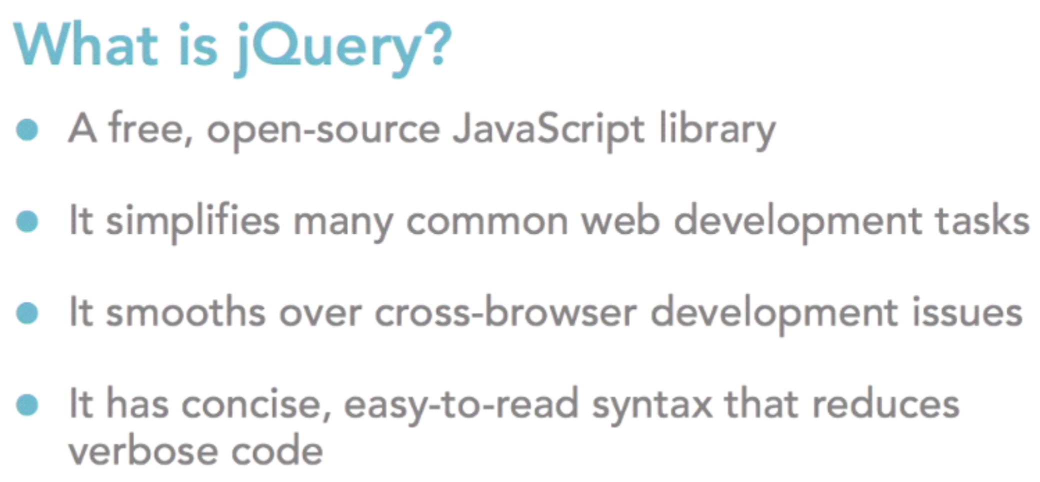
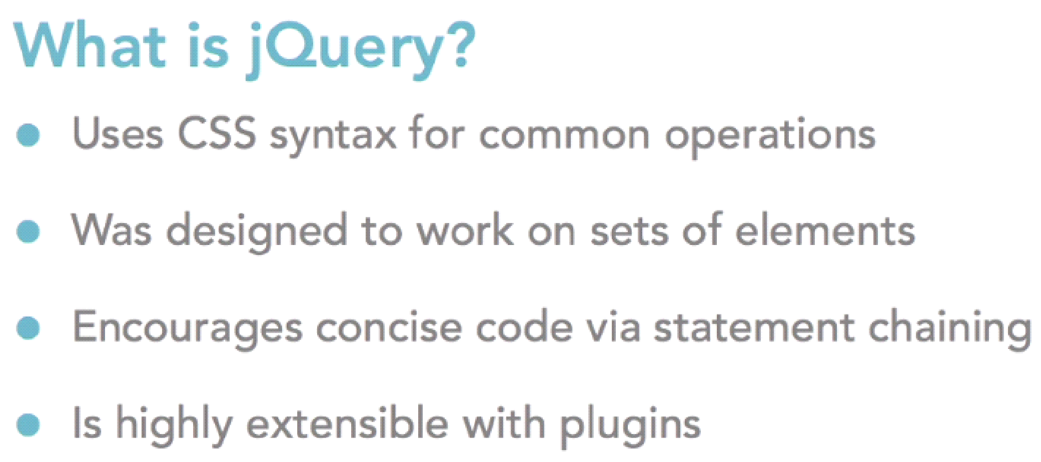
What is Jquery

Thursday, March 30, 2017 3:31 PM



What is jQuery?

- A free, open-source JavaScript library
- It simplifies many common web development tasks
- It smooths over cross-browser development issues
- It has concise, easy-to-read syntax that reduces verbose code



What is jQuery?

- Uses CSS syntax for common operations
- Was designed to work on sets of elements
- Encourages concise code via statement chaining
- Is highly extensible with plugins

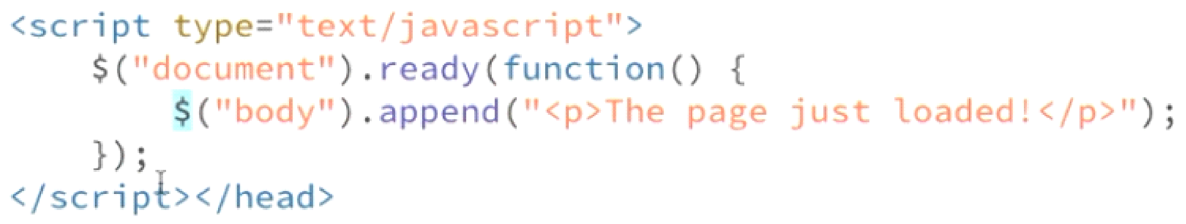
NOTE -

- One of the common pattern we see in Jquery is that, nearly all property functions have two versions
  1- property_name(value) - set the value of property
  2- Property_name() - return the value of property.

First Jquery Enabled Page

Thursday, March 30, 2017 3:48 PM

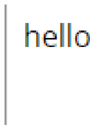
- Dollar sign is used for calling Jquery library , we can also user "jquery" in place of "$" sign



```
<script type="text/javascript">
    $("document").ready(function() {
        $("body").append("<p>The page just loaded!</p>");
    });
</script></head>
```

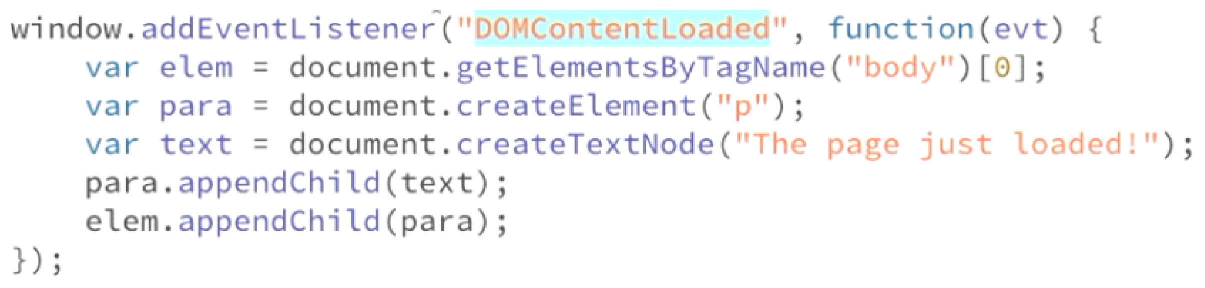
- Here we can see, ready() is event which is being activated on "document" , and a callback function is being passed for handling event.
  ready() event is called as soon as DOM content is loaded(different from onload of document, which fired when everything is Loaded including images.
- In handler, append() function is used which append something to anything,in this case it append to <body>

OUTPUT-



```
hello
```

Same code without Jquery



```
window.addEventListener("DOMContentLoaded", function(evt) {
    var elem = document.getElementsByTagName("body")[0];
    var para = document.createElement("p");
    var text = document.createTextNode("The page just loaded!");
    para.appendChild(text);
    elem.appendChild(para);
});
```

Selector and Filters Intro

Thursday, March 30, 2017 3:56 PM

Introduction to jQuery Selectors and Filters

Selectors and Filters provide a way of finding and extracting information from Web pages.

Selectors

Selectors are used to select parts of the Web page using a common CSS-style syntax.

For example, `$("p")` will select all of the paragraph tags in a document and return them as a list that can be further operated upon.

Filters

Filters are used to further refine the results returned from selectors.

For example, `$("p:first")` will select the first paragraph in the returned set from `$("p")`.

- This css function will set css of the element.
  so, this example will set css so every paragraph.

  ```
      $("p").css("border", "3px solid red");
  ```

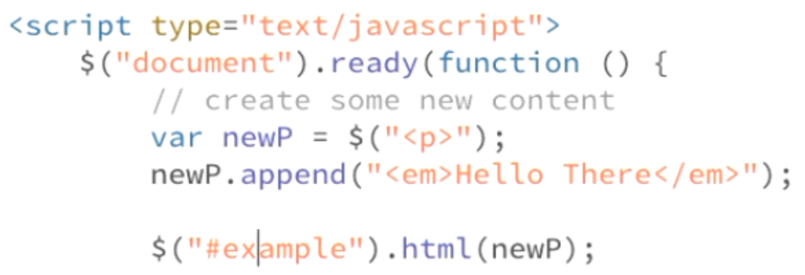
- Following example will set css to every 'h2' except having class 'selectors'

```
$("h2:not(.selectors)").css("border", "3px solid red");
```

Creating and changing page content Intro

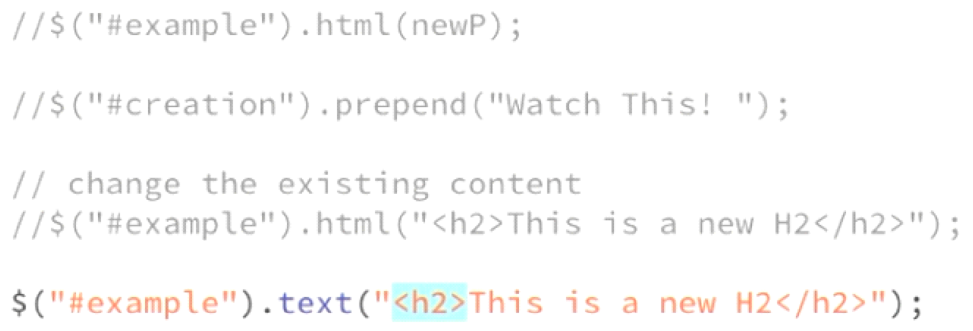
Thursday, March 30, 2017 5:45 PM

- Html() function is same as innerHTML() which changes the html content.



```
<script type="text/javascript">
    $("document").ready(function () {
        // create some new content
        var newP = $("<p>");
        newP.append("<em>Hello There</em>");

        $("#example").html(newP);
```

- We can use text() method to add something text same as innerText().

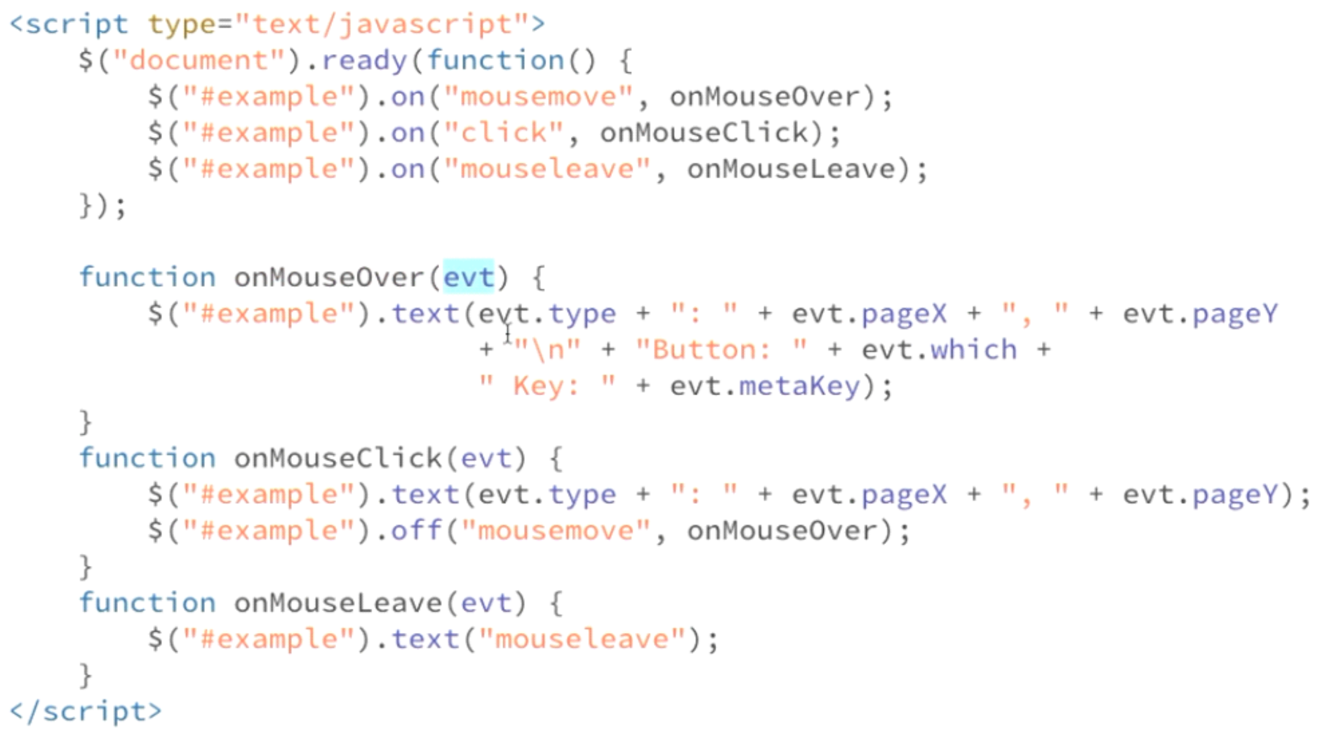


```
//$("#example").html(newP);

//$("#creation").prepend("Watch This! ");

// change the existing content
//$("#example").html("<h2>This is a new H2</h2>");

$("#example").text("<h2>This is a new H2</h2>");
```

Event Handling Intro

Thursday, March 30, 2017 6:27 PM

- Events in Jquery are very simple, we can bind an event using **on(<event_type>,callback_fn)** function , and similarly we can unbind an event using off(**<event_type>,callback_fn)** event.



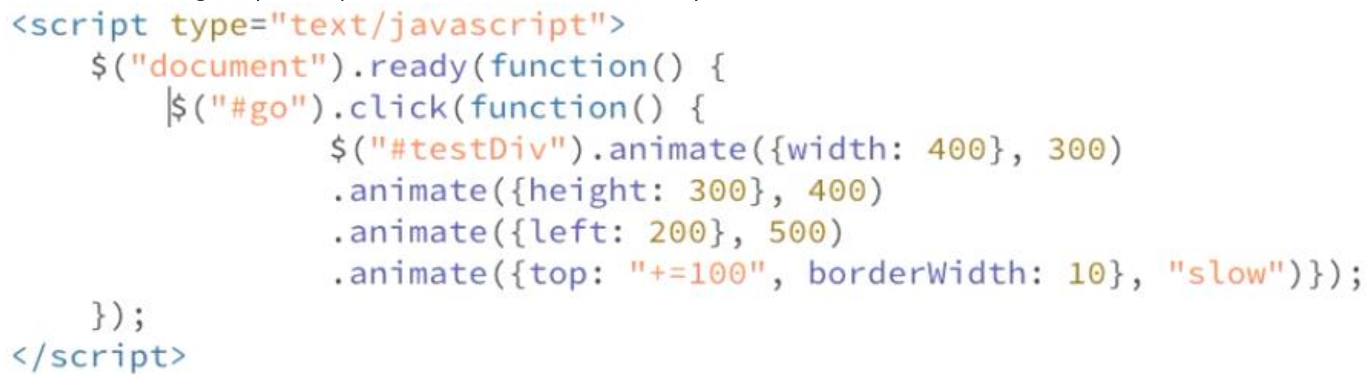
```
<script type="text/javascript">
    $("document").ready(function() {
        $("#example").on("mousemove", onMouseOver);
        $("#example").on("click", onMouseClick);
        $("#example").on("mouseleave", onMouseLeave);
    });

    function onMouseOver(evt) {
        $("#example").text(evt.type + ": " + evt.pageX + ", " + evt.pageY
                               + "\n" + "Button: " + evt.which +
                               " Key: " + evt.metaKey);
    }
    function onMouseClick(evt) {
        $("#example").text(evt.type + ": " + evt.pageX + ", " + evt.pageY);
        $("#example").off("mousemove", onMouseOver);
    }
    function onMouseLeave(evt) {
        $("#example").text("mouseleave");
    }
</script>
```

Animations intro

Thursday, March 30, 2017 6:27 PM

- Here we are attaching a click event, and in handler we are doing animation using **animate({parameters},time_in_ms)**

- So we are animating width to 400 in time=300 second an so on,
  in last we are adding 100px to top and time is "slow" which nearly to 600ms.



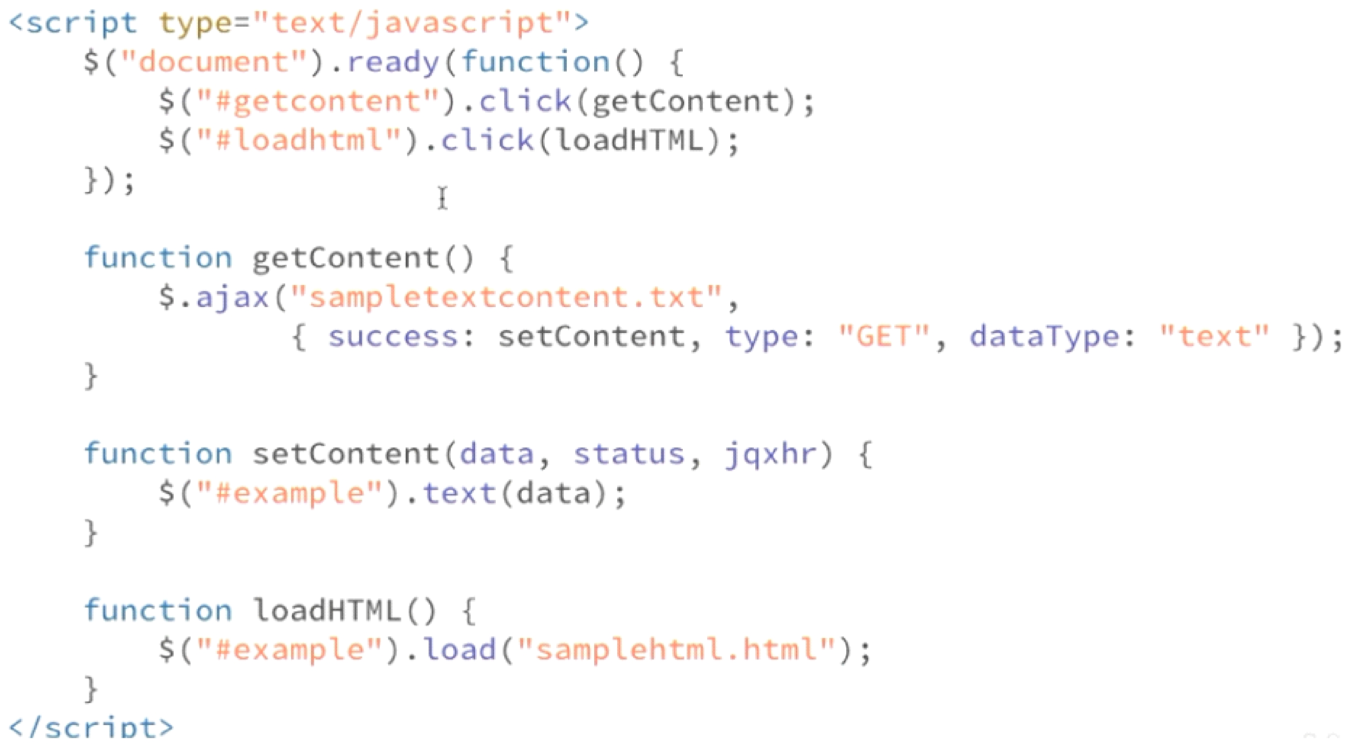
```
<script type="text/javascript">
    $("document").ready(function() {
        $("#go").click(function() {
                    $("#testDiv").animate({width: 400}, 300)
                    .animate({height: 300}, 400)
                    .animate({left: 200}, 500)
                    .animate({top: "+=100", borderWidth: 10}, "slow")});
    });
</script>
```

Ajax intro

Thursday, March 30, 2017 6:36 PM

- Global function(no selectro needed) **ajax()** can be used here,
  which passed everything to function **setContent()** if success happens.
- SetContent function then print out the data.

- Now the above scenario(loading html from server) is so common, that
  jquery provided as with function **load()** which loads from server as html and puts inside
  selected element.

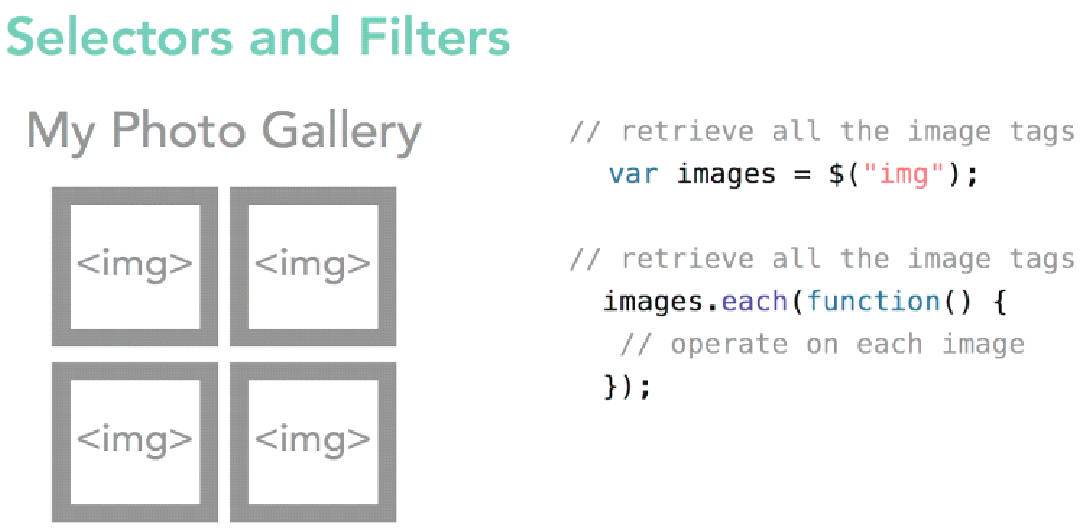


```
<script type="text/javascript">
    $("document").ready(function() {
        $("#getcontent").click(getContent);
        $("#loadhtml").click(loadHTML);
    });

    function getContent() {
        $.ajax("sampletextcontent.txt",
               { success: setContent, type: "GET", dataType: "text" });
    }

    function setContent(data, status, jqxhr) {
        $("#example").text(data);
    }

    function loadHTML() {
        $("#example").load("samplehtml.html");
    }
</script>
```

Detail Selectros and Filters

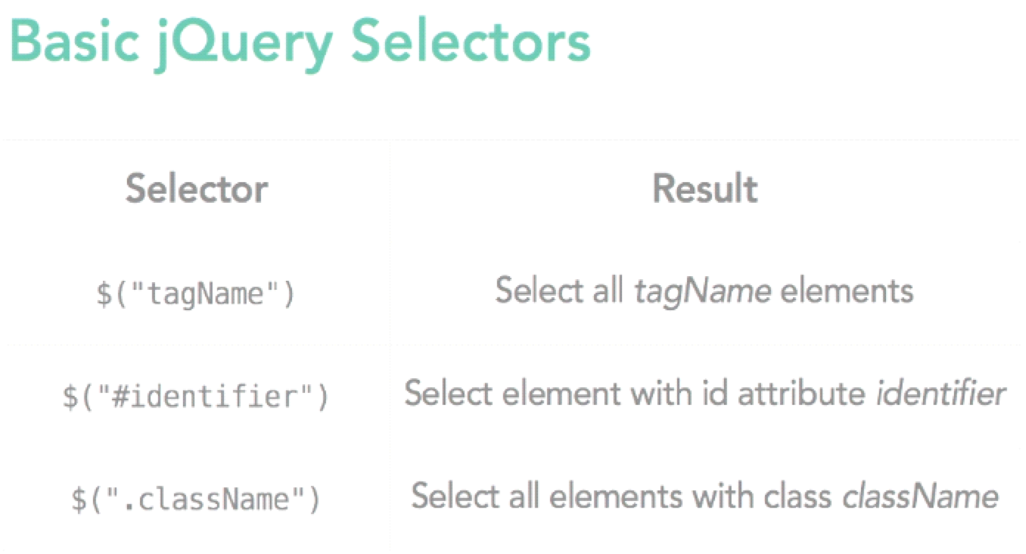
Friday, March 31, 2017 12:30 PM

- Simple CSS like syntax is used which is same as using with **document.querySelectorAll()**



Selectors and Filters

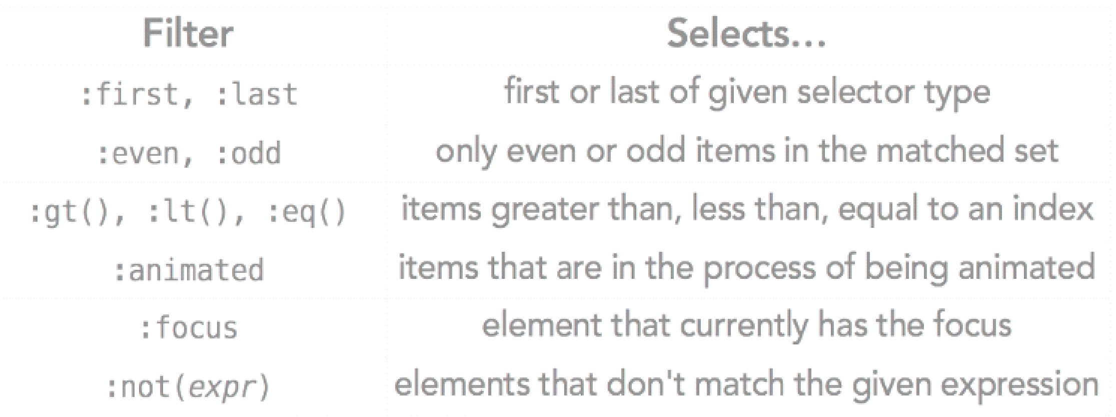
```
// retrieve all the image tags
  var images = $("img");

// retrieve all the image tags
  images.each(function() {
   // operate on each image
  });
```



Basic jQuery Selectors

Selector	Result
`$("tagName")`	Select all *tagName* elements
`$("#identifier")`	Select element with id attribute *identifier*
`$(".className")`	Select all elements with class *className*

Basic jQuery Filters

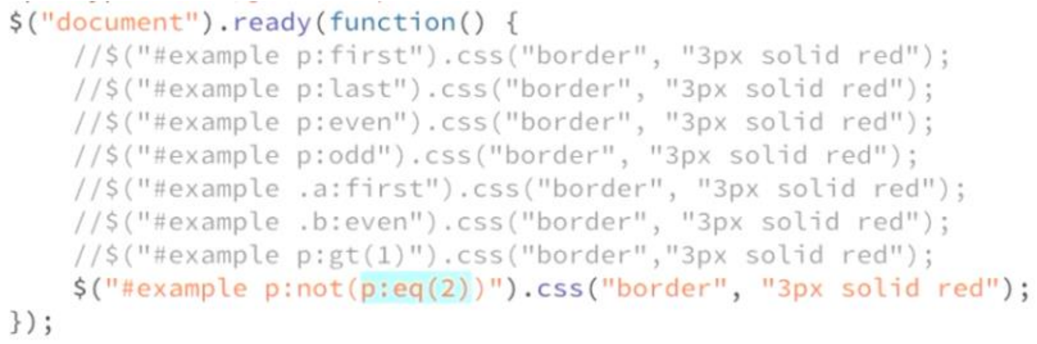


Filter	Selects…
`:first, :last`	first or last of given selector type
`:even, :odd`	only even or odd items in the matched set
`:gt(), :lt(), :eq()`	items greater than, less than, equal to an index
`:animated`	items that are in the process of being animated
`:focus`	element that currently has the focus
`:not(expr)`	elements that don't match the given expression

- Remember index starts with 0.

Example for :not(expr)

- In this example we are selecting the paragraphs inside id 'example' which are not equal to 2 index)



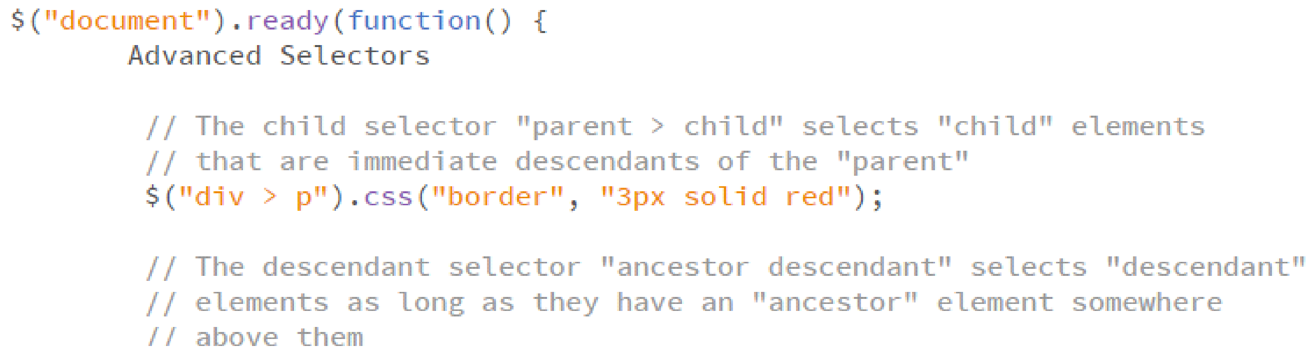
```
$("document").ready(function() {
    //$("#example p:first").css("border", "3px solid red");
    //$("#example p:last").css("border", "3px solid red");
    //$("#example p:even").css("border", "3px solid red");
    //$("#example p:odd").css("border", "3px solid red");
    //$("#example .a:first").css("border", "3px solid red");
    //$("#example .b:even").css("border", "3px solid red");
    //$("#example p:gt(1)").css("border","3px solid red");
    $("#example p:not(p:eq(2))").css("border", "3px solid red");
});
```

jQuery Hierarchy Selectors

To select elements based on their position in the document tree, use the Hierarchy Selectors.

The Hierarchy Selectors work by examining the position of target elements relative to other elements:

- `$("parent > child")`: selects "child" elements that are immediate descendants of the "parent"
- `$("ancestor descendant")`: selects "descendant" elements as long as they have an "ancestor" element somewhere above them
- `$("prev + next")`: selects the "next" element if it is immediately preceded by a "prev" element
- `$("prev ~ siblings")`: selects all "siblings" elements that come after a "prev" element



```
$("document").ready(function() {
        Advanced Selectors

         // The child selector "parent > child" selects "child" elements
         // that are immediate descendants of the "parent"
         $("div > p").css("border", "3px solid red");

         // The descendant selector "ancestor descendant" selects "descendant"
         // elements as long as they have an "ancestor" element somewhere
         // above them
```

```
$("document").ready(function() {
       Advanced Selectors

        // The child selector "parent > child" selects "child" elements
        // that are immediate descendants of the "parent"
        $("div > p").css("border", "3px solid red");

        // The descendant selector "ancestor descendant" selects "descendant"
        // elements as long as they have an "ancestor" element somewhere
        // above them
        $("div p.a").css("border", "3px solid red");

        // The next adjacent selector "prev + next" selects the "next"
        // element if it is immediately preceded by a "prev" element
        $("ul + div").css("border", "3px solid red");

        // Next sibling selector "prev ~ siblings" selects all "siblings"
        // elements that come after a "prev" element
        $("#para1 ~ p").css("border", "3px solid red");

});
```

Using Child and Content Filters

These selectors are used to retrieve content that matches certain conditions, such as whether they contain certain content, whether they are visible/hidden, or whether they are in a certain position within their parent.

- `:contains('text')`: select elements that contain specific text
- `:parent`: select elements have at least one child node (element or text)
- `:has(selector)`: select elements that contain at least one element that matches the selector
- `:first-child`: select elements that are the first child of their parents
- `:last-of-type`: select elements that are the last of their type among siblings
- `:nth-child()`: select elements that are the nth child of their parent

```
Advanced Filters

    $("p[class]").css("border", "3px solid red");
    $("p[id=para1]").css("border", "3px solid red");
    $("p[id^=para]").css("border", "3px solid red");
    $("p[id^=para][lang*=en-]").css("border", "3px solid red");


    $("p:contains('3')").css("border", "3px solid red");
    $("p:parent").css("border", "3px solid red");
    $("div:has(p[class=a])").css("border", "3px solid red");

    $("div p:first-child").css("border", "3px solid red");
    $("div p:last-of-type").css("border", "3px solid red");
    $("div p:nth-child(3)").css("border", "3px solid red");
    $("div p:nth-child(2n)").css("border", "3px solid red");
```

FIRST SET (using attribute filters of CSS)

- In first, "select all <p> having class attribute(no value required)
- In Second, we "select all <p> having id with value "para1"
- In Third, ^= means, starts with, so id should start with "para" so it will match id "para1" "para2" etc.
  s
  o id should start with "para" and lang should contain "en-" .
- In last example we are looking for two value. *= means contains

SECOND SET (Using Jquery filters)

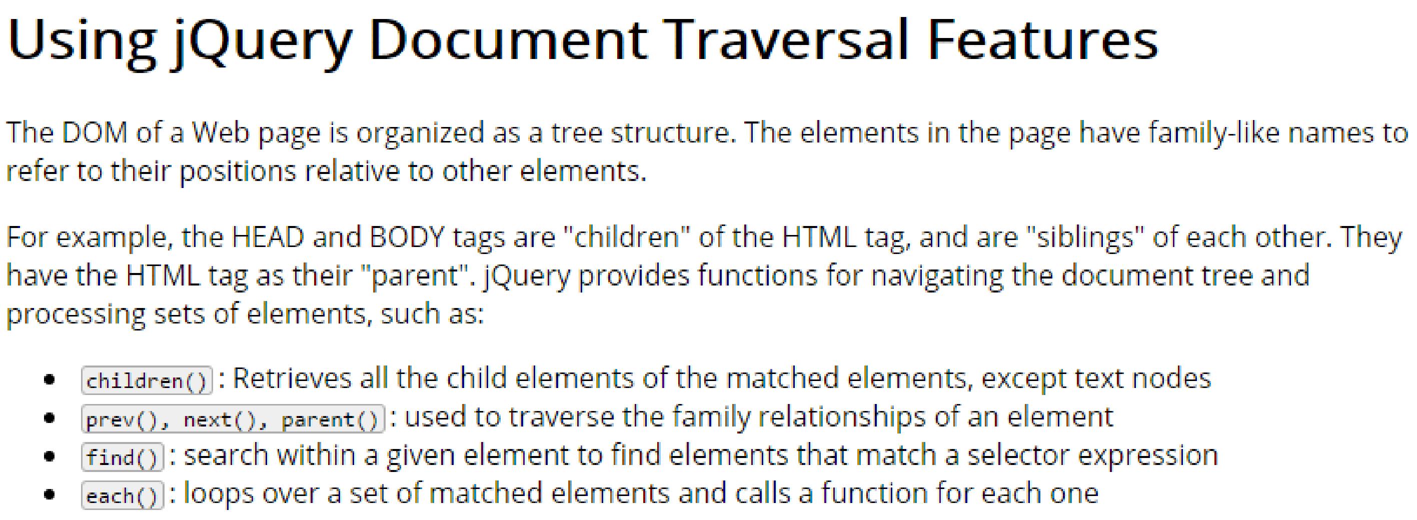
- First has contains('3'), so it will select the paragraph which has '3' in its content.
  so this will be selected as it has '3' in the content.
  ```
  <p class="b">This is paragraph 3</p>
  ```
- Second, will select all paragraphs which are parents(has at least one child)
- Third, we are looking for a <div> which has 'following expression'.

THIRD SET (Using Jquery child filter)

- First, select <p> which are inside <div> and are first-child
- Second, select<p> inside <div> which are last of its type, (no like that property has after it)
  Similarly, we have first-of-type.
- Third, nth child index, here we want to select <p> inside <div> which are 3rd child of <div>
  1 based child numbering.
- Fourth, 2n means every multiple of 2 is selected, so we can pass an expression inside

Traversing documents

Friday, March 31, 2017 2:15 PM



Using jQuery Document Traversal Features

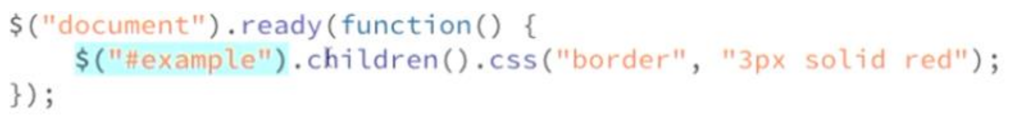
The DOM of a Web page is organized as a tree structure. The elements in the page have family-like names to refer to their positions relative to other elements.

For example, the HEAD and BODY tags are "children" of the HTML tag, and are "siblings" of each other. They have the HTML tag as their "parent". jQuery provides functions for navigating the document tree and processing sets of elements, such as:

- `children()`: Retrieves all the child elements of the matched elements, except text nodes
- `prev(), next(), parent()`: used to traverse the family relationships of an element
- `find()`: search within a given element to find elements that match a selector expression
- `each()`: loops over a set of matched elements and calls a function for each one

children()

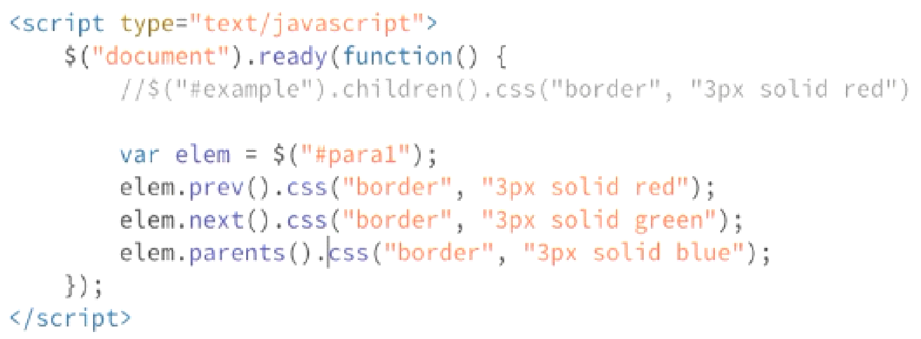
- This example select #example and then traverse to all it's children and apply css() to them.



```
$("document").ready(function() {
    $("#example").children().css("border", "3px solid red");
});
```

Prev(), next() and parents()

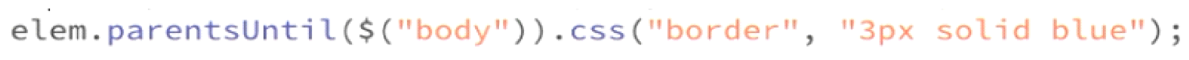
- This example use prev(),next() and parents() functions to style out content.



```
<script type="text/javascript">
    $("document").ready(function() {
        //$("#example").children().css("border", "3px solid red")

        var elem = $("#para1");
        elem.prev().css("border", "3px solid red");
        elem.next().css("border", "3px solid green");
        elem.parents().css("border", "3px solid blue");
    });
</script>
```

parentUntil()

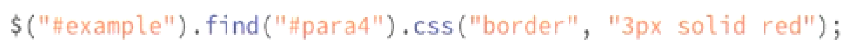
- This function will limit going upward, stop at body(also include <body>).



```
elem.parentsUntil($("body")).css("border", "3px solid blue");
```

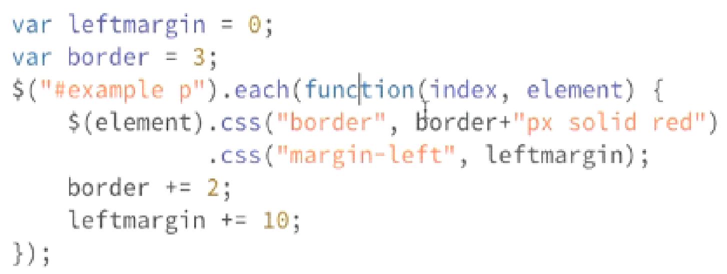
Find()

- Find function start at given point(here at #example) and try to find (#para4) inside the subtree.



```
$("#example").find("#para4").css("border", "3px solid red");
```

- Each() function can be used to loop over anything selected in Jquery.

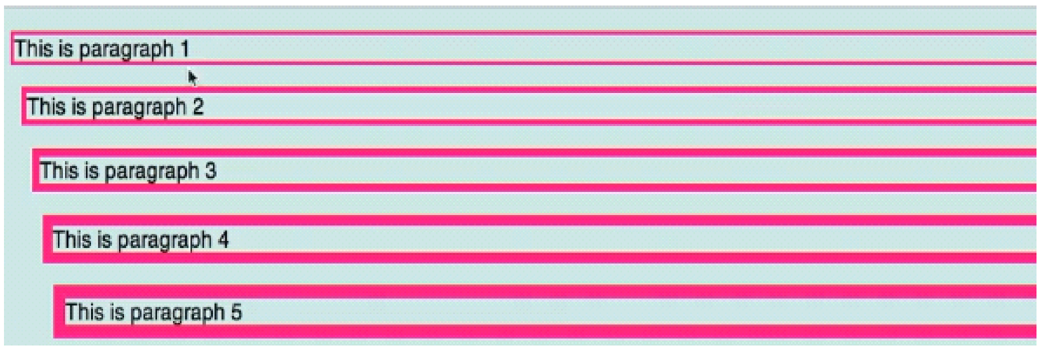


```
var leftmargin = 0;
var border = 3;
$("#example p").each(function(index, element) {
    $(element).css("border", border+"px solid red")
              .css("margin-left", leftmargin);
    border += 2;
    leftmargin += 10;
});
```

the following code inside the callback will be run for each <p> which is selected,

- arguments in callback
    - Index - index of element(0 based), tells which iteration is running
    - Selected element as dom element object(not Jquery Object), so we have to use $() notation, to convert to jquery object, so that we can apply Jquery function.
- Here we are increasing border by 2 and leftmargin by 10 each time.

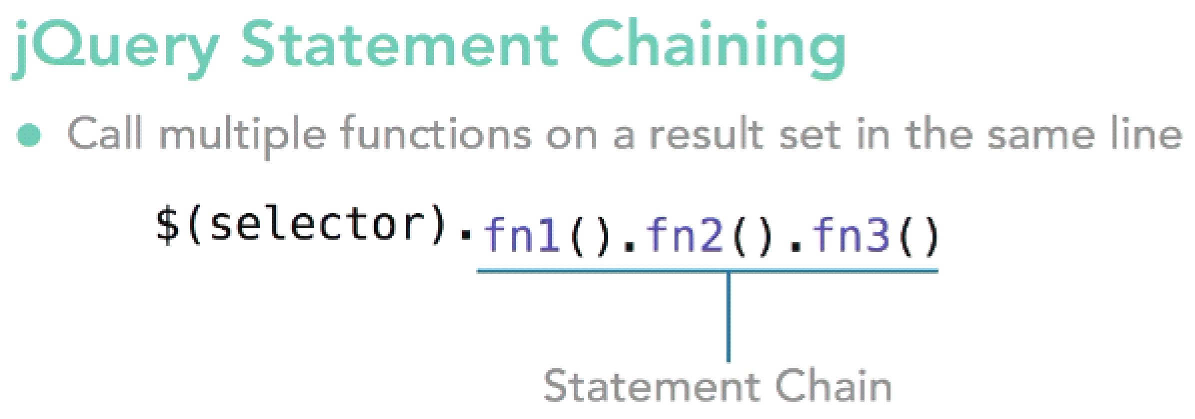
OUTPUT-



```
This is paragraph 1
This is paragraph 2
This is paragraph 3
This is paragraph 4
This is paragraph 5
```

Jquery Statement chaining

Friday, March 31, 2017 2:15 PM



jQuery Statement Chaining

- Call multiple functions on a result set in the same line

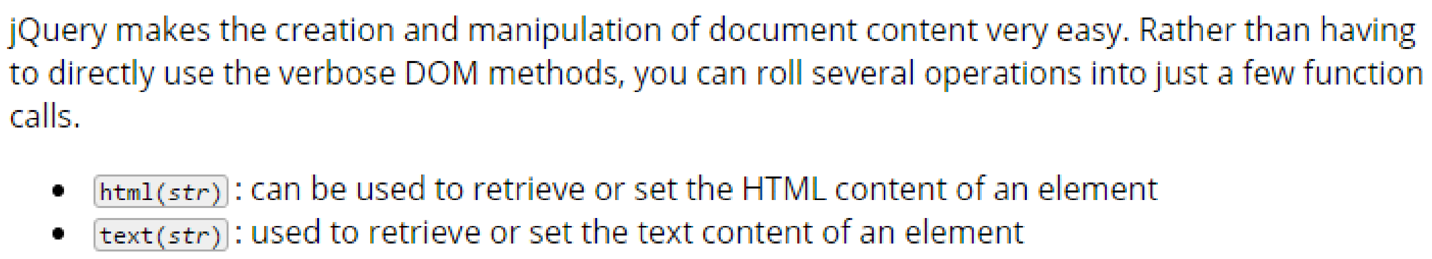
```
$(selector).fn1().fn2().fn3()
```

Statement Chain

- Statement chaining allows us to perform various operation which works from left to right.

Creating content

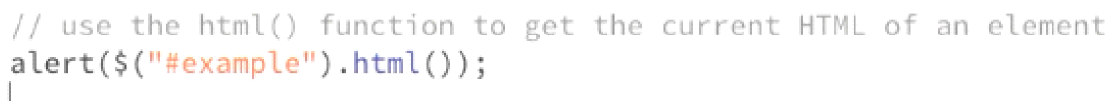
Friday, March 31, 2017 4:41 PM


jQuery makes the creation and manipulation of document content very easy. Rather than having to directly use the verbose DOM methods, you can roll several operations into just a few function calls.

- `html(str)` : can be used to retrieve or set the HTML content of an element
- `text(str)` : used to retrieve or set the text content of an element

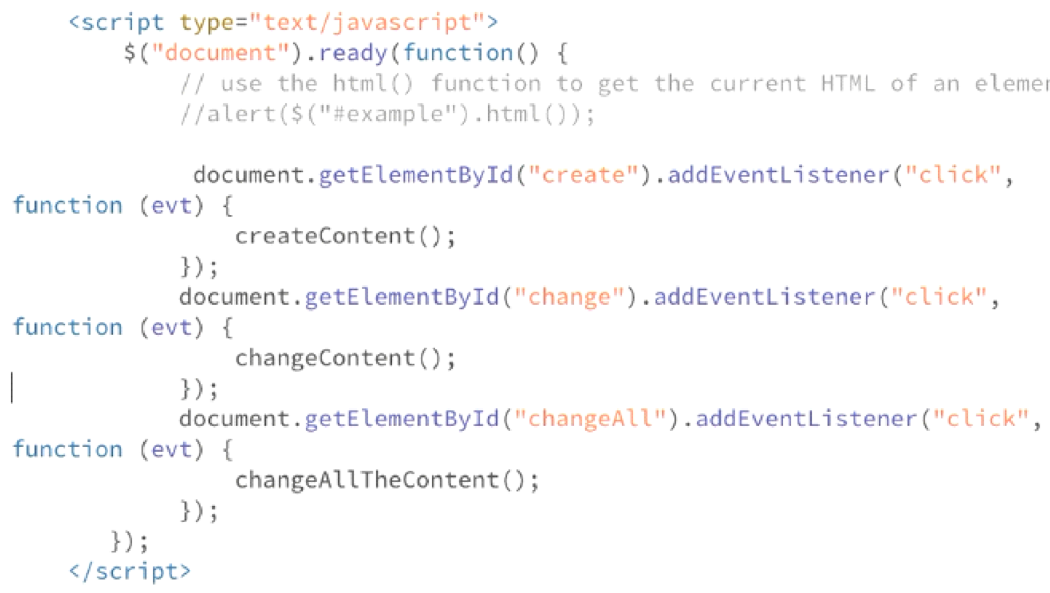
- One of the best feature of Jquery is that It takes of all cross browser supports.
- In jquery html() function does the work of innerHTML property in Dom,
  so we can access , the value as well as manipulate it.

- The below code will output all html content of #example.



```
// use the html() function to get the current HTML of an element
alert($("#example").html());
```

- Basic structure of the script is like this, now we create functions to illustrate different funcotinalities.
  - createContent() - creating content
  - ChangeContent()
  - changeAllTheContent()



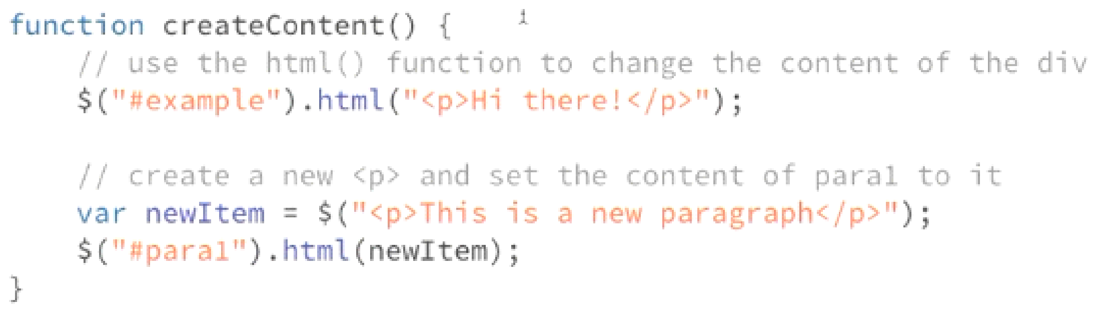
```
<script type="text/javascript">
    $("document").ready(function() {
        // use the html() function to get the current HTML of an element
        //alert($("#example").html());

         document.getElementById("create").addEventListener("click",
function (evt) {
            createContent();
        });
        document.getElementById("change").addEventListener("click",
function (evt) {
            changeContent();
        });
        document.getElementById("changeAll").addEventListener("click",
function (evt) {
            changeAllTheContent();
        });
    });
</script>
```

NOTE

- html() - innerHTML ,return as html string , and create element after parsing string as html
  Text() -innerText , return text , and create text node after parsing as text.
- Html() fuction parse the string passed to see if there is any html code inside it, text() doesn't.

Creating html content

- We can create any content by passing properly formated html element inside html() function,
  If passed inside text() it will add show thing as textnode as it do not parse as html string(parse as simple string)

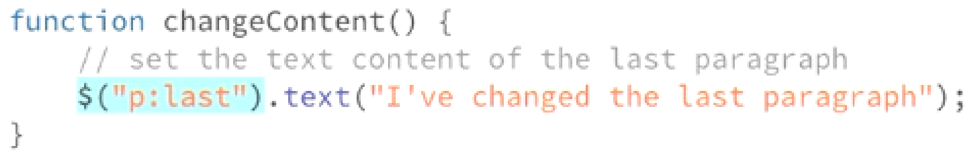


```
function createContent() {
    // use the html() function to change the content of the div
    $("#example").html("<p>Hi there!</p>");

    // create a new <p> and set the content of para1 to it
    var newItem = $("<p>This is a new paragraph</p>");
    $("#para1").html(newItem);
}
```

- In this case the content of #example will be replced by the content created.
- We can also pass html content as jquery object variable inside html() function.

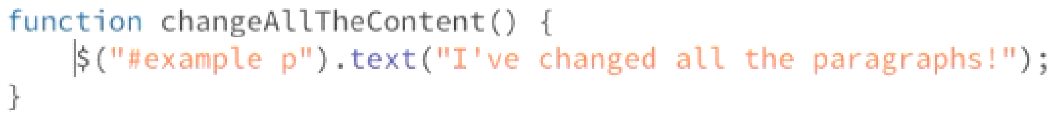
Changing content element

- The content can be change busing text()/html() function.



```
function changeContent() {
    // set the text content of the last paragraph
    $("p:last").text("I've changed the last paragraph");
}
```

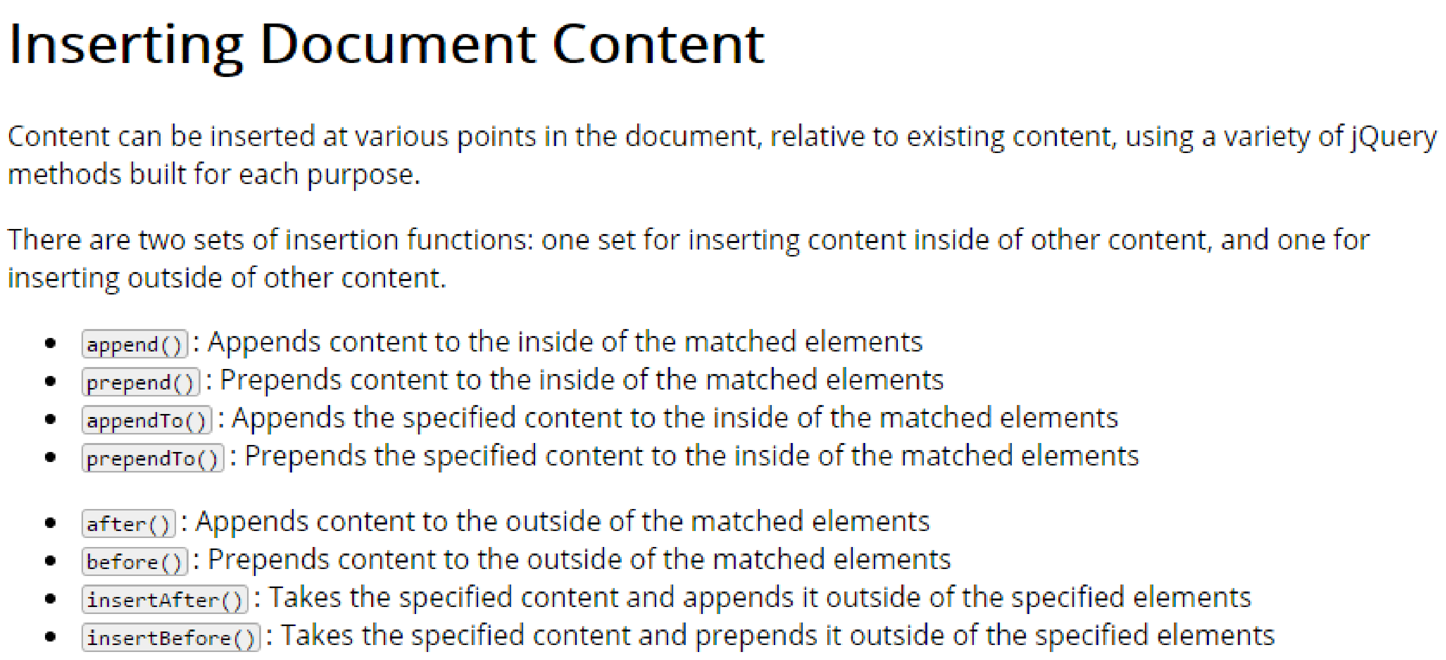
- Here we have selected last <p> and changed its content to given.



```
function changeAllTheContent() {
    $("#example p").text("I've changed all the paragraphs!");
}
```

Insert content

Friday, March 31, 2017 5:15 PM



Inserting Document Content

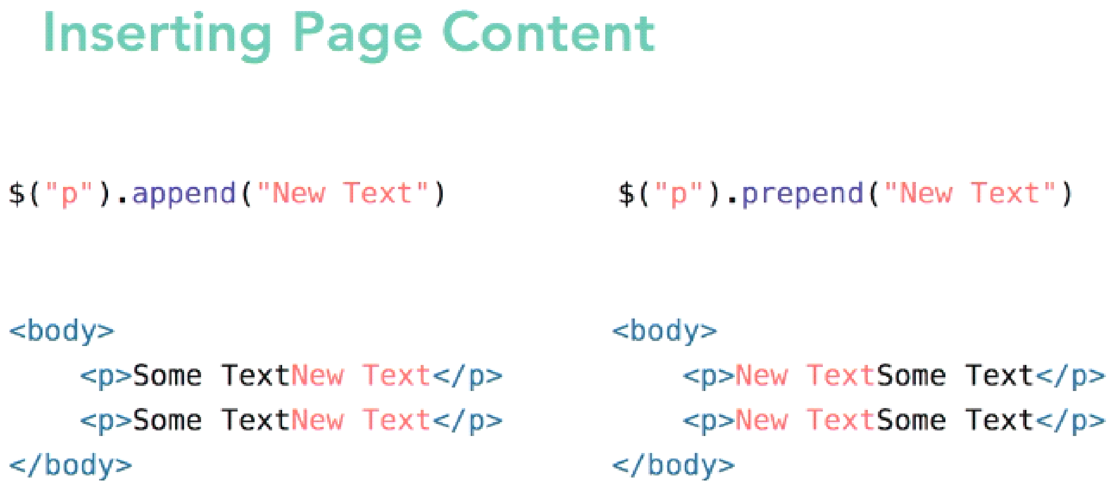
Content can be inserted at various points in the document, relative to existing content, using a variety of jQuery methods built for each purpose.

There are two sets of insertion functions: one set for inserting content inside of other content, and one for inserting outside of other content.

- `append()`: Appends content to the inside of the matched elements
- `prepend()`: Prepends content to the inside of the matched elements
- `appendTo()`: Appends the specified content to the inside of the matched elements
- `prependTo()`: Prepends the specified content to the inside of the matched elements

- `after()`: Appends content to the outside of the matched elements
- `before()`: Prepends content to the outside of the matched elements
- `insertAfter()`: Takes the specified content and appends it outside of the specified elements
- `insertBefore()`: Takes the specified content and prepends it outside of the specified elements

Note - insertAfter() is same as after() and insertBefore() is same as before().



Inserting Page Content

```
$("p").append("New Text")

<body>
    <p>Some TextNew Text</p>
    <p>Some TextNew Text</p>
</body>
```

```
$("p").prepend("New Text")

<body>
    <p>New TextSome Text</p>
    <p>New TextSome Text</p>
</body>
```

- Append() and prepend() function can append anything to end of selected elements, here we are appending text but we can append html element by passing html string.

appendTo()

- We can use appendTo() function to append some selected element to somewhere.
  Here we are appending 'new Text' to p:first.
- Similarly we have **preprendTo()**

```
$("New Text").appendTo("p:first")
```

```
<body>
    <p>Some TextNew Text</p>
    <p>Some Text</p>
</body>
```

Before()

- Creating a new element before selected element.(in append we are changing the already existing)

```
$("p").before("New Text")
```

```
<body>
    New Text<p>Some Text</p>
    New Text<p>Some Text</p>
</body>
```

After()

```
$("p").after("New Text")
```

```
<body>
    <p>Some Text</p>New Text
    <p>Some Text</p>New Text
</body>
```

Altering Content

Friday, March 31, 2017 6:05 PM

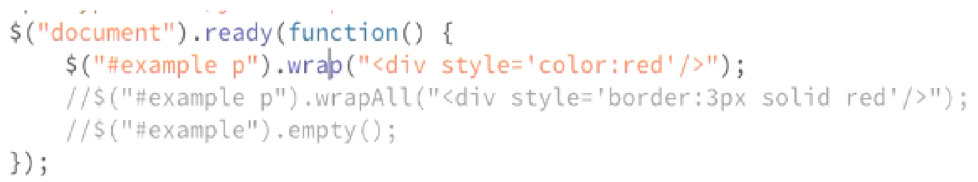
Altering Document Content

In addition to creating and inserting content, the ability to alter the current content of a page is a key scenario in Web development. jQuery has functions tailored to specific situations where content needs to be altered:

- `wrap()`: wrap the matched elements with the specified content
- `wrapAll()`: wrap content around the matched elements as a group
- `unWrap()`: remove the parents from the matched elements
- `empty()`: remove all the child elements from the matched elements
- `remove()`: removes elements from the page, including any embedded data and event handlers
- `detach()`: removes elements from the page, but maintains embedded data and event handlers
- `replaceAll()`: replaces the matched elements with the specified content
- `replaceWith()`: replaces matched elements with content or the results of a callback function

Wrap()

- We can use wrap() function to wrap selected elements around some thing individually.

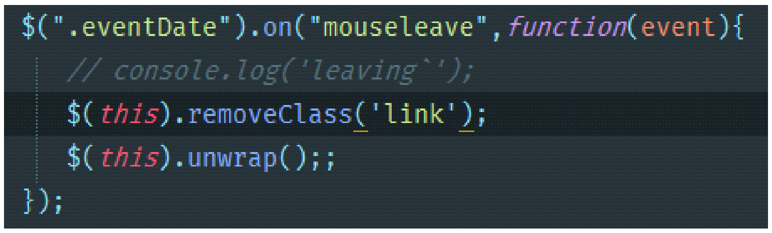


```
$("document").ready(function() {
    $("#example p").wrap("<div style='color:red'/>");
    //$("#example p").wrapAll("<div style='border:3px solid red'/>");
    //$("#example").empty();
});
```

- So here wrap() function is wrapping <p> inside #example with <div> which will apply styling on them.

unWrap()

- Similarly, **unwrap()** will remove parent and keep the element at its place, not it become child of its previous grandparent.



```
$(".eventDate").on("mouseleave",function(event){
    // console.log('leaving`');
    $(this).removeClass('link');
    $(this).unwrap();;
});
```

Wrapall()

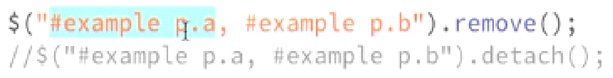
- Wrap() function wrap each selected inside given html individually,
  wrapAll() function wrap all together inside that html.

Empty()

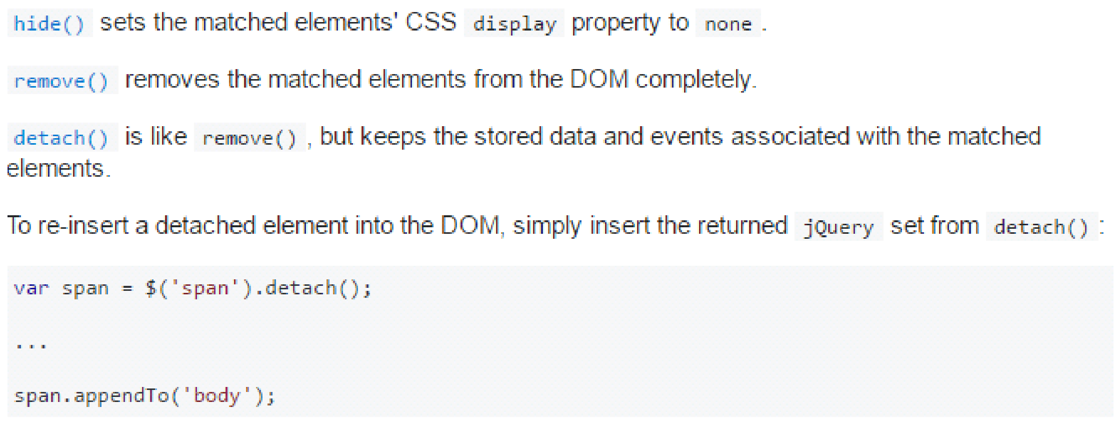
- Empty function empties out all selected elements by deleting all content inside them.
  It means it removes all childrens of an element.

Remove() and detach() and hide()

- Remove and detach are same and both removes an element.
  the only difference is that remove() also removes event handlersccc
- Hide() will hide the element using attribute hidden.



```
$("#example p.a, #example p.b").remove();
//$("#example p.a, #example p.b").detach();
```



`hide()` sets the matched elements' CSS `display` property to `none`.

`remove()` removes the matched elements from the DOM completely.

`detach()` is like `remove()`, but keeps the stored data and events associated with the matched elements.

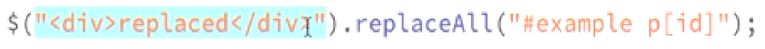
To re-insert a detached element into the DOM, simply insert the returned `jQuery` set from `detach()`:

```
var span = $('span').detach();

...

span.appendTo('body');
```

- So we should use detach() if we wanted to reinsert it again, we can use that returned and insert anywhere(here we done at end of <body>).

replaceAll() and replaceWith()

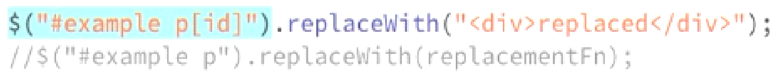
- here we are creating a <div> jquery object and then replace with every thing which match.



```
$("<div>replaced</div>").replaceAll("#example p[id]");
```

take this and replace all these.

- replaceWith() is same as replaceAll() only the positioning of both sides is different.



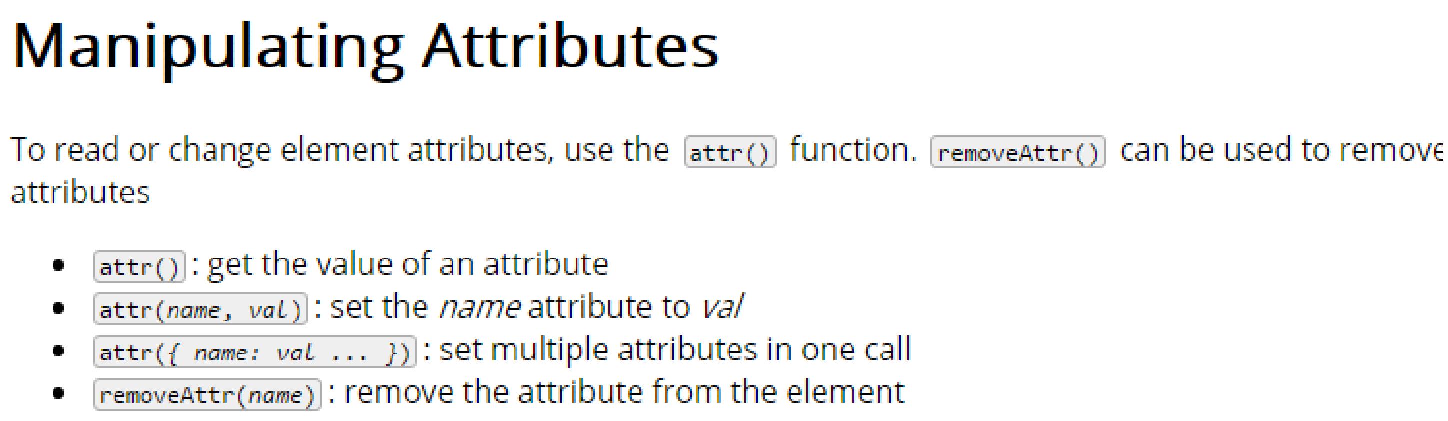
```
$("#example p[id]").replaceWith("<div>replaced</div>");
//$("#example p").replaceWith(replacementFn);
```

Here we first selecting we select and then replace with.
(take these and replacewith that)

- USE Replacewith() as it is also valid to pass on callback() to **replacewith()** which is not valid in replaceAll(), also it has same structure as others(first select then argument).

Manipulating attributes

Friday, March 31, 2017 11:35 PM



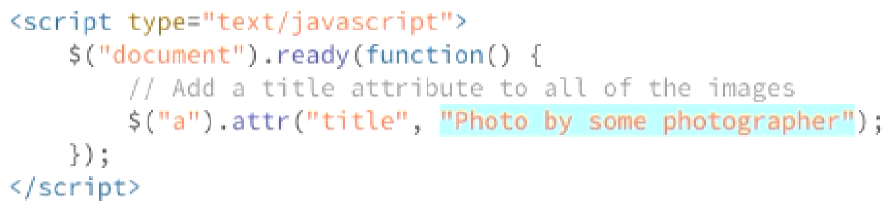
Manipulating Attributes

To read or change element attributes, use the `attr()` function. `removeAttr()` can be used to remove attributes

- `attr()` : get the value of an attribute
- `attr(name, val)` : set the *name* attribute to *val*
- `attr({ name: val ... })` : set multiple attributes in one call
- `removeAttr(name)` : remove the attribute from the element

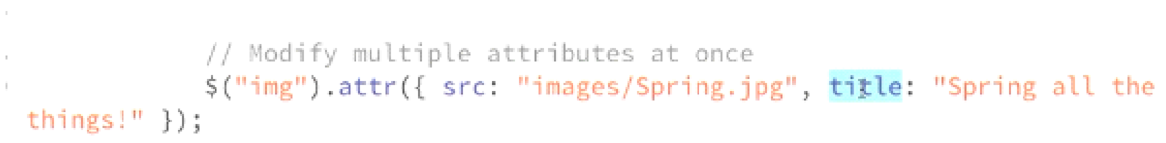
Accessing and Deleting Attribute

- Attr(name,value) function can be used to access and manipulate the value of attribute of selected element.
- If second argument is not passed then we can retrieve the value of attribute.



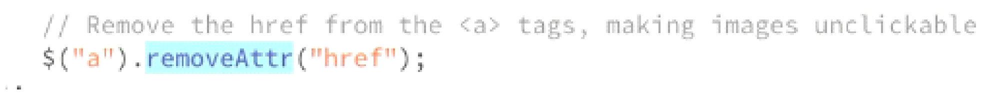
```
<script type="text/javascript">
    $("document").ready(function() {
        // Add a title attribute to all of the images
        $("a").attr("title", "Photo by some photographer");
    });
</script>
```

- We can manipulate multiple attribute by passing JS object having **attribute:value** ,



```
            // Modify multiple attributes at once
            $("img").attr({ src: "images/Spring.jpg", title: "Spring all the
things!" });
```

Removing Attribute

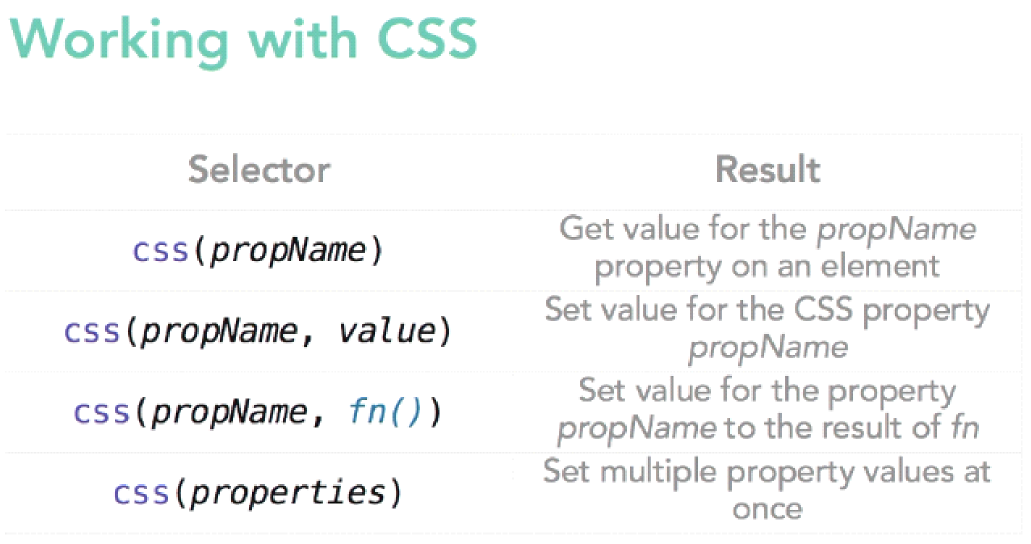


```
  // Remove the href from the <a> tags, making images unclickable
  $("a").removeAttr("href");
```

Working with CSS

Saturday, April 1, 2017 12:09 AM

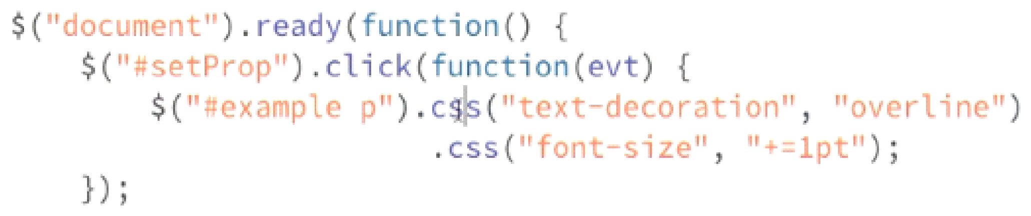
- We have been using css() function for setting styling for our selected element by now, but there is a lot more to it.
- Jquery makes is very easy to work with



Working with CSS

Selector	Result
css(*propName*)	Get value for the *propName* property on an element
css(*propName*, *value*)	Set value for the CSS property *propName*
css(*propName*, *fn()*)	Set value for the property *propName* to the result of *fn*
css(*properties*)	Set multiple property values at once

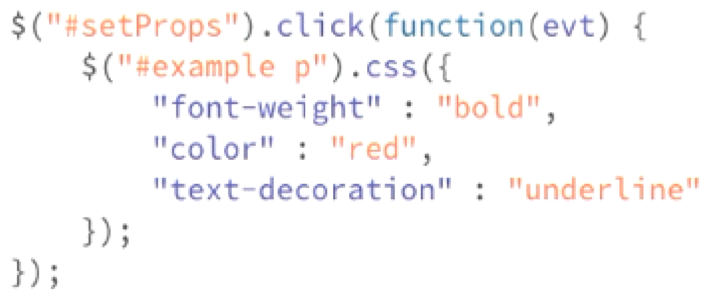
Example 1- setting single property at a time using 2nd format of css().



```
$("document").ready(function() {
    $("#setProp").click(function(evt) {
        $("#example p").css("text-decoration", "overline")
                        .css("font-size", "+=1pt");
    });
```

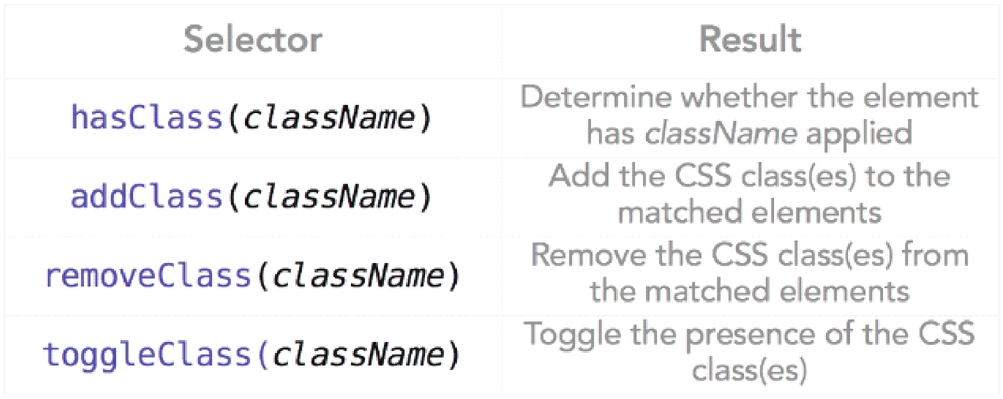
- In the second property, we have used **+=1pt Jquery** will see this syntax and increase by 1.

Example 2 - setting multiple properties at a time using 4th format of css(), by passing all properties as JS function.



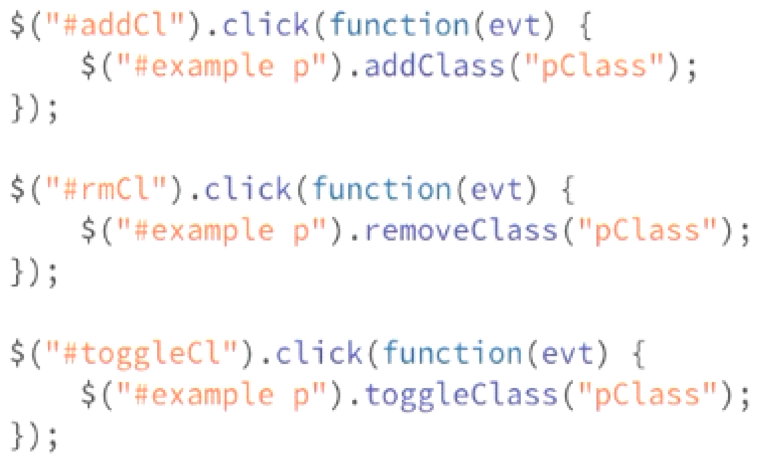
```
$("#setProps").click(function(evt) {
    $("#example p").css({
        "font-weight" : "bold",
        "color" : "red",
        "text-decoration" : "underline"
    });
});
```

Manipulating Classes



Selector	Result
`hasClass(className)`	Determine whether the element has *className* applied
`addClass(className)`	Add the CSS class(es) to the matched elements
`removeClass(className)`	Remove the CSS class(es) from the matched elements
`toggleClass(className)`	Toggle the presence of the CSS class(es)

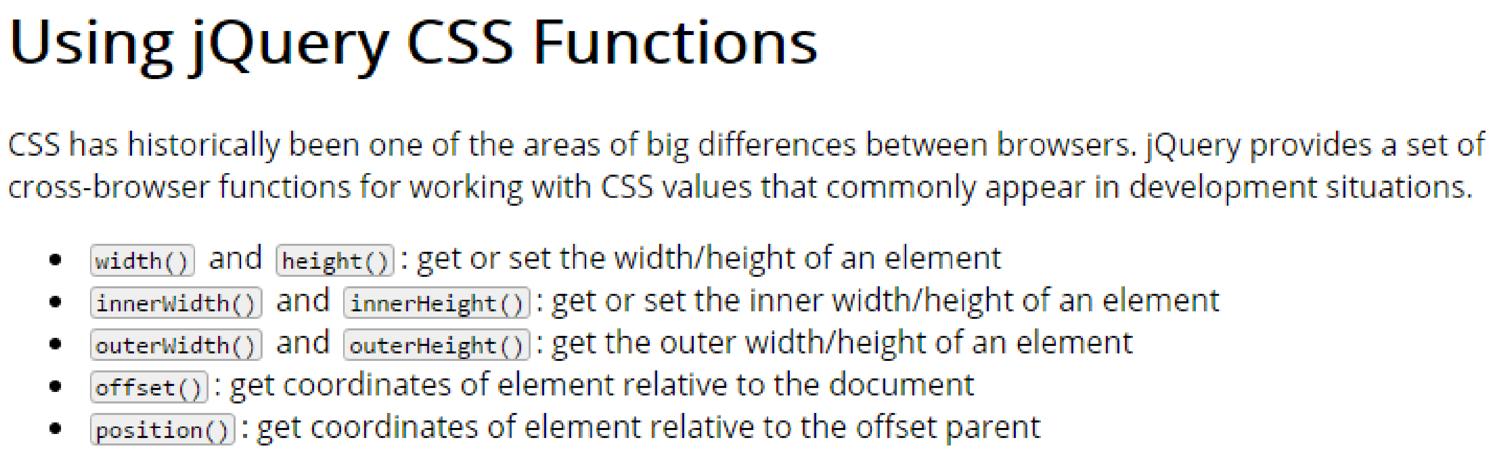
- These function add, remove and toggle a css class, which is equivalent to using **classList** in simple dom.



```
$("#addCl").click(function(evt) {
    $("#example p").addClass("pClass");
});

$("#rmCl").click(function(evt) {
    $("#example p").removeClass("pClass");
});

$("#toggleCl").click(function(evt) {
    $("#example p").toggleClass("pClass");
});
```

Using CSS functions

Saturday, April 1, 2017 12:54 AM

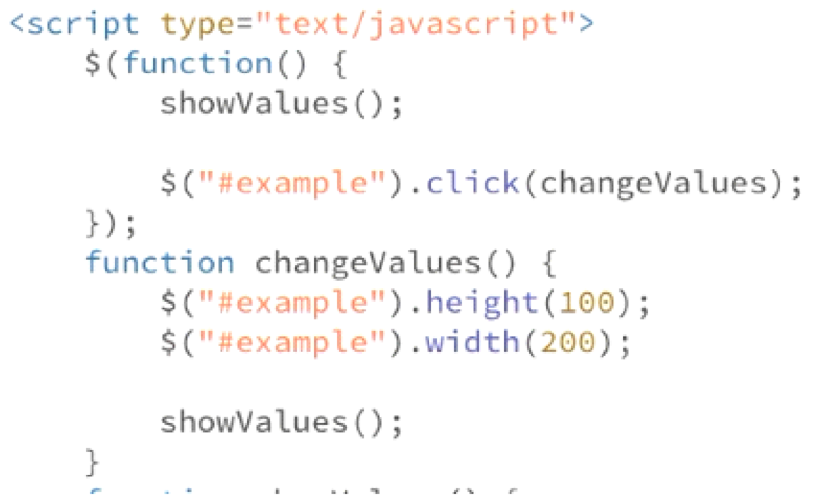


Using jQuery CSS Functions

CSS has historically been one of the areas of big differences between browsers. jQuery provides a set of cross-browser functions for working with CSS values that commonly appear in development situations.

- `width()` and `height()`: get or set the width/height of an element
- `innerWidth()` and `innerHeight()`: get or set the inner width/height of an element
- `outerWidth()` and `outerHeight()`: get the outer width/height of an element
- `offset()`: get coordinates of element relative to the document
- `position()`: get coordinates of element relative to the offset parent

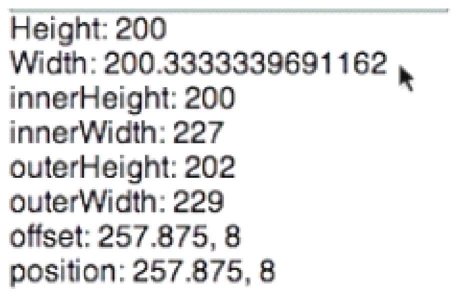
- In the following example (leave showValues() it is written to show on screen),
  we are setting height() and width() of elements.
  If values are not passed they return the value.



```
<script type="text/javascript">
    $(function() {
        showValues();

        $("#example").click(changeValues);
    });
    function changeValues() {
        $("#example").height(100);
        $("#example").width(200);

        showValues();
    }
```

Content of showValues() function

- The important thing to note here is that, even we has set width(200) , it is not necessarily 200,
  so we have to take care of these things and this is because browser renders css in different ways.(so we might use round() function to check if we are checking at runtime).
- Different browsers show different values (in float) , so can't match the exact value at runtime.



```
Height: 200
Width: 200.333339691162
innerHeight: 200
innerWidth: 227
outerHeight: 202
outerWidth: 229
offset: 257.875, 8
position: 257.875, 8
```

Adding Custom data

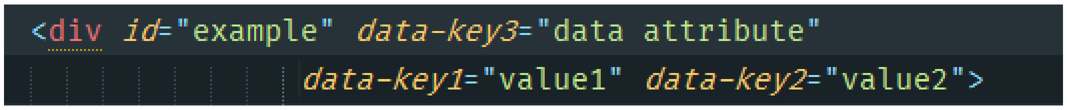
Saturday, April 1, 2017 1:13 AM

Using the Data Methods

- Jquery provides some pretty handy ways to interact with custom data attributes in HTML, Like "data-coolness=200".
- Jquery data Methods can be used for this purpose.

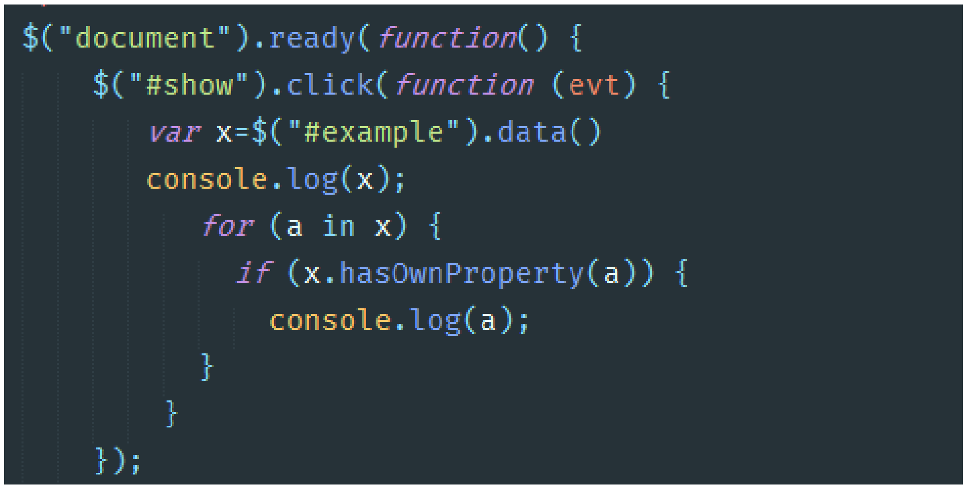
Accessing Data-variables

- Let we have following <div> with 3 data variables.



```
<div id="example" data-key3="data attribute"
        data-key1="value1" data-key2="value2">
```

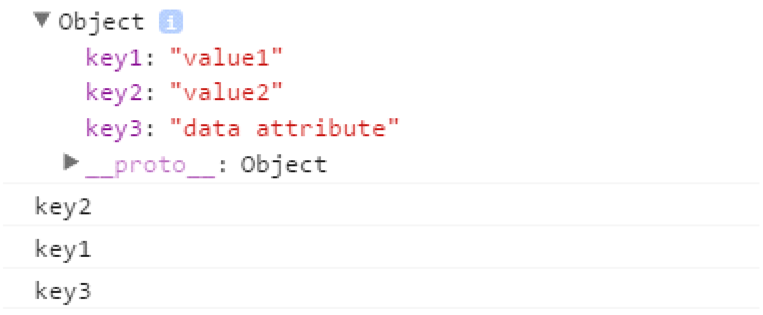
- Now we have a #show button , on clicking which we have to show data.



```
$("document").ready(function() {
    $("#show").click(function (evt) {
        var x=$("#example").data()
        console.log(x);
          for (a in x) {
            if (x.hasOwnProperty(a)) {
              console.log(a);
            }
          }
    });
```

OUTPUT -

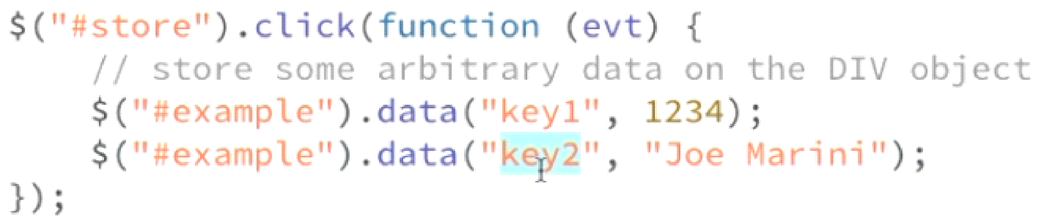
- Data() function return a object which has all "data-xxx" as "xxx" keys.
- And now we can access them using foreach loop.



```
▼Object
   key1: "value1"
   key2: "value2"
   key3: "data attribute"
  ▶__proto__: Object
key2
key1
key3
```

Modifying Data-variable

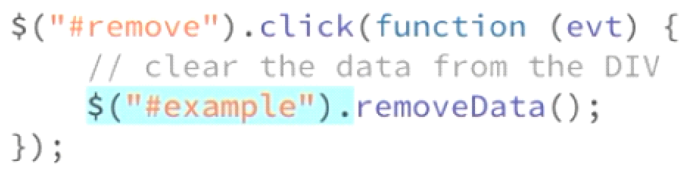
- It will update if already there, or add.



```
$("#store").click(function (evt) {
    // store some arbitrary data on the DIV object
    $("#example").data("key1", 1234);
    $("#example").data("key2", "Joe Marini");
});
```

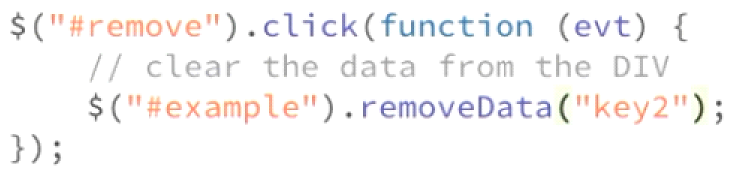
Removing Data-variable

- This function will remove all data-xxx attributes from the selected element if name is not passed.



```
$("#remove").click(function (evt) {
    // clear the data from the DIV
    $("#example").removeData();
});
```

- This will remove data-xxx attributes from the selected element whose name is passed.

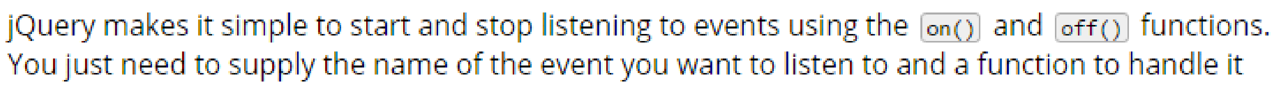


```
$("#remove").click(function (evt) {
    // clear the data from the DIV
    $("#example").removeData("key2");
});
```

Binding and Unbinding events

Saturday, April 1, 2017 1:42 AM

- Jquery handles all support for older browser etc. so we do not need to worry about anything at all.

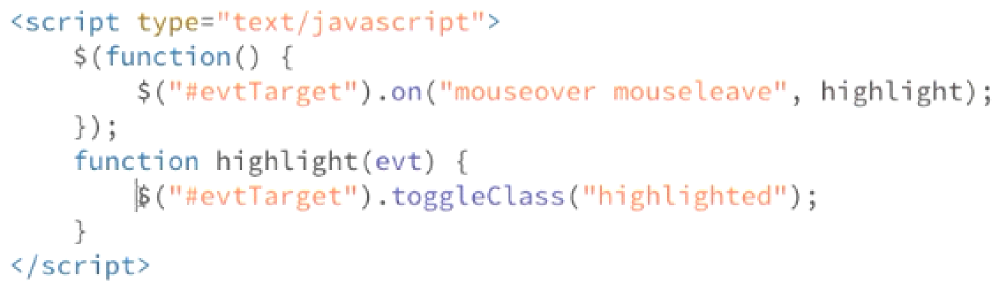


jQuery makes it simple to start and stop listening to events using the `on()` and `off()` functions. You just need to supply the name of the event you want to listen to and a function to handle it

- There are many deprecated functions, for binding and unbinding event.
    - Modern way is using on() and off() methods.

EXAMPLE 1 -

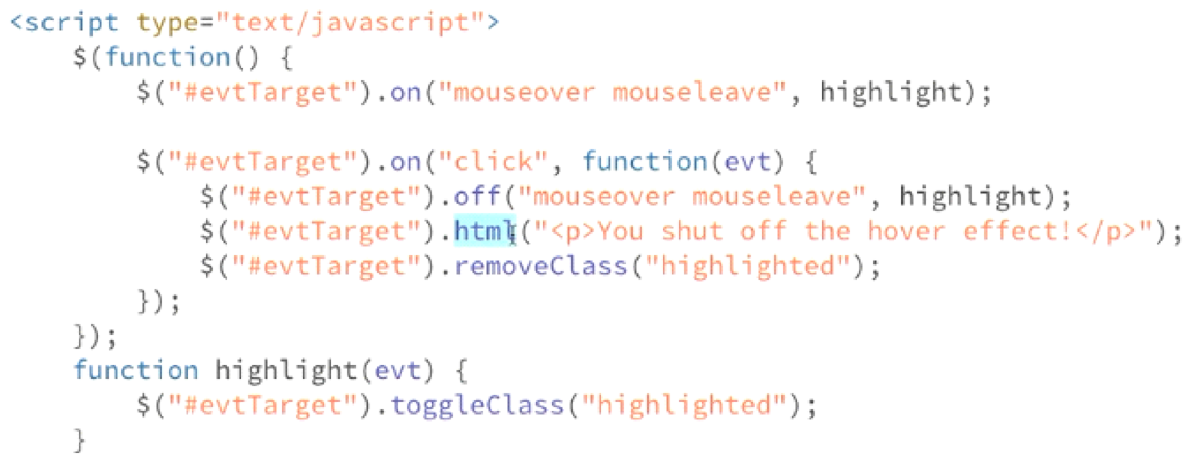
- Here in this example we are registering two events on #evtTarget and the handler is "highlight" function, which is just toggeling the class.



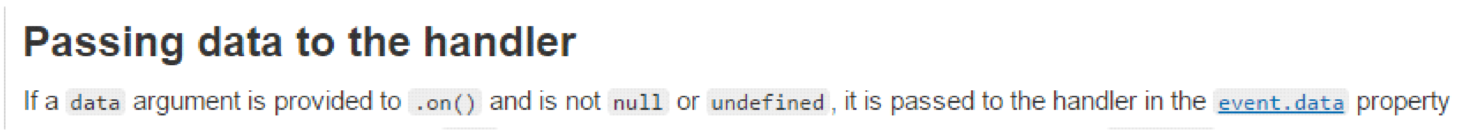
```
<script type="text/javascript">
    $(function() {
        $("#evtTarget").on("mouseover mouseleave", highlight);
    });
    function highlight(evt) {
        $("#evtTarget").toggleClass("highlighted");
    }
</script>
```

EXAMPLE 2-

- Here on clicking the area , we turn of the older events.
  similary like **removeEventListener(type,callback)** here also we have to pass the same handler, which was used in **on()** for registering event.

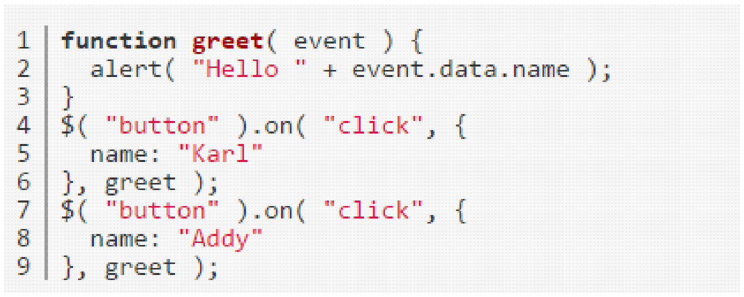


```
<script type="text/javascript">
    $(function() {
        $("#evtTarget").on("mouseover mouseleave", highlight);

        $("#evtTarget").on("click", function(evt) {
            $("#evtTarget").off("mouseover mouseleave", highlight);
            $("#evtTarget").html("<p>You shut off the hover effect!</p>");
            $("#evtTarget").removeClass("highlighted");
        });
    });
    function highlight(evt) {
        $("#evtTarget").toggleClass("highlighted");
    }
```



Passing data to the handler

If a `data` argument is provided to `.on()` and is not `null` or `undefined`, it is passed to the handler in the `event.data` property



```
function greet( event ) {
  alert( "Hello " + event.data.name );
}
$( "button" ).on( "click", {
  name: "Karl"
}, greet );
$( "button" ).on( "click", {
  name: "Addy"
}, greet );
```

- Any data can be passed in the form of Javascript Object

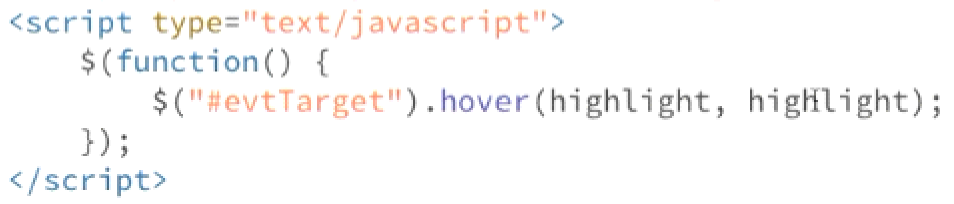
Event Helper Features

Saturday, April 1, 2017 3:21 AM

- There are various shorthand function for events like
  for 'click' we have click() function, which can user directly

```
        $("#elem").click(function(){
           // some code here
        });
```

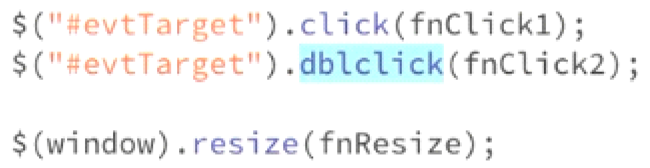
Example 1 - here we using hover() function for 'mousehover' event.
- Here we get same functionality as we get in previous using "mouseover mouseleave" events.
- Highlight() is handler function.



```
<script type="text/javascript">
    $(function() {
        $("#evtTarget").hover(highlight, highlight);
    });
</script>
```

- We pass two methods,
  1- First for hover
  2- Second for hover remove.

Example 2 -here we are using some other handler functions.

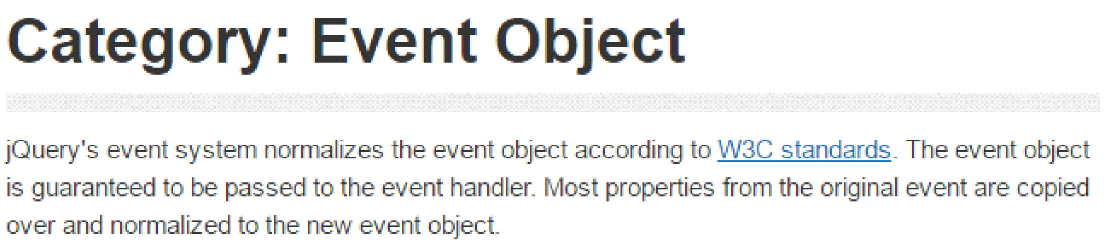


```
$("#evtTarget").click(fnClick1);
$("#evtTarget").dblclick(fnClick2);

$(window).resize(fnResize);
```

Jquery Event object

Saturday, April 1, 2017 3:55 AM

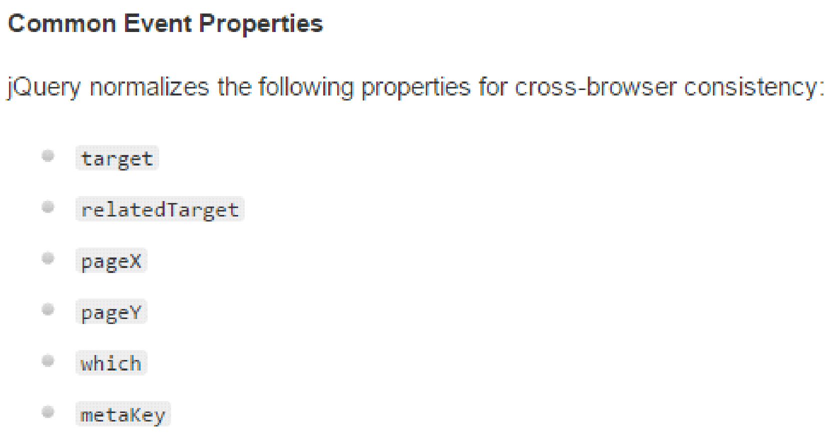
- Jquery provides event object which is consistent across all browsers, this event object is passed to event handler if we pass a variable to event handler.



Category: Event Object

jQuery's event system normalizes the event object according to W3C standards. The event object is guaranteed to be passed to the event handler. Most properties from the original event are copied over and normalized to the new event object.

- All browsers handles event differently but jQuery normalizes the property using it's event Object and makes sure that these properties held for all browsers.



Common Event Properties

jQuery normalizes the following properties for cross-browser consistency:

- `target`
- `relatedTarget`
- `pageX`
- `pageY`
- `which`
- `metaKey`

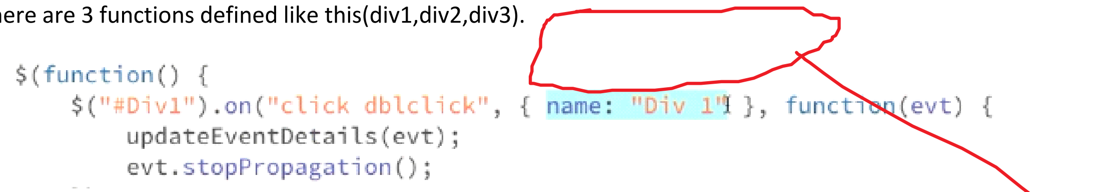
- All properties may not be filled according to event type.
  - Target - element to which event occurred.
  - relatedTarget - another element to which event may be associated.
  - pageX, pageY - cordinates of mouse when event occurred.
  - Which - in keypress, it tell which key pressed, or in mouse event which mouse button was clicked.
  - Metakey - boolean property tell if meta key was pressed or not(meta - window key)

- There are many other properties that may be filled according to event.

Example -

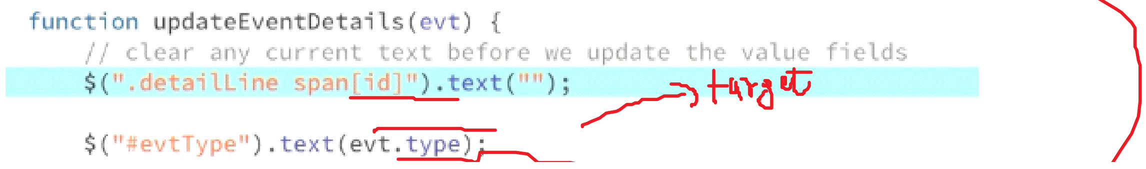
- Here we have passed id of <div> , which can be accessed through, **evt.data.name,** where **Evt** is event object, here we stop propagation using **stopPropagation()** function, which is same(here defined for jquery Event object) dom event function **stopPropagation()**

  There are 3 functions defined like this(div1,div2,div3).



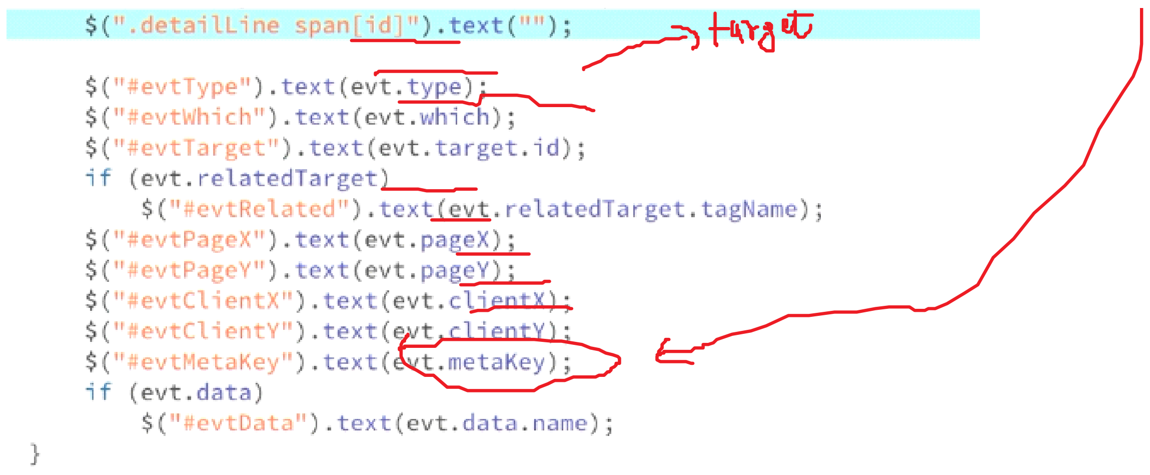
```
$(function() {
    $("#Div1").on("click dblclick", { name: "Div 1" }, function(evt) {
        updateEventDetails(evt);
        evt.stopPropagation();
```

- In updateEventDetails() handler, we



```
function updateEventDetails(evt) {
    // clear any current text before we update the value fields
    $(".detailLine span[id]").text("");

    $("#evtType").text(evt.type);
```

$\rightarrow$ target

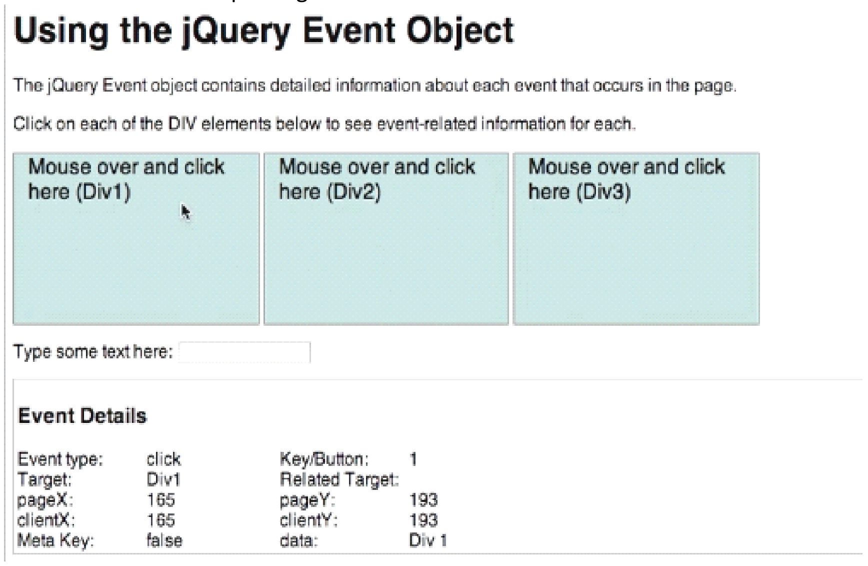


```
        $(".detailLine span[id]").text("");

        $("#evtType").text(evt.type);
        $("#evtWhich").text(evt.which);
        $("#evtTarget").text(evt.target.id);
        if (evt.relatedTarget)
            $("#evtRelated").text(evt.relatedTarget.tagName);
        $("#evtPageX").text(evt.pageX);
        $("#evtPageY").text(evt.pageY);
        $("#evtClientX").text(evt.clientX);
        $("#evtClientY").text(evt.clientY);
        $("#evtMetaKey").text(evt.metaKey);
        if (evt.data)
            $("#evtData").text(evt.data.name);
    }
```

→ target

- Here we can see we puts out different values for event object **evt,** here also we can see we have used **evt.data.name** from data passed.

OUTPUT - values are updating as mouse moved



Using the jQuery Event Object

The jQuery Event object contains detailed information about each event that occurs in the page.

Click on each of the DIV elements below to see event-related information for each.

Mouse over and click here (Div1)
Mouse over and click here (Div2)
Mouse over and click here (Div3)
Type some text here:

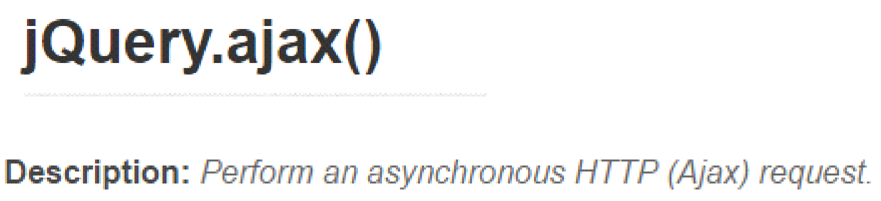
Event Details

Event type:	click	Key/Button:	1
Target:	Div1	Related Target:	
pageX:	165	pageY:	193
clientX:	165	clientY:	193
Meta Key:	false	data:	Div 1

jQuery and ajax

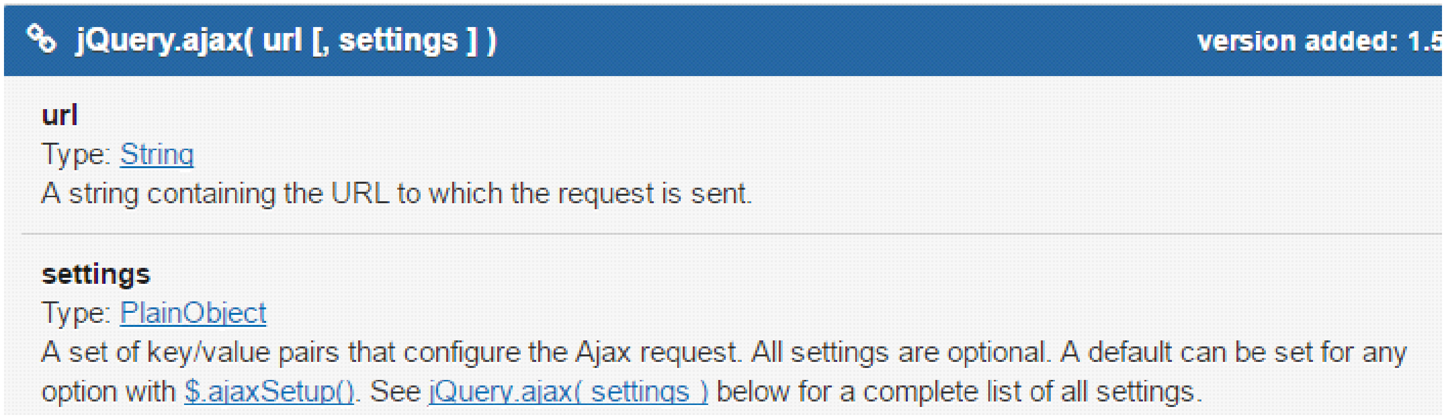
Saturday, April 1, 2017   12:33 PM

- Ajax() global function can be used to fire Asynchronous ajax request in Jquery.



jQuery.ajax()

Description: *Perform an asynchronous HTTP (Ajax) request.*



jQuery.ajax(url [, settings]) **version added: 1.5**

url
Type: String
A string containing the URL to which the request is sent.

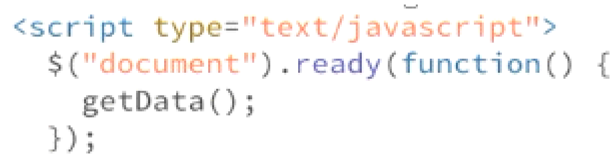
settings
Type: PlainObject
A set of key/value pairs that configure the Ajax request. All settings are optional. A default can be set for any option with $.ajaxSetup(). See jQuery.ajax(settings) below for a complete list of all settings.

- Settings can be passed along with url, or using same function only with "settings".
- "settings" is a jS object.

jQuery.ajax([settings])

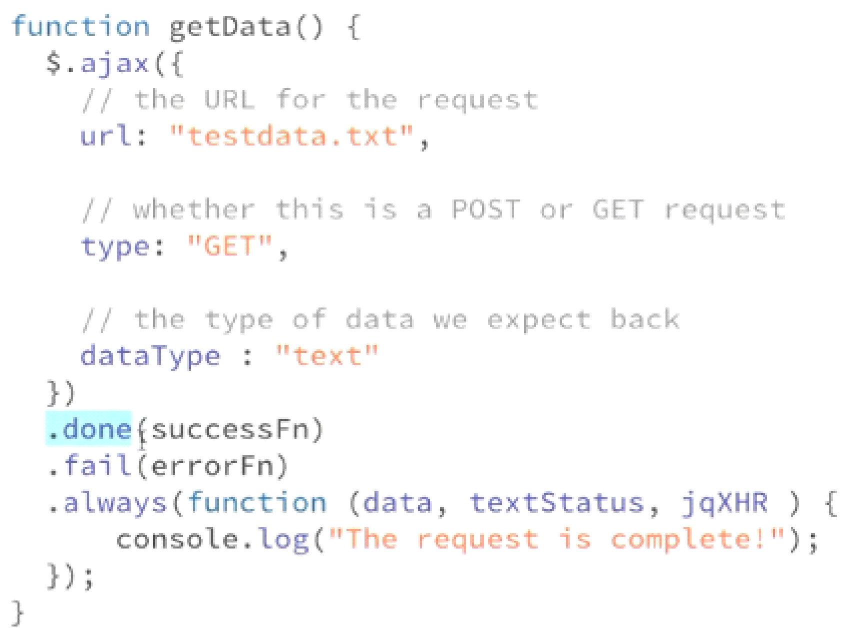
- Function used for setting ajax settings.

Example - retrieving 'text' data



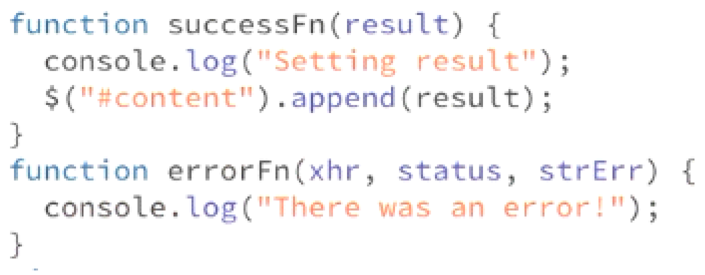
```
<script type="text/javascript">
  $("document").ready(function() {
    getData();
  });
```

- Here in getData() , we defining global ajax method to get data from url.



```
function getData() {
  $.ajax({
    // the URL for the request
    url: "testdata.txt",

    // whether this is a POST or GET request
    type: "GET",

    // the type of data we expect back
    dataType : "text"
  })
  .done(successFn)
  .fail(errorFn)
  .always(function (data, textStatus, jqXHR ) {
      console.log("The request is complete!");
  });
}
```

- As we can see three functions are applied on given object which will run on different cases
    - At success - done()
    - At failure - fail()
    - Always - always()
        - Handlers re passed to these functions/events.

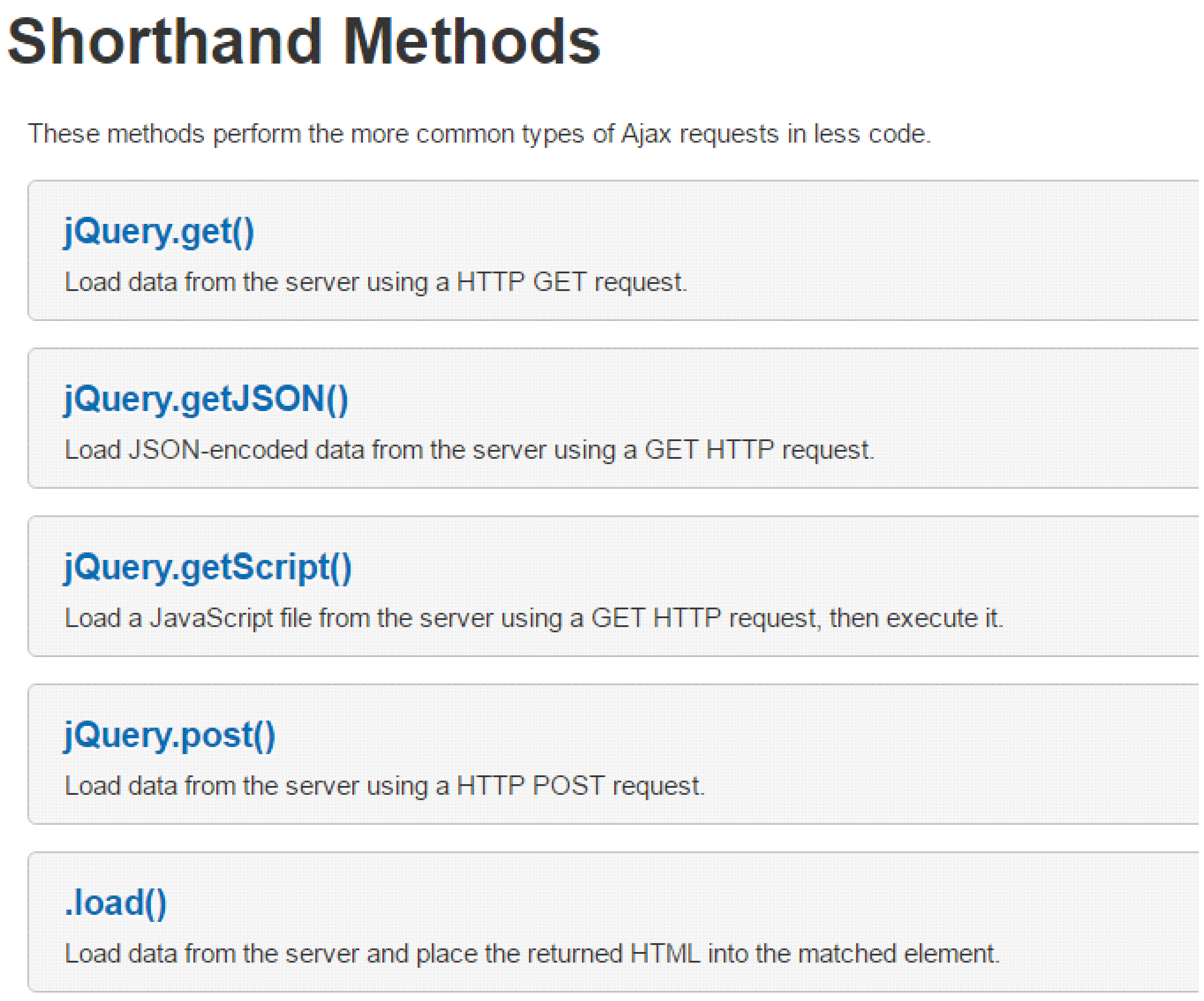


```
function successFn(result) {
  console.log("Setting result");
  $("#content").append(result);
}
function errorFn(xhr, status, strErr) {
  console.log("There was an error!");
}
```

Ajax helper functions

Saturday, April 1, 2017 1:15 PM

- Some scenarios/patterns happens pretty frequently with ajax so jquery provides builtin helpers for them.



Shorthand Methods

These methods perform the more common types of Ajax requests in less code.

jQuery.get()
Load data from the server using a HTTP GET request.

jQuery.getJSON()
Load JSON-encoded data from the server using a GET HTTP request.

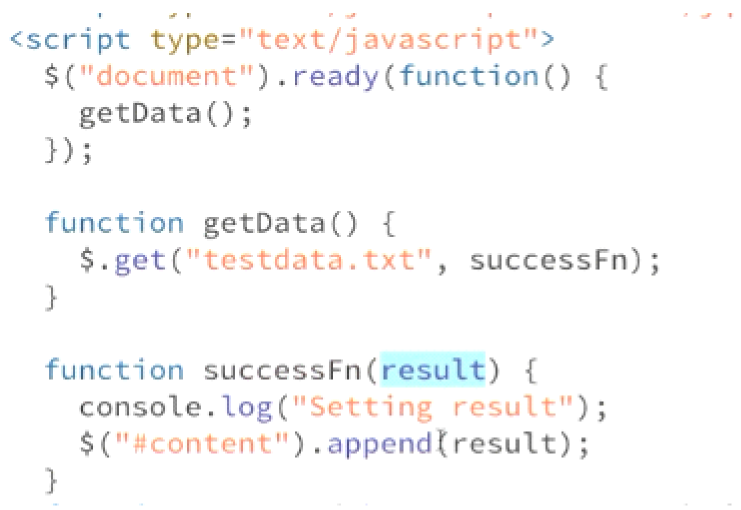
jQuery.getScript()
Load a JavaScript file from the server using a GET HTTP request, then execute it.

jQuery.post()
Load data from the server using a HTTP POST request.

.load()
Load data from the server and place the returned HTML into the matched element.

Get()

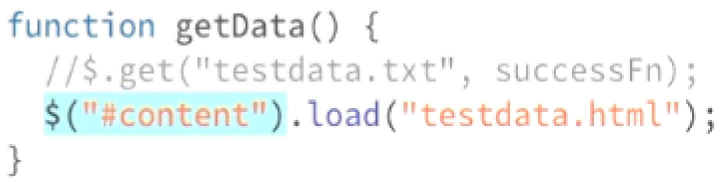
- Can be used to get data(any type).



```
<script type="text/javascript">
  $("document").ready(function() {
    getData();
  });

  function getData() {
    $.get("testdata.txt", successFn);
  }

  function successFn(result) {
    console.log("Setting result");
    $("#content").append(result);
  }
```

load()

- Load function can be used to get HTML data, for $.ajax() we can set dataType:'html'.
- The data return will be in form of HTML.



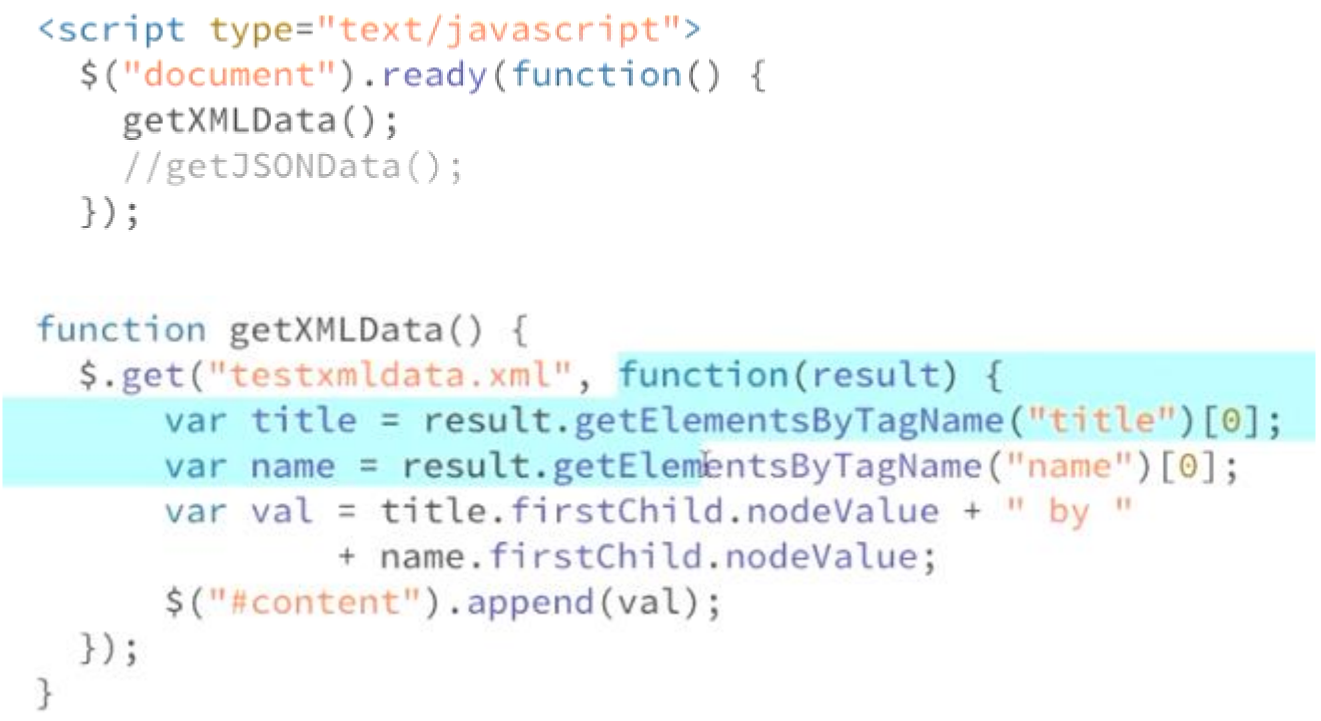
```
function getData() {
  //$.get("testdata.txt", successFn);
  $("#content").load("testdata.html");
}
```

Working with Different type of data

Saturday, April 1, 2017 1:27 PM

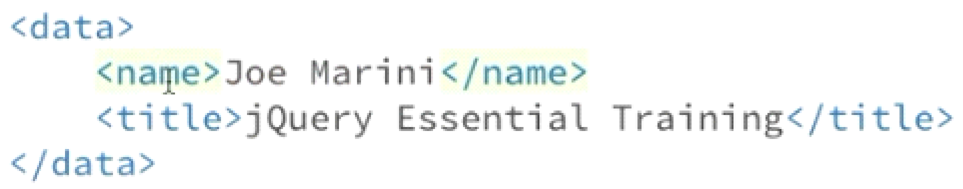
- Jquery can work with different types of data, either that could be JSON, XML etc.

Working with XML data



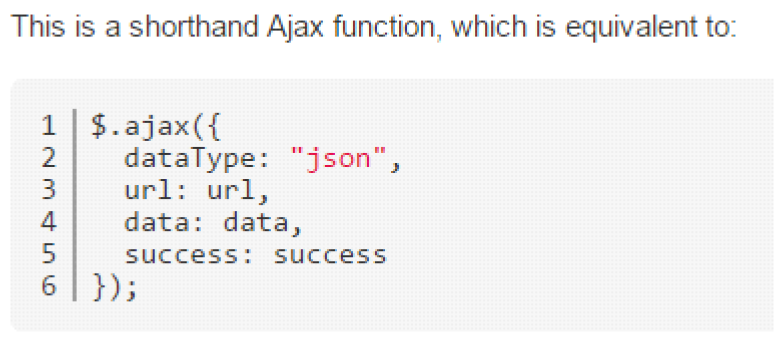
```
<script type="text/javascript">
  $("document").ready(function() {
    getXMLData();
    //getJSONData();
  });


function getXMLData() {
  $.get("testxmldata.xml", function(result) {
      var title = result.getElementsByTagName("title")[0];
      var name = result.getElementsByTagName("name")[0];
      var val = title.firstChild.nodeValue + " by "
              + name.firstChild.nodeValue;
      $("#content").append(val);
  });
}
```

- As we can see we can use get(), function and then we can use w3c, function **getElementsByTagName()** function to retrieve the values as values are inside xml tags.



```
<data>
    <name>Joe Marini</name>
    <title>jQuery Essential Training</title>
</data>
```

Working with JSON data



This is a shorthand Ajax function, which is equivalent to:

```
$.ajax({
  dataType: "json",
  url: url,
  data: data,
  success: success
});
```

EXAMPLE - 1

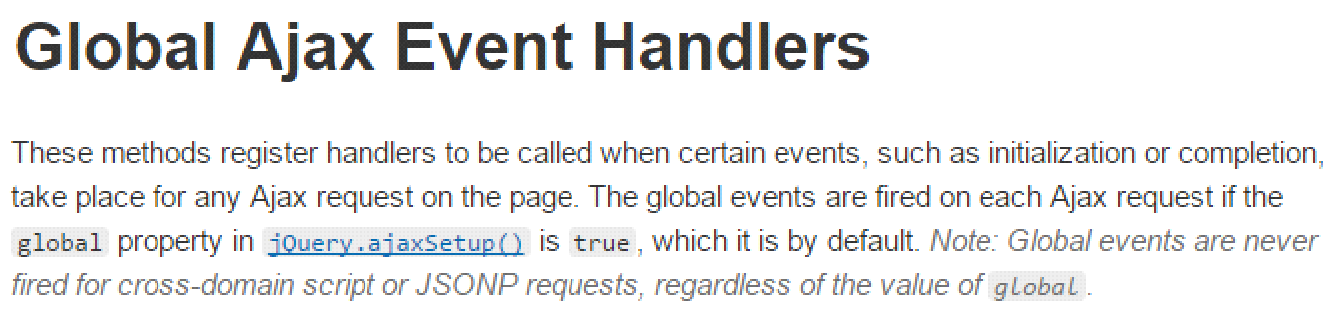
```
$.getJSON( "ajax/test.json", function( data ) {
  var items = [];
  $.each( data, function( key, val ) {
    items.push( "<li id='" + key + "'>" + val + "</li>" );
  });
```

EXAMPLE - 2

```
// Assign handlers immediately after making the request,
// and remember the jqxhr object for this request
var jqxhr = $.getJSON( "example.json", function() {
  console.log( "success" );
})
  .done(function() {
    console.log( "second success" );
  })
  .fail(function() {
    console.log( "error" );
  })
  .always(function() {
    console.log( "complete" );
  });

// Perform other work here ...

// Set another completion function for the request above
jqxhr.complete(function() {
  console.log( "second complete" );
});
```

Using global Ajax Handlers

Saturday, April 1, 2017 1:45 PM

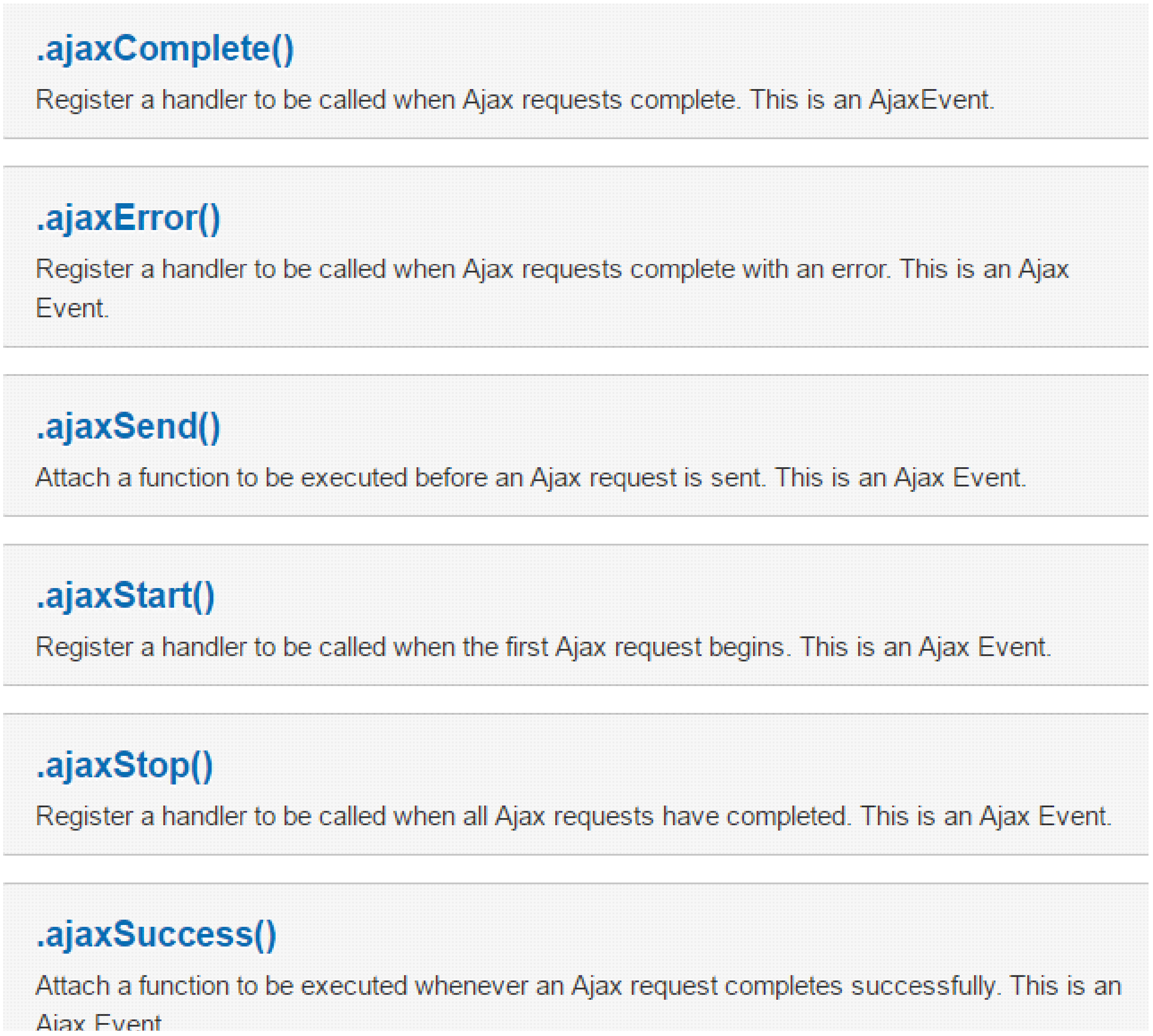


Global Ajax Event Handlers

These methods register handlers to be called when certain events, such as initialization or completion, take place for any Ajax request on the page. The global events are fired on each Ajax request if the `global` property in `jQuery.ajaxSetup()` is `true`, which it is by default. *Note: Global events are never fired for cross-domain script or JSONP requests, regardless of the value of* `global`.

- Jquery ajax module provides set of global event handlers that we can use to register functions to listen all interesting events which takes place in ajax request-response cycle.
- These functions can be used to set some events which will be triggered during ajax request life cycle.

NOTE - THESE ARE THE GLOBAL HANDLERS AND CALLED FOR EVERY REQUEST ON THE PAGE.



.ajaxComplete()

Register a handler to be called when Ajax requests complete. This is an AjaxEvent.

.ajaxError()

Register a handler to be called when Ajax requests complete with an error. This is an Ajax Event.

.ajaxSend()

Attach a function to be executed before an Ajax request is sent. This is an Ajax Event.

.ajaxStart()

Register a handler to be called when the first Ajax request begins. This is an Ajax Event.

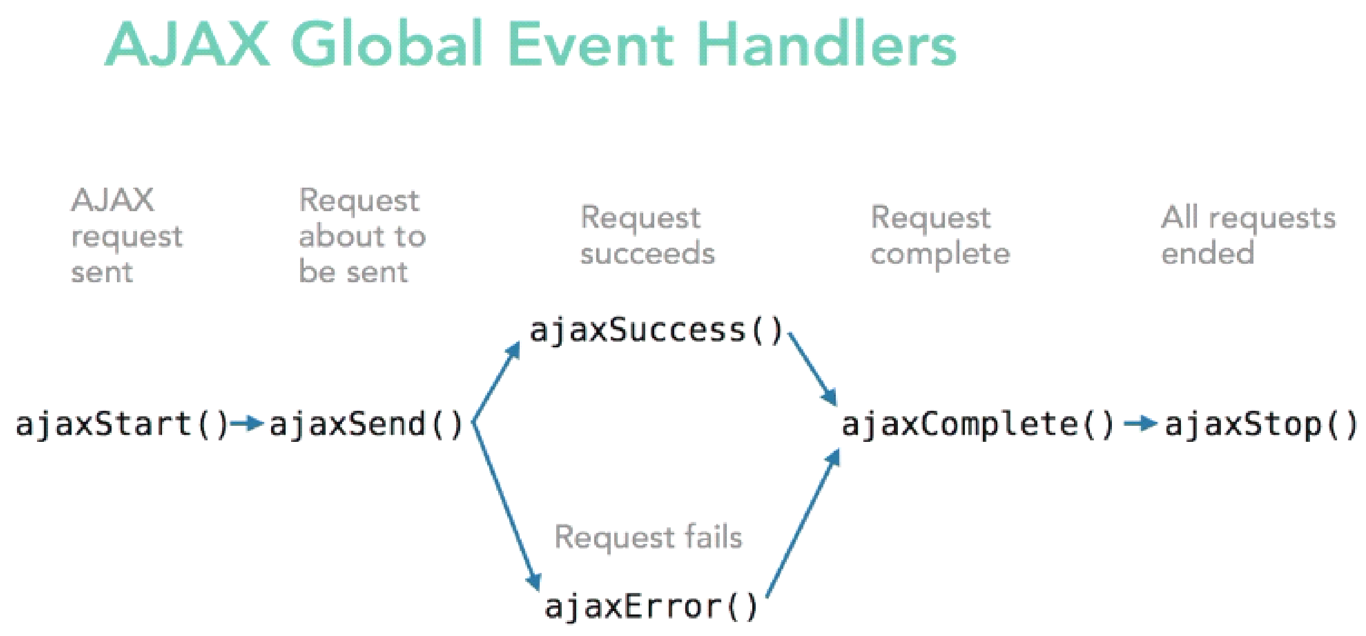
.ajaxStop()

Register a handler to be called when all Ajax requests have completed. This is an Ajax Event.

.ajaxSuccess()

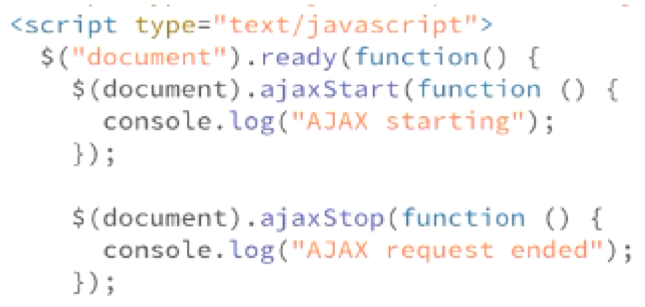
Attach a function to be executed whenever an Ajax request completes successfully. This is an Ajax Event

Ajax Event

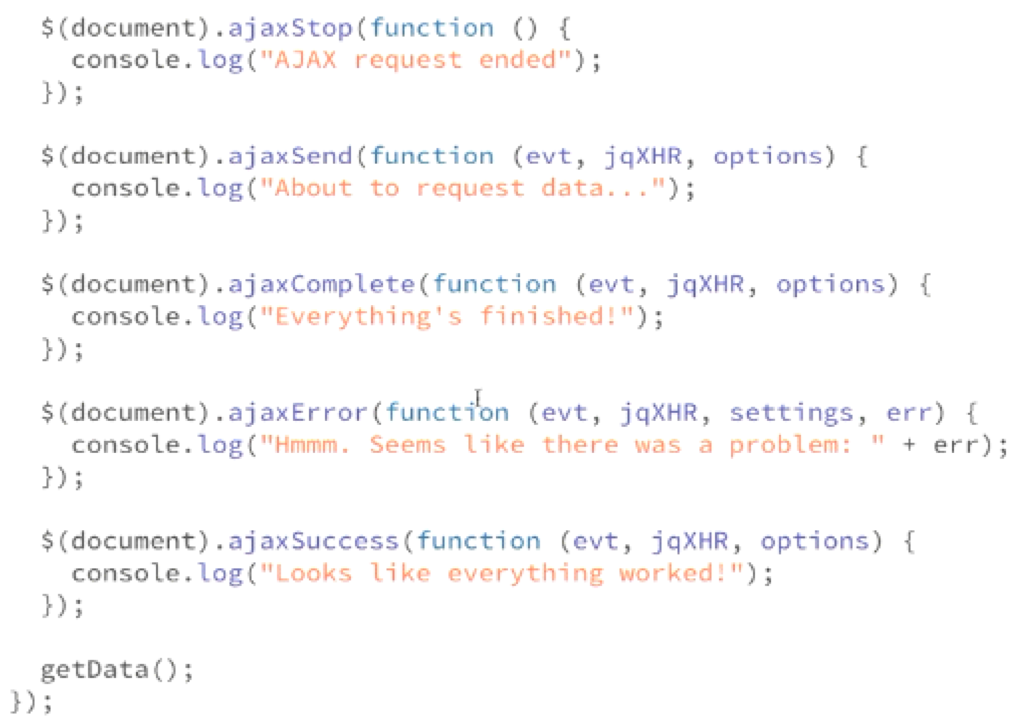


Example -

- Here we are setting event, which will be called when ajax life cycle happens.

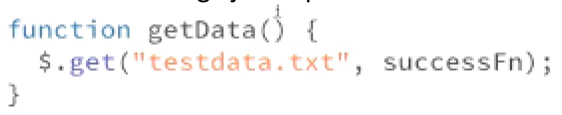


```
<script type="text/javascript">
  $("document").ready(function() {
    $(document).ajaxStart(function () {
      console.log("AJAX starting");
    });

    $(document).ajaxStop(function () {
      console.log("AJAX request ended");
    });
```



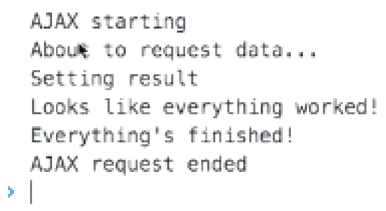
```
  $(document).ajaxStop(function () {
    console.log("AJAX request ended");
  });

  $(document).ajaxSend(function (evt, jqXHR, options) {
    console.log("About to request data...");
  });

  $(document).ajaxComplete(function (evt, jqXHR, options) {
    console.log("Everything's finished!");
  });

  $(document).ajaxError(function (evt, jqXHR, settings, err) {
    console.log("Hmmm. Seems like there was a problem: " + err);
  });

  $(document).ajaxSuccess(function (evt, jqXHR, options) {
    console.log("Looks like everything worked!");
  });

  getData();
});
```

- Function for sending ajax request.



```
function getData() {
  $.get("testdata.txt", successFn);
}
```

OUTPUT In console-



```
AJAX starting
About to request data...
Setting result
Looks like everything worked!
Everything's finished!
AJAX request ended
```