jQuery

write less, do more.

Prakash R
Email : Prakash.raman@servion.com

servion
CUSTOMER INTERACTION MANAGEMENT SPECIALISTS

Discover
Customer expectations

Design
Compelling interaction strategies

Deliver
The experience your brand promises

Customer Experience by Design

Agenda

- Introduction to jQuery
- Selectors
- Events
- HTML Manipulation
- CSS Manipulation
- HTML Traversing
- AJAX
- Effects

Introduction

Introduction

- jQuery is a fast, small, and feature-rich JavaScript library.
- Cross browser capability
- jQuery has simple APIs for the following things
  - HTML document traversing and manipulation
  - CSS manipulation
  - Event handling
  - Ajax
  - Effects

Installation

- Download the jQuery library from jQuery.com.

```
<html>

    <head>
        //should be at the top of all other dependent libraries
        <script type="" src="jquery.js"></script>
    </head>

    <body>
    </body>
</html>
```

Syntax

- jQuery syntax is made for
  - selecting HTML elements
  - performing some action on the element(s).
- Basic syntax is: $(selector).action()
  - A $ sign to define/access jQuery. We can also use “jQuery” instead of $.
  - A (selector) to "query (or find)" HTML elements
  - A jQuery action() to be performed on the element(s)

Syntax - Examples

- $(this).hide() - hides the current element.
- $("p").hide() - hides all elements.
- $(".test").hide() - hides all elements with class="test".

- $("#test").hide() - hides the element with id="test".

Document Ready Event

- It is good practice to wait for the document to be fully loaded and prevent any jQuery code from running before the document is finished loading.

- Some examples of actions that can fail if methods are run before the document is fully loaded:

  - Trying to hide an element that is not created yet
  - Trying to get the size of an image that is not loaded yet

```
$(document).ready(function(){

  // jQuery methods go here...

});
```

noConflict()

- There are many other popular JavaScript frameworks which may use “$” as shortcut identifier.
- If two different frameworks are using the same shortcut, one of them might stop working.
- noConflict() method releases the hold on the $ shortcut identifier, so that other scripts can use it.
- Use jQuery, simply by writing the full name instead of the shortcut.

```
$.noConflict();
jQuery(document).ready(function(){
   jQuery("button").click(function(){
      jQuery("p").text("jQuery is still working!");
   });
});
```

noConflict()

- You can create your own shortcut very easily.
- noConflict() method returns a reference to jQuery, that you can save in a variable, for later use.

```
var jq = $.noConflict();

jq(document).ready(function(){
   jq("button").click(function(){
      jq("p").text("jQuery is still working!");
   });
});
```

noConflict()

- If a block of jQuery code uses the $ shortcut and you do not want to change it all, you can pass the $ sign in as a parameter to the ready method.

- This allows you to access jQuery using $, inside this function - outside of it, you will have to use "jQuery"

```
$.noConflict();

jQuery(document).ready(function($){
   $("button").click(function(){
      $("p").text("jQuery is still working!");
   });
});
```

Selectors

Selectors

- jQuery selectors allow you to select and manipulate HTML element(s).
- jQuery selectors are used to select HTML elements based on their name, id, classes, types, attributes, values of attributes and much more.
- All selectors in jQuery start with the dollar sign and parentheses: $().
- We can also use "jQuery" instead of $: jQuery()

element Selector

- jQuery element selector selects elements based on the element name.
- Select all <p> elements on a page like this : $("p")

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

#id Selector

- jQuery #id selector uses the id attribute of an HTML tag to find the specific element.

- An id should be unique within a page, so you should use the #id selector when you want to find a single, unique element.

```
$(document).ready(function(){

        $(“#test").hide();

});

<p id=”test”></p>
```

.class Selector

- jQuery class selector finds elements with a specific class.
- You can use class selector if you want to select a group of elements with similar operations.

```
$(document).ready(function(){

        $(".mylink").click (function(){
                // do some operation...
        );

});

<a class="mylink" href="www.google.com">Google</a>
<a class="mylink" href="www.facebook.com">Facebook</a>
```

More Selectors

Syntax	Description
$("*")	Selects all elements
$(this)	Selects the current HTML element
$("p.intro")	Selects all <p> elements with class="intro"
$("p:first")	Selects the first <p> element
$("ul li:first")	Selects the first element of the first
$("ul li:first-child")	Selects the first element of every
$("[href]")	Selects all elements with an href attribute
$("a[target='_blank']")	Selects all <a> elements with a target attribute value equal to "_blank"
$("a[target!='_blank']")	Selects all <a> elements with a target attribute value NOT equal to "_blank"
$(":button")	Selects all <button> elements and <input> elements of type="button"
$("tr:even")	Selects all even <tr> elements
$("tr:odd")	Selects all odd <tr> elements

Events

Events

Mouse Events	Keyboard Events	Form Events	Document/Window Events
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

Syntax For Event Methods

- To assign a click event to all paragraphs on a page.

```
$("p").click();
```

- To define what should happen when the event fires, You must pass a function to the event.

```
$("p").click(function(){
  // action goes here!!
});
```

on() Method

- on() method attaches one or more event handlers for the selected elements.

```
$("p").on("click", function(){
   $(this).hide();
});
```

```
$("p").on({
   mouseenter: function(){
      $(this).css("background-color", "lightgray");
   },
   mouseleave: function(){
      $(this).css("background-color", "lightblue");
   },
   click: function(){
      $(this).css("background-color", "yellow");
   }
});
```

HTML Manipulation

HTML Manipulation

- jQuery comes with a bunch of DOM (Document Object Model) related methods that make it easy to access and manipulate elements and attributes.

- Available Methods Types

  - Get
  - Set
  - Add
  - Remove

GET

- Get Content - text(), html(), and val()
  - text() - Sets or returns the text content of selected elements
  - html() - Sets or returns the content of selected elements (including HTML markup)
  - val() - Sets or returns the value of form fields

```
$("#btn1").click(function(){
   alert("Text: " + $("#test").text());
});
$("#btn2").click(function(){
   alert("HTML: " + $("#test").html());
});
```

```
$("#btn1").click(function(){
   alert("Value: " + $("#test").val());
});
```

GET

- Get Attributes - attr()

```
$("button").click(function(){
   alert($("#test").attr("href"));
});
```

SET

- Set Content - text(), html(), and val()
  - text() - Sets or returns the text content of selected elements
  - html() - Sets or returns the content of selected elements (including HTML markup)
  - val() - Sets or returns the value of form fields

```
$("#btn1").click(function(){
   $("#test1").text("Hello world!");
});
$("#btn2").click(function(){
   $("#test2").html("<b>Hello world!</b>");
});
$("#btn3").click(function(){
   $("#test3").val("Dolly Duck");
});
```

SET

- Set Attributes - attr()

```
$("button").click(function(){
   $("#test").attr("href", "http://www.google.com");
});


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

   $("#test").attr({
      "href" : "http:// www.google.com",
      "title" : “Google"
   });

});
```

SET - Callback Function

- The callback function has two parameters:
  - index of the current element in the list of elements selected
  - the original value

```
$("# button ").click(function(){
   $("#test").html(function(i, origText){
      return "Old html: " + origText + "
replaced with New html: Hello
<b>world!</b>  (index: " + i + ")";
   });
});
```

```
$("#button").click(function(){
   $("#link").attr("href", function(i, origValue){
      return origValue + "/main.html";
   });
});
```

ADD

- Following methods are used to add content to elements

  - append() - Inserts content at the end of the selected elements
  - prepend() - Inserts content at the beginning of the selected elements
  - after() - Inserts content after the selected elements
  - before() - Inserts content before the selected elements

ADD

```
$("p").append("Some appended text.");
```

```
$("p").prepend("Some prepended text.");
```

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

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

- These methods can take infinite number of new elements as parameters

```
function appendText() {

   var txt1 = "<p>Text.</p>";
   var txt2 = $("<p></p>").text("Text.");
   var txt3 = document.createElement("p");
   txt3.innerHTML = "Text.";
   $("body").append(txt1, txt2, txt3);

}
```

REMOVE

- Following methods are used to remove contents and elements
  - remove() - Removes the selected element (and its child elements)
  - empty() - Removes the child elements from the selected element

```
$("#div1").remove();
```

```
$("#div1").empty();
```

REMOVE - Filters

- remove() method accepts one parameter, which allows you to filter the elements to be removed
- The parameter can be any of the jQuery selector syntaxes.

The following example removes all <p> elements with class="test":

```
$("p").remove(".test");
```

The following example removes all <p> elements with class="test“ and class="demo":

```
$("p").remove(".test, .demo");
```

CSS Manipulation

CSS Manipulation

- Following methods are used for CSS manipulation

  - addClass() - Adds one or more classes to the selected elements
  - removeClass() - Removes one or more classes from the selected elements
  - toggleClass() - Toggles between adding/removing classes from the selected elements
  - css() - Sets or returns the style attribute

CSS Manipulation

```
$("button").click(function(){
   $("h1, h2, p").addClass("blue");
   $("div").addClass("important");
});
```

```
$("button").click(function(){
   $("#div1").addClass("important blue");
});
```

```
$("button").click(function(){
   $("h1, h2, p").removeClass("blue");
});
```

```
$("button").click(function(){
   $("h1, h2, p").toggleClass("blue");
});
```

css()

- css() method sets or returns one or more style properties for the selected elements
  - css("propertyname") - return the value of a specified CSS property
  - css("propertyname","value") - set a specified CSS property
  - css({"propertyname":"value","propertyname":"value",...}) - Set multiple css properties

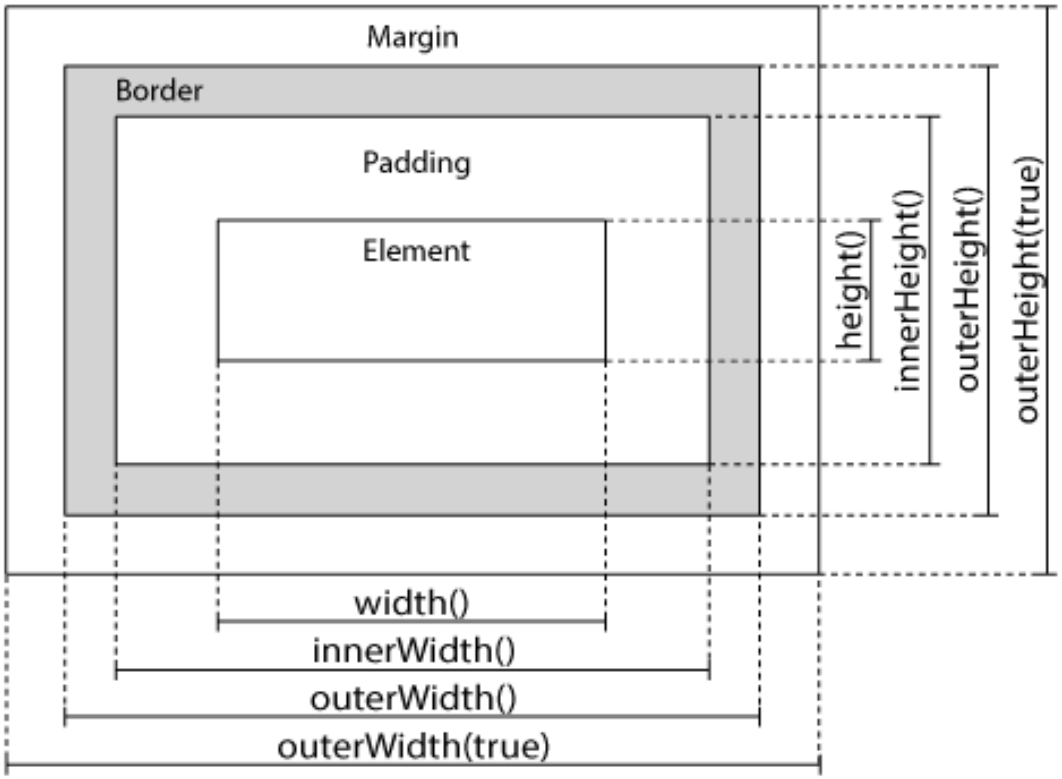
```
$("p").css("background-color");
```

```
$("p").css("background-color", "yellow");
```

```
$("p").css({"background-color": "yellow", "font-size": "200%"});
```

Dimensions

- Following methods are used to get and set dimensions
  - width()
  - height()
  - innerWidth()
  - innerHeight()
  - outerWidth()
  - outerHeight()



Dimensions

- width() method sets or returns the width of an element (excludes padding, border and margin)
- height() method sets or returns the height of an element (excludes padding, border and margin)
- innerWidth() method returns the width of an element (includes padding)
- innerHeight() method returns the height of an element (includes padding)

Dimensions

- outerWidth() method returns the width of an element (includes padding and border)
- outerHeight() method returns the height of an element (includes padding and border)
- outerWidth(true) method returns the width of an element (includes padding, border, and margin)
- outerHeight(true) method returns the height of an element (includes padding, border, and margin)

TRAVERSING

Traversing

- Traversing is used to find HTML elements based on their relation to other elements

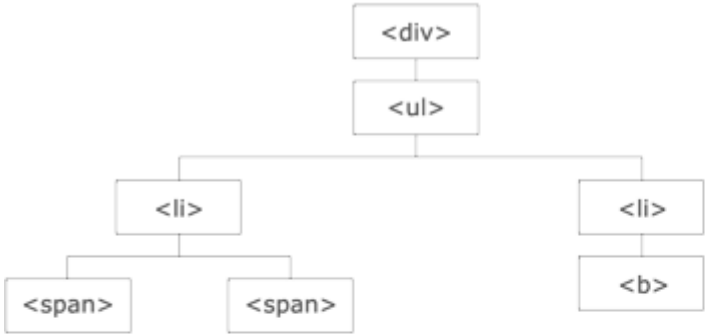


Illustration explained:

- 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

Traversing - Ancestors

- An ancestor is a parent, grandparent, great-grandparent, and so on.
- jQuery methods for traversing up the DOM tree are:
  - parent() - returns the direct parent element of the selected element
  - parents() - returns all ancestor elements of the selected element (up to document root)
  - parentsUntil() - returns all ancestor elements between two given arguments

Traversing - Ancestors

```
$("span").parent(); //returns the direct parent element of each <span> elements

$("span").parents();   //returns all ancestors of all <span> elements

$("span").parents("ul");   //returns all ancestors of all <span> elements
                           that are <ul> elements

$("span").parentsUntil("div");   //returns all ancestor elements between a
                                 <span> and a <div> element
```

Traversing - Descendants

- A descendant is a child, grandchild, great-grandchild, and so on.
- jQuery methods for traversing down the DOM tree are:
  - children() - returns all direct children of the selected element. This method only traverse a single level down the DOM tree
  - find() - returns descendant elements of the selected element, all the way down to the last descendant.

Traversing - Descendants

```
$("div").children();
```

//returns all elements that are direct children of each elements

```
$("div").children("p.first");
```

**//returns all <p> elements with the class name "first", that are direct children of **

```
$("div").find("span");
```

**//returns all elements that are descendants of **

```
$("div").find("*");
```

**//returns all descendants of **

Traversing - Siblings

- Siblings share the same parent.

- jQuery methods for traversing sideways in the DOM tree are:

  - siblings() - returns all sibling elements of the selected element
  - next() - returns the next sibling element
  - nextAll() - returns all next sibling elements
  - nextUntil() - returns all next sibling elements between two given arguments
  - prev() - returns the previous sibling element
  - prevAll() - returns all previous sibling elements
  - prevUntil() - returns all previous sibling elements between two given arguments

Traversing - Siblings

```
$("h2").siblings();
```

//returns all sibling elements of <h2>

```
$("h2").siblings("p");
```

//returns all sibling elements of <h2> that are <p> elements

```
$("h2").next();
```

//returns the next sibling of <h2>

```
$("h2").nextAll();
```

//returns all next sibling elements of <h2>

```
$("h2").nextUntil("h6");
```

//returns all sibling elements between a <h2> and a <h6> element

Traversing - Filtering

$("div p").**first**();	**selects the first <p> element inside the first element**
$("div p").**last**();	**selects the last <p> element inside the last element**
$("p").**eq**(1);	**selects the second <p> element**

The index numbers start at 0, so the first element will have the index number 0 and not 1

$("p").**filter**(".intro");	**returns all <p> elements with class name "intro"**
$("p").**not**(".intro");	**returns all <p> elements that do not have class name "intro"**

AJAX

AJAX

- AJAX (Asynchronous JavaScript and XML) is exchanging data with a server, and updating parts of a web page - without reloading the whole page.

- jQuery provides several methods for AJAX functionality.

  - Load()
  - Get ()
  - Post()

AJAX – load()

- load() method loads data from a server and puts the returned data into the selected element

$(*selector*).load(*URL , data , callback*);

  - The required URL parameter specifies the URL you wish to load.
  - The optional data parameter specifies a set of query string key/value pairs to send along with the request.
  - The optional callback parameter is the name of a function to be executed after the load() method is completed.

AJAX – load()

```
$("#div1").load("demo_test.html");
```

loads the content of the file "demo_test.html" into a specific element

```
$("#div1").load("demo_test.html #p1");
```

loads the content of the element with id="p1", inside the file "demo_test.html", into a specific element

AJAX – load()

- Callback parameter specifies a callback function to run when the load() method is completed.

- The callback function can have different parameters:

  - **responseTxt** - contains the resulting content if the call succeeds
  - **statusTxt** - contains the status of the call
  - **xhr** - contains the XMLHttpRequest object

```
$("#div1").load("demo_test.html", function(responseTxt, statusTxt, xhr){

    if(statusTxt == "success")
        alert("External content loaded successfully!");
    if(statusTxt == "error")
        alert("Error: " + xhr.status + ": " + xhr.statusText);
  });
```

HTTP Request: GET vs. POST

- GET is basically used for just getting (retrieving) some data from the server. The GET method may return cached data.

- POST can also be used to get some data from the server. However, the POST method NEVER caches data, and is often used to send data along with the request.

- jQuery get() and post() methods are used to request data from the server with an HTTP GET or POST request.

AJAX – get()

- $.get() method requests data from the server with an HTTP GET request

$.get(*URL,callback*);

  - The required URL parameter specifies the URL you wish to request.
  - The optional callback parameter is the name of a function to be executed if the request succeeds.

```
$.get("/rest/demo/getdata", function(data, status){
      alert("Data: " + data + "\nStatus: " + status);
 });
```

AJAX – post()

- $.post() method requests data from the server using an HTTP POST request

$.post(*URL, data, callback*)

  - The required URL parameter specifies the URL you wish to request.
  - The optional data parameter specifies some data to send along with the request.
  - The optional callback parameter is the name of a function to be executed if the request succeeds.

AJAX – post()

```
$.post(

  “/rest/demo/getpostdata",        // URL

  {
      name: “Servion",
      city: “Chennai"                          // Data
   },

   function(data, status){
      alert("Data: " + data + "\nStatus: " + status);        // Callback Function
   }

);
```

EFFECTS

Effects – hide() and show()

- hide() and show() methods are used to hide and show HTML elements.

$(*selector*).hide(*speed,callback*);

$(*selector*).show(*speed,callback*);

  - The optional speed parameter specifies the speed of the hiding/showing, and can take the following values: "slow", "fast", or milliseconds.
  - The optional callback parameter is a function to be executed after the hide() or show() method completes.

```
$("p").hide(1000);
```

```
$("p").show("slow");
```

Effects – toggle()

- toggle() is used to toggle between hide and show methods.

$(*selector*).toggle(*speed,callback*);

  - The optional speed parameter specifies the speed of the hiding/showing, and can take the following values: "slow", "fast", or milliseconds.
  - The optional callback parameter is a function to be executed after the hide() or show() method completes.

```
$("p").toggle();
```

```
$("p").toggle("slow");
```

Effects – Fade

- Following methods are used to fade in and out an element
  - $(selector).**fadeIn**(speed,callback)
  - $(selector).**fadeOut**(speed,callback)
  - $(selector).**fadeToggle**(speed,callback)
  - $(selector).**fadeTo**(speed,opacity,callback)
- Speed and opacity are required parameters for fadeTo method, while they are optional for others.

Effects – Slide

- Following methods are used for sliding effect on elements
  - $(selector).**slideDown**(speed,callback)
  - $(selector).**slideUp**(speed,callback)
  - $(selector).**slideToggle**(speed,callback)

Effects – Animate

- animate() method is used to create custom animations.

$(*selector*).animate({*params*} , *speed, callback*)

- The required params parameter defines the CSS properties to be animated.

- 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 animation completes.

Effects – Animate

```
$("div").animate({left: '250px'});
```

Moves a element to the right, until it has reached a left property of 250px

```
$("div").animate({
      left: '250px',
      opacity: '0.5',
      height: '150px',
      width: '150px'
   });
```

Multiple properties animated at same time

```
$("div").animate({
      left: '250px',
      height: '+=150px',
      width: '+=150px'
   });
```

Animate with relative values

Effects – Animate Queue

If you write multiple animate() calls after each other, jQuery creates an "internal" queue with these method calls. Then it runs the animate calls ONE by ONE

```
$("button").click(function(){
   var div = $("div");
   div.animate({height: '300px', opacity: '0.4'}, "slow");
   div.animate({width: '300px', opacity: '0.8'}, "slow");
   div.animate({height: '100px', opacity: '0.4'}, "slow");
   div.animate({width: '100px', opacity: '0.8'}, "slow");
});
```

Effects – Stop

- stop() method is used to stop an animation or effect before it is finished.

$(selector).stop(stopAll, goToEnd)

- The optional stopAll parameter specifies whether also the animation queue should be cleared or not. Default is false, which means that only the active animation will be stopped, allowing any queued animations to be performed afterwards.

- The optional goToEnd parameter specifies whether or not to complete the current animation immediately. Default is false.

```
$("#panel").stop()
```

kills the current animation being performed on element with id “panel”

Effects – Chaining

- Chaining allows us to run multiple jQuery methods, on the same element, within a single statement.
- To chain an action, you simply append the action to the previous action.

```
$("#p1").css("color", "red").slideUp(2000).slideDown(2000);
```

Above example chains together the css(), slideUp(), and slideDown() methods.

The "p1" element first changes to red, then it slides up, and then it slides down

REFERENCES

References

- http://www.w3schools.com/jquery
- http://www.jquery.com

UI Frameworks

- https://jqueryui.com
- http://www.jeasyui.com

QUESTIONS

Thank You!

servion

CUSTOMER INTERACTION MANAGEMENT SPECIALISTS

Discover
Customer
expectations

Design
Compelling
interaction strategies

Deliver
The experience
your brand promises

Customer Experience by Design