**jQuery**

**document.ready()**

jQuery(document).ready(function() {

jQuery(’#button1’).click(function() {

alert(’...’);

});

});

... <input type=”button” id=”button1”...

$(document).ready(function() is a jQuery event. It fires on as soon as the DOM is loaded and ready to be manipulated by script. This is the earliest point in the page load process where the script can safly access elements in the page’s html dom. This event is fired before all the images, CSS etc are fully loaded.

$(window).load(function() {

Alert(’Window loaded’);

});

**CDN**

CDN : Content Delivery Network. A CDN is a system of distributed servers that hosts resources as images, CSS, JavaScript files, etc.

Distributed CDN servers: the jQuery file can be downloaded from the CDN server that is closest to the user.

Browser-jQuery caching so no need to redownload.

**jQuery selectors**

Element ID

Element Tag Name

Element Class Name

Element attribute

Element attribute values etc.

$(document).ready(function() {

$(’#button1’).css(’background-color’,’orange’);

//In plain JavaScript it would be: document.getElementById(’button2’).style.backgroundColor = ’turqoise’;

To check if the element is found use : if($(’#button1’).length > 0)

});

JavaScripts document.getElementById() and jQuery(#id) selectors are not the same. Document.getElementById() returns a raw DOM object whereas jQuery(’#id’) selector returns a jQuery object that wraps the DOM object and provides jQuery methods. This is the reason you are able to call jQuery methods like css(), click() on the object returned by jQuery. To get the underlying DOM object from a jQuery object use $(’#id’)[0].

document.getElementById() is fanster than jQuery(’#id’).

**jQuery Element selector**

E.g. $(’td’).length will tell how many tds are in the document.

$(’div a’).css(...) will select all anchor elements that are descendants of div element.

$(’#id tr: even’).css() will access the even rows of the table with id.

**jQuery Class Selector**

$(’.class’).css()... or $(’element.class1’) selects only the element with class1

Alternatively use the filter method $(’.class1’).filter(’.class\_n’).css(...)

**jQuery Attribute (value) Selector**

Syntax: $(’attribute’) or $(’[attribute=”value”]’)

$(’element[title],[style]’) accesses elements with either a title **or** style attribute. The or condition is imposed by the comma.

**jQuery case insensitive attribute selection**

$(’div[title]’).filter(function () {

Return $(this).attr(’tetle’).toLowerCase() == ”divtitle”;

}).css(...);

$(’div[title=\*”div”]) will access the div elements whose title attribute contains the substring div.

Alternatively

$(’div[title]’).filter(function () {

return (/Dies/i).test($(this).attr(’title’)); //Return the element div with a title attribute which

}).css(’border’,’3px solid red’); //ignoring the case contains the Dies substring. $(this)

//comes from $(’div[title]’).

Alternatively

$(’div[title]’).filter(function () {

return RegExp(’Dies’, ’i’).test($(this).attr(’title’));

}).css(’border’,’3px solid red’);

**jQuery input vs :input**

$(:input) selects all input, textarea, select and button elements whereas $(input) just selects elements with an input tag.

$(document).ready(function () {

$(’input’).each(function() {

Alert($(this).val());

});

});

$(’:input[type==”text”]’) needs to scan all input elements, textarea, select etc, whereas $(’input[type=”text”]’) scans only input elements.

**jQuery checked selector**

$(document).ready(function () {

$(’#tbtnSubmit’).click(function () {

var res = $(’input[type=”radio”]:checked’); // Selects all checked checkbox elements

if(res.length>0) {

$(#divResult’).html(res.val() + ” is checked”);

}

else {

$(’#divResult’).html(„No radio button checked”);

}

});

});

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

$(’input[name=”cities”]’).click(function () {

getSelectedBoxes(’cities’);

});

Var getSelectedBoxes = function (groupname) {

Var res= $(’input[name=”’ + groupName + ’”]:checked’);

If(res.length>0) {

Var resultString = res.length +” checkboxe(s) checked<br />;

res.each(function() {

var selectedValue = $(this).val();

resultString += selectedValue + ” – ” + $(’label[for=”cb-’ + selectedValue + ’”]’).text() + ”<br/>”;

});

$(’#div’ + groupName).html(resultString);

}

Else {

$(’#div’ + groupName).html(„No checkbox checked”);

}

);

});

**jQuery :selected selector**

To select all checked checkboxes or radio buttons, we use :checked selector. To select all selected options of a select element use :selected selector.

The following retrieves selected options from a multiple select element (<select multiple=”multiple”>).

$(document).ready(function () {

$(’selectCountries’).change(function () {

Var selectedOption = $(’#selectCountries option:selected’);

If (selectedOptions.length > 0) {

Var resultString = ’ ’;

selectedOptions.each(function () {

resultString += ”Value = ” + $(this).val() + ”, Text = ” + $(this).text();

});

$(’#divResult’).html(’Value = ’ + selectedOption.val() +”, Text = ” + selectedOption.text());

}

});

});

**jQuery disabled/enabled selector**

<input type =”text” disabled=”disabled” />

$(document).ready(funtion () {

$(’input[type=”text”]:disabled’).css(’border’, ’3px solid red’);

// or $(’:enabled’).css(...); selects all enabled elements

});

To read more about jQuery selectors visit api.jquery.com/button-selector/

**jQuery each() function**

<script type=”text/javascript”>

$(document).ready(function () {

// Technique of caching an element.

var result = ’ ’;

$(’li’).each(function () {

result += ’<br/>’ + $(this).text();

});

$(’#divResult’).html(result);

/\* If ($(this).text() == ’Gabon’) {

return false; /\*When this executes it stops accessing elements li and exits the each() \*/

}

alert($(this).text());

}); \*/

});

</script>

</heaad>

<body style=”font-family:Arial”>

<ul>

<li>Mauritania</li>

<li>Senegal</li>

<li>Gabon</li>

<li>Central African Rep</li>

</ul>

<div id=”divResult”></div>

</body>

</html>

<script type=”text/javascript”>

$(document).ready(function () {

$(’li’).css(’color’, ’red’);

});

</script>

**Implicit iteration** in jQuery: The $(’li’) selector returns all list item elements. Notice that we rae calling the css() jquery function on the jquery collection itself. In this case, css() method implicitly iterates over each element in the entire collection.

**jQuery method chaining**

<script src=”jquery-3.5.1.js”></script>

<script type=”text/javascript”>

$(document).ready(function () {

$(’li’).css(’color’, ’blue’)

.slideUp(2000)

.slideDown(2000)

.attr(’title’, ’Diese Aussage!’);

});

</script>

</head>

<body style=”font-family:Arial”>

<ul>

<li>Mauritania</li>

<li>Senegal</li>

<li>Gabon</li>

<li>Central African Rep</li>

</ul>

</body>

</html>

Method chaining will not work when one of the methods in the sequence will not return an object.

**JSON- JavaScript Object Notation**

JSON arrays creation

Wrap the objects in square brackets. Each object must be separated with a comma.

var objJSON = [{

”ersteName”: ”Uwe”,

”NachName”: ”Grolsch”

},

{

”ersteName”: ”Ludwig”,

”NachName”: „”Gunther”

}];

Accessing them: objJSON[0].NachName

It is possible to nest many objects in the JSON object

var Obj\_JSON = {

”A”: {

”ersteName”: ...

},

”B”: {...

}, ...

};

**Conversion of a JSON object to string and vice versa**

//From the code above

let jsonString = JSON.stringify(objJSON);

$(’divResult’).html(result);

// Converting the JSON string to JSON array. JSON parse() method converts a JSON string to JSON array.

let employee = JSON.parse(jsonString);

let result = ’ ’;

$.each(employee, function (i, item) {

result += ’ersteName = ’ + ite.ersteName + ’<br/>’;

result += ’NachName = ’ + ite.NachName + ’<br/>’;

});

**jQuery DOM Manipulation Methods**

attr() is used to set or get the attribute value of html elements.

$(’document).ready(function () {

$(’div’).each(function () {

alert($(this).attr(’title’));

})

//Setting the attribute of title

/\* $(’div’).attr(’title’, ’New DIV Title’);

$(’div’).each(function () {

alert($(this).attr(’title’));

}) \*/

});

**jQuery wrap elements**

wrap: wraps an HTML structure around each element in the set of matched elements

unwrap: remove the parents of the set of matched elements from the DOM

wrapAll: wrap an HTML structure around all elements in the set of matched elements.

.containerDiv {

background-color: green;

color: white;

font-weight: bold;

margin: 3px;

}

</style>

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

$(’div’).wrap(’<div class=”containerDiv”></div>’);

});

wrapInner: wraps an HTML structure around the content of each element in the set of matched elements.

**jQuery append elements**

append() or appendTo()

prepend() or prependTo()

$(document).ready(function () {

$(’div’).append(’<b> Text </b>’);

//Alternatively $(’div’).append(’<b> Text </b>’); The argument in prepend() will be

//placed before the content of the div in $() constructor.

});

Prepend() inserts the specified content to the beginning of each element in the set of matched elements, whereas append() inserts the specified content to the end of each element in the set of matched elements.

$(document).ready(function () {

$(’div’).append($(’#thisspan’));

});

...

<body>

<span id= ”thisspan” > Text </span>

<div>...

<div>...

<div>...

</body> </html>

**jQuery insert before and after Tutorial 24**

$(document).ready(function () {

//The two following give the same result

$(’div’).before(’<h3>Text></h3>’);

// Alternatively $(’<h3>Text></h3>’).insertBefore($(’div’);

//Using after

$(’div’).after(’<h3>Text></h3>’);

// Alternatively $(’<h3>Text></h3>’).insertAfter($(’div’);

/\*Using after or before with an existing HTML element-see below HTML code.

$(’div’).after($(’span’); \*/

});

...

<body>

<span id= ”thisspan” > Text </span>

<div>...

<div>...

<div>...

</body> </html>

**Add/remove class Tutorial 25**

hasClass: Returns true if an element has the specified class othw false

addClass: Adds one or more specified classes. To add multiple classes separate them with a space.

removeClass: Removes one or multiple or all classes. To remove multiple classes separate them with a space. To remove all classes, don’t specify any class name.

toggleClass: Toggles one or more specified classes. If the element has the specified class then it is removed, if the class is not present then it is added.

<style>

.boldClass {

font-weight: bold;

}

.italicsClass {

font-style: italic;

}

.textBoxStyle

{

Background-color: green;

Color: white;

}

</style>

<script type=”text/javascript”>

$(document).ready(function () {

$(’input[type=”text”]’).each(function () {

$(this).focus(function () {

$(this).addClass(’textBoxStyle’);

});

$(this).blur(function () {

$(this).removeClass(’textBoxStyle’);

})

})

});

</head>

...

...<table>

<tr>

<td>

<input id=”btn1” style=”width250px” type=”button”

Value=”Add Color and Italics Classes” />

</td>

<td>

<input id=”btn2” style=”width250px” type=”button”

Value=”Remove Color and Italics Classes” />

</td>

<td>

<input id=”btn7” style=”width250px” type=”button”

Value=”Toggle Color and Italics Classes” />

</td>

...

$(document).ready(function () {

$(’#btn1’).click(function () {

$(’p’).addClass(’colorClass italicsClass’);

});

$(’#btn2’).click(function () {

$(’p’).removeClass(’colorClass italicsClass’);

});

$(’#btn7’).click(function () {

$(’p’).toggleClass(’colorClass italicsClass’);

});

});

**Difference between $.each and .each**

Two versions .each() and $.each or jQuery.each()

.each() is used to iterate over the items in a jQuery collection while $.each() is used to iterate over javascript objects or arrays.

$(’document).ready(function () {

Let result =’’;

$(’li’).each(function (index, element) {

result += ’Index = ’ + index + ’, Value = ’+$(this).text() +’<br/>;

//$(this) is the element argument of the anonymous function.

//Likewise one can loop through a JS array or JS(ON) object

let jsObject = {

’Senegal: ’Dakar’,

’Gabon’:’Libreville’,

’Madagascar’:’Antananarivo’,

’Mauritius’: ’Port Louis’,

’Democratic Republic of Congo’: ’Kinshasa’,

’Central African Republic’: ’Bangui’

};

Let result = ’ ’;

$.each(jsObject, function(key, element) {

result += ’Key = ’ + key + ’, Value = ’ + element + ’<br/>’;

});

$(’#resultDiv’).html(result);

});

...

<body style=”font-family:Arial”>

<ul>

<li>Mauritania</li>

<li>Senegal</li>

<li>Gabon</li>

<li>Mauritius</li>

<li>Madagascar</li>

<li>Central African Rep</li>

</ul>

<div id=”resultDiv”></div>

**map method**

map() is used to iterate over matched elements just like each(). In general if you want to create an array or concatenated string based on all matched elements in a jQuery selector, it is better to use map() over each().

Creation of array list elements with each

<head>...

<script src= ”jquery-3.5.1.js”></script>

<script type=”text/javascript”>

$(document).ready(function () {

var resu = [];

//each() method\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

$(’li’).each(function (index, element) {

Resu.push($(element).text());

});

alert(resu);

//map() method \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// No array resu[] creation here!

alert($(’li’).map(function (index, element) {

return $(element).text();

}).get()); //get() creates a js array

});

//each method\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

var resu =’’;

$(’li’).each(function (index, element) {

Resu += $(element).text() + ’|’;

});

resu = resu.substr(0, resu.length – 1);

alert(resu);

});

//map method\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

alert($(’li’).map(function (index, element) {

return $(element).text();

}).get().join(’|’));

});

</script>

<body style=”font-family:Arial”>

<ul>

<li>Mauritania</li>

<li>Senegal</li>

<li>Gabon</li>

<li>Central African Rep</li>

</ul>

...

$.map() $.each()

It can be used as an iterator. It is an immutable iterator.

Returns a new array. Returns the original array.

The order of callback arguments-element, index The order of callback arguments- index, element

$.map(elems, function() {element, index), arg) $.each(elems, function() { index, element}, arg)

Does not have a way to terminate the iteration. Return false to terminate the iteration.

**Change event**

Change event is fired when an element value changes. All the following elements fire this event.

1. Input
2. Textarea
3. Select

Select, radio buttons and checkboxes fire the change event as soon as a selection is made, whereas the other element types wait until they loose focus.

<script src= ”jquery-3.5.1.js”></script>

<script type=”text/javascript”>

$(document).ready(function () {

$(’#idelement’).change(function () {

var selValue = $(this).val();

$(’#divRes’).html(selValue);

});

});

//Alternatively\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

$(document).ready(function () {

var resu = ’’;

$(’input’).change(function () {

If(resu == ’’) {

Resu = $(this).val();

}

else {

resu += ’, ’ +$(this).val();

}

$(’#divResult’).html(resu);

});

});

**Tutorial 30 Tutorial 30 Tutorial 30**

**Mouse events**

<script src=”jquery-3.5.1.js”></script>

<script type = „text/javascript”>

$(document).ready(function () {

$(’#img[src=”help.png”]’).mouseover(function () {

$(’#’ +getDivId(this).fadeIn(500);

}).mouseout(function () {

$(’#’ + getDivId(this)).fadeOut(500);

});

function getDivId(helpIcon) {

var helpInconId = $(helpIcon).attr(’id’);

return helpInconId.replace(’img’, ’div’);

}

});

</script>

<body>

<table>

<tr>

<td>First Name</td>

<input id=”txtFirstName” type=”text” class =”inputRequired” />

</td>

<td><img id=”imgFirstNameHelp” src=”help.png” /></td>

<td><div id=”divFirstNameHelp” style =”display:none”>Passport first name </td>

</tr>

<tr>

<td>Last Name</td>

<td>

<input id=”txtLastName” type=”text” class=”inputRequired” />

</td>

<td><img id=”imgLastName1p” src=”help.png” /></td>

<td><div id=”divLastNameHelp” style=”display:none”> Last Name in passport</td>

</tr>

<tr>

<td>City</td>

<td>

<select id=”ddlCity” class=”inputRequired”>

<option value=”Select”>Selectionez</option>

<option value=”Jakarta”>Jakarta</option>

<option value=”Taipei”>Taipei</option>

<option value=”Hong Kong”>Hong Kong</option>

</select>

</td>

</tr>

</table>

</body>

</html>

**jQuery event object**

<html>

<head>

<title></title>

<script src=”jquery-3.5.1.js”></script>

<script type=”text/javascript”>

$(document).ready(function () {

$(’#btn’).click(function (event) {

getEventDetails(event);

}).mouseover(function (event) {

getEventDetails(event);

}).mouseout(function (event) {

getEventDetails(event);

});

//**The event object is a jQuery object! Allows multi-browser compatibility.**

function getEventDetails(event) {

//**Alternatively $.event.fix(event); will convert the JS object to jQuery object**

var eventDetails = ’Event = ’ + event.type + ’<br/>’ +

’X = ’ + event.pageX + ’<br/>’ +

’Y’ = ’ + event.pageY + ’<br/> +

’Target Element = ’ + event.target.type + ’<br/>’ +

’Target Element Tag Name = ’ + event.target.tagName;

$(’#divResult’).html(eventDetails);

}

});

</script>

</head>

<body style=”font-family:Arial”>

<input id=”btn” type=”button” value=”Klikker” />

<br /><br />

<div id=”divResult”></div>

</body>

</html>

**jQuery mouse button detection**

With raw JS event object (depending on the browser) event.button or event.which properties of the event object are used to determine which mouse button is clicked.

<html>

<head>

<title></title>

<script src=”jquery-3.5.1.js”></script>

<script type=”text/javascript”>

$(document).ready(function () {

$(’#btn’).mouseup(function (event) {

var whichButton = ’’ ;

switch (event.which) {

case 1: whichButton = ”Left Button Clicked”; break;

case2: whichButton = ”Middle Button Clicked”; break;

case 3: whichButton = ”Right Button Clicked”; break;

default: whichButton = ”Invalid Button Clicked”; break;

}

$(’#divResult’).html(whichButton);

});

});

</script>

</head>

<body style=”font-family:Arial”>

<input id=”btn” type=”button” value=”Klikker” />

<br /><br />

<div id=”divResult”></div>

</body>

</html>

**Binding event handlers**

**The bind() method is a way to bind event handlers to events.**

<html>

<head>

<title></title>

<style>

.ButtonStyle {

background-color: red;

font-weight: bold;

color: white;

cursor: pointer;

}

</style>

<script src=”jquery-3.5.1.js”></script>

<script type=”text/javascript”>

$(document).ready(function () {

$(’#btnClickHere’).bind(’click mouseover mouseout’, function (event) {

If (event.type == ’click’) {

$(’divResult’).html(’Button Clicked at X = ’ + event.pageX + ’ Y = ’ + event.pageY);

}

else if(event.type ==’mouseover’) {

$(this).addClass(’ButtonStyle’);

}

else {

$(this).removeClass(’ButtonStyle’);

}

});

});

</script>

</head>

<body style=”font-family:Arial”>

<input id=”btnClickHere” type=”button” value=”Klikker” />

<input id=”btnEnableMouseOverEffect” type=”button” value=”EnMOvEff” />

<input id=”btnDisableMouseOverEffect” type=”button” value=”DisMOvEff” />

<br /><br />

<div id=”divResult”></div>

</body>

</html>

To unbind the event handlers one may use the unbind() method.

$(’#btnClickHere’).unbind(’mouseover’);

To unbind all the event handlers of an element, use the unbind() method without any parameters.

$(’#btnClickHere’).unbind();

For jQuery 1.7 or higher the bind() and unbind() methods have been substituted by the on() and off() methods.

**Binding event handlers using the on() method**

<html>

<head>

<title></title>

<style>

.ButtonStyle {

background-color: red;

font-weight: bold;

color: white;

cursor: pointer;

}

</style>

<script src=”jquery-1.11.2.js”></script>

<script type=”text/javascript”>

$(document).ready(function () {

$(’#btnClickHere’).on({ // **This method uses the json model.**

click: function (event) {

$(’#divResult’).html(’Button Clicked at X = ’ + event.pageX + ’ Y = ’ + event.pageY);

},

mouseover: function () {

$(this).addClass(’ButtonStyle’);

},

mouseout: function () {

$(this).removeClass(’ButtonStyle’);

}

});

/\*Alternatively \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

$(’#btnEnableMouseOverEffect’).click(function () {

$(’#btnClickHere’).on(’mousover’, function () {

$(this).addClass(’ButtonStyle’);

});

});

$(’#btnDisableMouseOverEffect’).click(function () {

$(’#btnClickHere’).off(’mousover’, function () {

$(this).addClass(’ButtonStyle’);

});

});

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \*/

</script>

</head>

<body style=”font-family:Arial”>

<input id=”btnClickHere” type=”button” value=”Klikker” />

<input id=”btnEnableMouseOverEffect” type=”button” value=”EnMOvEff” />

<input id=”btnDisableMouseOverEffect” type=”button” value=”DisMOvEff” />

<br /><br />

<div id=”divResult”></div>

</body>

</html>

**Passing data to the event handler**

<html>

<head>

<script src=”jquery-3.5.1.js”></script>

<script type=”text/javascript”>

$(document).ready(function () {

$('#btnClickHere').on('click', {

firstName:'Lin'

lastName:'Xue'

}, Greet);

$('#btnClickHere').on('click', {

firstName:'Jing'

}, Greet);

function Greet() {

if (event.data == null){

alert('No data entered.');

}

else {

alert('Greeting' + event.data.firstName + ' ' + (event.data.lastName != null ? ' ' + event.data.lastName: ''));

}

}

});

</script>

</head>

<body style = "font-family: Arial">

<input id = "btnClickHere" type = "button" value = "Kliker" />

</body>

</html>

**Adding event handlers to dynamically created elements**

Dynamically create a new list element <li>, attach a click event handler and add it to the unordered list (ul). The problem with this approach is that one binds a click event handler to every list item. This means if there are 500 list items, there will be 500 list event handlers in the memory, which definitely deteriorates the system performance.

An optimal way is to attach the click event handler to the list element (li) to the parent element (ul). In this case if there are 500 list items, there is only one click event handler in the memory.

<html>

<head>

<script src=”jquery-3.5.1.js”></script>

<script type=”text/javascript”>

$(document).ready(function () {

$('li').on('click', 'li', function () {

$(this).fadeOut(500);

});

$('#btnAdd').on('click', function () {

/\* The following is used if there is no the selector ’li’ in the on method arguments above. A new click event handler is created per item. Inferior performance.

var newListItem = $('<li>Dynamically new item </li>').on('click', function () {

$(this).fadeOut(500);

$('ul').append(newListItem);

\*/

$('ul').append('<li>Dynamically new item </li>');

});

});

</script>

</head>

<body style = "font-family: Arial">

<input id = "btnAdd" type = "button" value = "New list element" />

<ul>

<li>Item 11 </li>

<li>Item 22 </li>

</ul>

</body>

</html>

**jQuery event delegation**

Event delegation allows to attach a single event listener to a parent element that will fire for all descendants matching a selector, whether those descendants exist now or are added in the future. Both on() and delegate() methods allow one to perform event delegation.

<html>

<head>

<script src=”jquery-3.5.1.js”></script>

<script type=”text/javascript”>

$(document).ready(function () {

$('li').delegate('click', 'li', function () {

$(this).fadeOut(500);

});

$('#btnAdd').on('click', function () {

$('ul').append('<li>Dynamically new item </li>');

});

$('#btnUndelegate').click(function () {

$('ul').off('click','li');

/\*Alternatively use undelegate()

$('ul').undelegate('li',click'); \*/

});

});

</script>

</head>

<body style = "font-family: Arial">

<input id = "btnAdd" type = "button" value = "New list element" />

<input id = "btnUndelegate" type = "button" value = "Undelegate" />

<ul>

<li>Item 11 </li>

<li>Item 22 </li>

</ul>

</body>

</html>

**Event delegation by live() and die() functions (up to jQuery 1.7v)**

With the live() function the event gets bubbled up all the way to the document object whereas with the on() and delegate() functions the event gets bubbled up to the specified parent element.

<html>

<head>

<script src="http://ajax.aspnetcdn.com/ajax/jquery/-1.7.js"></script>

<script type=”text/javascript”>

$(document).ready(function () {

$('li').live('click', 'li', function () {

$(this).fadeOut(500);

});

$('#btnAdd').on('click', function () {

$('ul').append('<li>Dynamically new item </li>');

});

$('#btnUndelegate').click(function () {

$('ul').die('click');

});

});

</script>

</head>

<body style = "font-family: Arial">

<input id = "btnAdd" type = "button" value = "New list element" />

<input id = "btnUndelegate" type = "button" value = "Undelegate" />

<ul>

<li>Item 11 </li>

<li>Item 22 </li>

</ul>

</body>

</html>

**jQuery event execution only once-use of one() method**

The jQuery one() function registers an event handler to an event for one and only time. The click event is raised only once. On the first click the alert statement executes and for subsequent clicks the alert statements cease to execute.

<html>

<head>

<script src="http://ajax.aspnetcdn.com/ajax/jquery/-1.7.js"></script>

<script type=”text/javascript”>

$(document).ready(function () {

$('input[type="button"]').one({

mouseover:function () {

$(this).css('background-color','orange');

},

mouseout:function () {

$(this).css('background-color','white');

},

click: function () {

alert('Button clicked!')

}

});

});

</script>

</head>

<body style = "font-family: Arial">

<input type = "button" value = "Kliker" />

<input type = "button" value = "Kliker" />

<input type = "button" value = "Kliker" />

</body>

</html>

**Checking if an event is bound**

<html>

<head>

<script src=”jquery-3.5.1.js”></script>

<script type=”text/javascript”>

$(document).ready(function () {

$('#btn').on({

click: function () {

alert('Button clicked');

},

mouseover: function() {

$(this).css('background-color', 'yellow');

}

mouseout: function() {

$(this).css('background-color', 'white');

}

});

let jqueryObject = $('#btn');

let rawDomElement = jqueryObject.get(0);

let eventObject = $.\_data(rawDomElement, 'events');

if(eventObject != undefined && eventObject.click != undefined) {

alert('Click event already exists');

}

else {

alert('Click event does not exist');

}

/\*Alternatively to attach a unique event handler one may use the on() and off()

methods. The off() method ensures that all existing click event handlers of

the button are removed before again adding a new click event handler using the on()

method.

$('#btn').on('click', function () { alert('Button clicked')});

$('#btn').off('click').on('click', function () { alert('Button clicked')});

\*/

});

</script>

</head>

<body style = "font-family: Arial">

<input id = "btn" type = "button" value = "Click here." />

</body>

</html>

**jQuery preventdefault**

<html>

<head>

<script src=”jquery-3.5.1.js”></script>

<script type=”text/javascript”>

$(document).ready(function () {

$(this).on(’contextmenu’, function (e) {

e.preventDefault();

$(’#divResult’).append(’Right Click Disabled<br/>’);

});

$(’#HyperLink’).on(’click’, function (e) {

e.preventDefault();

$(’#divResult’)

.append(„Hyper link default action disabled <br/>’)

});

});

</script>

</head>

<body style=”font-family: Arial”>

<h3>

Right click is disabled on this page. Try to right click

And see what happens.

</h3>

<a id=”HyperLink” href=<http://todorimasu.com>>

Hyperlink disabled.

</a>

<div id=”divResult”></div>

</body>

</html>

**Scroll event**

<html>

<head>

<script src=”jquery-3.5.1.js”></script>

<script type=”text/javascript”>

$(document).ready(function () {

let count = 0;

//let lastScrollTop = 0;

$(’div’).scroll(function () {

//let currentScrollPos = $(this).scrollTop();

$(’span’).text(count+=1);

//$(’h3’).css(’display’,’inline’).fadeOUt(1000);

//$(’h3’).text(currentScrollPosition);

});

});

</script>

</head>

<body style=”font-family: Arial”>

<div style=”height:200px; width:300px; border:1px solid black; overflow:scroll’>

Txt....

</div>

<br />

<h3 style=”display:none; color: green”> Scroll event handled<span>0</span> time(s)</h3>

</body>

</html>

**Image Gallery**

<html>

<head>

<script src=”jquery-3.5.1.js”></script>

<style>

.imgStyle {

height: 100px;

width: 100px;

border: 3px solid grey;

}

</style>

<script type=”text/javascript”>

$(document).ready(function () {

$(’#divContainer img’).on({

mouseover: function () {

$(this).css({

’cursor’: ’pointer’,

’border-Color’: ’red’

});

},

mouseout: function () {

$(this).css({

’cursor’: ’default’,

’border-Color’: ’red’

});

},

click: function () {

let imageUrl = $(this).attr(’src’);

$(’#mainImage’).fadeOut(600, function () {

$(this).attr(’src’, imageUrl);

}).fadeIn(600);

}

});

});

</script>

</head>

<body >

<img id=”mainImage” height=”500” width=”540” style=”border: 3px solid grey”

Src=”Images/Hydragenas.jpg” />

<br />

<div id=”divContainer”>

<img src=”Images/Hydragenas.jpg” class=”imgStyle” />

<img class=”imgStyle” src=”/Images/Jellyfish.jpg” />

<img class=”imgStyle” src=”/Images/Koala.jpg” />

<img class=”imgStyle” src=”/Images/Penguins.jpg” />

<img class=”imgStyle” src=”/Images/Tulips.jpg” />

</div>

</body>

</html>

<html>

<head>

<script src=”jquery-3.5.1.js”></script>

<style>

.imgStyle {

height: 100px;

width: 100px;

border: 3px solid grey;

}

</style>

**jQuery Image gallery optimized**

<script type=”text/javascript”>

$(document).ready(function () {

$('#divContainer img').on({

mouseover: function () {

$(this).css({

’cursor’: ’pointer’,

’border-Color’: ’red’

});

},

click: function () {

let imageURL = $(this).attr('src');

let effect = $('#selectImgEffect').val();

let duration = $('#selectImgDuration').val()\*1000;

let mainImg = $('#mainImage');

if (effect == 'Fade')

{

$('#mainImage').fadeOut(duration, function () {

$(this).attr('src', imageURL);

}).fadeIn(duration);

}

else {

$('#mainImage').slideUp(duration, function () {

$(this).attr('src', imageURL);

}).slideDown(500);

}

}

}, 'img');

let mainImgElement = $('#mainImage');

let height = parseInt(mainImgElement.attr('height'));

let width = parseInt(mainImgElement.attr('width'));

$('#btnEnlarge').click(function () {

height += 100;

width += 100;

mainImgElement.animate({

'height': height,

'width' : width

});

});

$('#btnShrink').click(function () {

height -= 100;

width -= 100;

mainImgElement.animate({

'height': height,

'width' : width

});

});

});

</script>

</head>

<body >

Select Effect : <select id = "selectImgEffect">

<option value= "Fade">Fade</option>

<option value= "Slide">Fade</option>

</select>

Time in seconds : <select id= "selectImgDuration">

<option value= "0.5">0.5</option>

<option value= "1">1</option>

<option value= "2">2</option>

<option value= "3">3</option>

<option value= "4">4</option>

<option value= "5">5</option>

</select>

<input type="button" id="btnEnlarge" value="Enlarge" />

<input type="button" id="btnShrink" value="Shrink" />

<br />

<img id="mainImage" height="500" width="540" style="border: 3px solid grey"

src="/Images/Hydragenas.jpg" />

<br />

<div id="divContainer">

<img src="Images/Hydragenas.jpg" class="imgStyle" />

<img class="imgStyle" src="/Images/Jellyfish.jpg" />

<img class="imgStyle" src="/Images/Koala.jpg" />

<img class=”imgStyle” src=”/Images/Penguins.jpg” />

<img class=”imgStyle” src=”/Images/Tulips.jpg” />

</div>

</body>

</html>

**Image Slideshow with Thumbnails**

<html>

<head>

<script src=”jquery-3.5.1.js”></script>

<style>

.imgStyle {

height: 100px;

width: 100px;

border: 3px solid grey;

}

</style>

<script type=”text/javascript”>

$(document).ready(function () {

let imageURLs = new Array();

let intervalId;

let btnStart = $('#btnStartSlideShow');

let btnStop = $('#btnStopSlideShow');

$('#divContainer img').each(function () {

imagesURLs.push

});

function setImage() {

let mainImageElement = $('#mainImage');

let currentImageURL = mainImageElement.attr('src');

let currentImageIndex = $.inArray(currentImageURL, imagesURLs);

if(currentImageIndex == (imagesURLs.length -1))

{

currentImageIndex = -1;

}

mainImageElement.attr('src', imagesURLs[currentImageIndex + 1])

}

btnStart.click(function () {

intervalId = setInterval(setImage, 500);

$(this).attr('disabled', 'disabled');

btnStop.removeAttr('disabled');

});

btnStop.click(function () {

clearInterval(intervalId);

$(this).attr('disabled', 'disabled');

btnStart.removeAttr('disabled');

}).attr('disabled', 'disabled');

});

</script>

</head>

<body >

<input type="button" id="btnStartSlideShow" value="Start Slideshow" />

<input type="button" id="btnStopSlideShow" value="Stop Slideshow" />

<br />

<img id="mainImage" height="500" width="540" style="border: 3px solid grey"

src="/Images/Hydragenas.jpg" />

<br />

<div id="divContainer">

<img src="Images/Hydragenas.jpg" class="imgStyle" />

<img class="imgStyle" src="/Images/Jellyfish.jpg" />

<img class="imgStyle" src="/Images/Koala.jpg" />

<img class=”imgStyle” src=”/Images/Penguins.jpg” />

<img class=”imgStyle” src=”/Images/Tulips.jpg” />

</div>

</body>

</html>

**animate() function**

<html>

<head>

<script src=”jquery-3.5.1.js”></script>

<script type=”text/javascript”>

$(document).ready(function () {

$('#myImg').on({

mouseover: function () {

$(this).animate({

'height': 400,

'width': 400

}, 3000, 'linear', animationComplete);

},

mouseout: function () {

$(this).animate({

'height': 100,

'width': 100

}, 3000);

}

});

function animationComplete() {

alert('Animation Completed');

}

/\*Alternative effect

$('myImg').click(function () {

$(this)

.animate({'left': 300})

.animate({'top': 300})

.animate({'left': 10})

.animate({'top': 10})

});

}); \*/

});

</script>

</head>

<body >

<img id="myImg" height="100" width="100" src="Images/Tulips.jpg" />

</body>

</html>

**jQuery animation queue()**

<html>

<head>

<script src=”jquery-3.5.1.js”></script>

<script type=”text/javascript”>

$(document).ready(function () {

$('#myButton').click(function () {

$.fx.off = $('#chkbox').is(':checked'); //will turn off the animation if the chkbox is checked

$('#myDiv1')

.animate({'width': 500}, 1500)

.queue(function () {

console.log('Queued calls =' + $(this).queue('fx').length);

$(this).dequeue();

})

.animate({'padding': 20}, 1500)

.animate({'font-size': 50}, 1500)

.animate({'border-width': 10}, 1500)

.animate({'opacity': 1}, 1500)

.queue(function () {

console.log('Queued calls =' + $(this).queue('fx').length);

$(this).dequeue();

});

$('#myDiv2')

.animate({'width': 500}, 1500)

.animate({'padding': 20}, 1500)

.animate({'font-size': 50}, 1500)

.animate({'border-width': 10}, 1500)

.animate({'opacity': 1}, 1500);

});

});

</script>

</head>

<body >

<input type="button" id="myButton" value="Start Animation" />

<input id="chkbox" type="checkbox" /> Turn off animations

<br /><br />

<div id="myDiv1" class="myDivClass">

jQuery Animations

</div>

<br />

<div id="myDiv2" class="myDivClass">

Second frame

</div>

</body>

</html>

To globally disable animations use $.fx.off = true or jQuery.fx.off = true

For animate() to be executed simultaneously set the queue to false.

$('#myDiv1')

.animate({'width': 500}, {duration: 1500, queue: false})

.animate({'padding': 20}, {duration: 1500, queue: false})

.animate({'font-size': 50}, {duration: 1500, queue: false})

.animate({'border-width': 10}, {duration: 1500, queue: false})

.animate({'opacity': 1}, {duration: 1500, queue: false});

Another way to animate multiple css properties in parallel is to include all those css properties in a single

jSON object.

**jQuery Progress Bar**

<html>

<head>

<script src=”jquery-3.5.1.js”></script>

<script type=”text/javascript”>

$(document).ready(function () {

$('myButton').click(function () {

animateProgressBar($('#ddlPercentage').val());

});

function animateProgressBar(percentageCompleted) {

$('#innerDiv').animate({

'width': (500\*percentageCompleted)/100

},3000);

$({counter:1}).animate({counter:percentageCompleted}, {

duration:3000,

step : function() {

$('#innerDiv').text(Math.ceil(this.counter) +' %');

}

})

}

});

</script>

</head>

<body>

Select Percentage:

<select id="ddlPercentage">

<option value="10">10</option>

<option value="20">20</option>

<option value="30">30</option>

<option value="40">40</option>

<option value="50">50</option>

<option value="60">60</option>

<option value="70">70</option>

<option value="80">80</option>

<option value="90">90</option>

<option value="100">100</option>

</select>

<input type="button" id="myButton" value="Start Animation" />

<br /><br />

<div id="outerDiv" style= " background-color:#EEEEEE; height:20px; width:500px; padding:5px">

<div id="innerDiv" style="background-color:red; height:19px; width:0px; color:white; text-align:center">

</div>

</div>

</body>

</html>

**Tutorial 50 Tutorial 50 Tutorial 50**

**jQuery show/hide password**

<html>

<head>

<script src="jquery-3.5.1.js"></script>

<script type="text/javascript">

$(document).ready(function () {

$(’cbShowPassword’).click(function () {

let currentPasswordField = $('#txtPassword');

let currentPassword = currentPasswordField.val();

currentPasswordField.remove();

if($(this).is(':checked')) {

/\* When the checkbox is checked we insert a

new input field with type text so that the password

is visible.\*/

$(this).before('<input type="text" id="txtPassword" value="' + currentPassword + '"/>');

}

else { /\* When the checkbox is unchecked we insert again

an input field with type password so that the password

will be hidden.\*/

$(this).before('<input type="password" id="txtPassword" value="' + currentPassword + '"/>');

}

});

});

</script>

</head>

<body style="font-family:Arial">

Password :

<input type="password" id="txtPassword" />

<input type="checkbox" id="cbShowPassword" />Show password

</body>

</html>