jQuery

The purpose of jQuery is to make it much easier to use JavaScript on your website.

jQuery Syntax

Basic syntax is: $(*selector*).*action*()

- A $ sign to define/access jQuery
- A (*selector*) to "query (or find)" HTML elements
- A jQuery *action*() to be performed on the element(s)

Examples:

- $(this).hide() → hides the current element.
- $("p").hide() → hides all <p> elements.
- $(".test-jquery").hide() → hides all elements with class=" test-jquery ".
- $("# test-jquery ").hide() → hides the element with id=" test-jquery ".

The Document Ready Event

$(document).ready(function(){	$(function(){
// jQuery methods ...	*// jQuery methods – shorter version*
});	});

This is to prevent any jQuery code from running before the document is finished loading.

jQuery Selectors

jQuery selectors allow you to select and manipulate HTML element(s).

```
// click on p tag
 $('p').click(function(){
     console.log(`clicked on 'p' tag on ${this.innerHTML}`);
     console.log(this.innerHTML);
     $(this).hide()      // hide only current p tag which is clicked
 })
 // click on h3 tag
 $('h3').click(function(){
     console.log("clicked on 'h3' tag");
     $('h3').hide();
 })
 // id selector
 $("#hidden-para").click(function(){
     $('p').hide();
 })
 // class selector
 $('.display-text').click(function(){
     $("h3").css({
         "color": "blue",
     })
 })
```

jQuery HTML / CSS Methods

- **addClass - Adds one or more class names to selected elements**

```
// add class on paragraph one
 $("#add-class").click(function(){
     $('#p1').addClass("new-class");
 })
 // remove class on paragraph one
 $("#remove-class").click(function(){
     $('#p1').removeClass("new-class");
 })
```

- **after - Inserts content after selected elements**

```
$("#after-p2").click(function(){
     $('#p2').after("<p>Paragraph Three</p>");
 })
  // remove the p element
 $("#remove-after-p2").click(function(){
     $('#p2').next('p').remove();
 })
```

- **clone - Makes a copy of selected elements**

```
$("#clone-p2").click(function(){
  // clone the node withDataAndEvents - true
   var newnode = $('#p2').clone(true);
   $('#p2').append(newnode);
})
```

Events of jQuery

- **click()** → executes when user click on 'p' tag

```
$('p').click(function(){
  console.log("click event is occured");
})
```

- **dbclick()** → executes when user double click on 'p' tag

```
$('p').dbclick(function(){
   console.log("double click event is occured");
})
```

- **mouseenter**() → executes when user enter the mouse on perticular tag

```
$('.text').mouseenter(function(){
    console.log(`mouse is entered in ${this.innerHTML}`);
})
```

- **mouseleave()** → executes when user leave the mouse from perticular tag

```
$('.text').mouseleave(function(){
      console.log(`mouse has leaved in ${this.innerHTML}`);
})
```

- **mousedown()** → when press the mouse key

```
$('.text').mousedown (function(){
      console.log(`mouse down in ${this.innerHTML}`);
})
```

- **mouseup()** → when release the mouse key

```
$('.text').mouseup(function(){
      console.log(`mouse up in ${this.innerHTML}`);
})
```

- **submit()** → submit the form

```
$('form').on('submit', function(e){
      e.preventDefault();
      console.log(`form submitted`);
})
```

- **focus()** → focus on the input tag

```
$('#name').on('focus', function(){
      console.log(`focus`);
})
```

- Multiple event handler

```
$('.multi').on({
         mouseenter : function(){
            console.log("enter");
        },
        mouseleave : function(){
            console.log("leave");
        }
    })
```

Validations

// vallidation of **name**

```
    $('#name').blur(function(){
        let name = $('#name').val();

        // validate the name for blank sapce
        if(name.trim()===''){
            $('#nameError').text('*Name is required.');
            $('#name').focus();
        }
        else{
            $('#nameError').text('');
        }
    })
```

// validate the **email**

```
        $('#email').blur(function(){
            const emailPattern =  /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-
Z]{2,4}$/;
            if(!emailPattern.test($(this).val())){
                $('#emailError').text('*Enter valid email.');
                $('#email').focus();
            }
            else{
                $('#emailError').text('');
            }
        })

        // validate the phone number
        $('#phone').blur(function(){
            if($(this).val().length!=10){
                $('#phoneError').text('*Length of phone number must be 10 digits.');
                $('#phone').focus();
            }
            else{
                $('#phoneError').text('');
            }
        })
```

```
// validate the password
$('#pass').blur(function(){
    const passwordPatttern = /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d).{8,}$/;

    // check the lenght of the password
    if($(this).val().length < 8){
        $('#passError').text('*Password should be 8 digits long');
        $('#pass').focus();
    }

    // match with pattern
    else if(!passwordPatttern.test($(this).val())){
        $('#passError').text('*Password must be strong');
        $('#pass').focus();
    }
    else{
        $('#passError').text('');
    }
})

// validate the password to compare with password
$('#cpass').blur(function(){
    let pass = $('#pass').val();
    let cpass = $('#cpass').val();
```

```
            if(pass!==cpass){

                $('#cpassError').text("*Confirm password must be same as
password");

                $('#cpass').focus();

            }

            else{

                $('#cpassError').text('');

            }

        })



        // validate the checkbox -- out of all checkbox need to check alteast 2
checkbox

        $('input[name="checkboxGroup"]').change(function(){

            if ($("input[name='checkboxGroup']:checked").length < 2){

                $('#checkError').text("*Select atleast two checkbox")

                $('#check1').focus();

            }

            else{

                $('#checkError').text("")

            }

        })



        // validate the dropdown

        $('#dropdown').change(function(){
```

```
        let drpdwn = $('#dropdown').val();


        if(drpdwn==0){

            $('#dropError').text("*Please Select any option");

            $('#dropdown').focus();

        }

        else{

            $('#dropError').text("");

        }

    })
```

- Validation using plugin
- Writes a rule according to required validation and based on write error message if input is invalid so gives error based on message and submit the form at last.

```
    $('#myForm').validate({

        rules:{

            name:"required",

            email:{

                required:true,

                email:true

            },

        },


        // it is an error message
```

```
            message:{
                name:"*Name is required.",
                email:{
                    email:"*Enter valid email."
                },
            },
            // submit the form
            submitHandler: function (form) {
                alert("form submitted");
            },
```

Functions

```
    var numbers = [1, 2, 3, 4, 5];
```

- **map** -- manipute the array

```
    var squaredNumbers = $.map(numbers, function (num) {
        return num * num;
    });
    console.log("map : "+squaredNumbers);
```

- **grep** -- like a filter in javascript, filter the data

```
    var evens = $.grep(numbers, function (num) {
        return num % 2 === 0;
    });
```

```
console.log("grep: "+evens);
```

- **extend** -- merge the object into target object perform override

```
var obj1 = { a: 1,b: 2 };
var obj2 = {  b: 3, c: 4 };
var result = $.extend({}, obj1, obj2);
console.log("extend: "+JSON.stringify(result));
```

- **each** -- traverse the array

```
$.each(numbers, function (index, value) {
    console.log("Idx: " + index + ", Value: " + value);
});
```

- **merge** -- merge the array

```
var array1 = [1, 2,4, 3];
var array2 = [4, 5, 6];
$.merge(array1, array2);
console.log("merge: "+array1);
```

Regex function

- to find the whether 'e' is existing or not -- return true/false

```
const pattern_test = /e/;
const result_test = pattern_test.test("sdedjg");
console.log(result_test);
```

- to find the whether 'e' is existing or not -- return the object with index

```
const pattern_exec = /e/;

const result_exec = pattern_exec.exec("sdedjg");

console.log(result_exec);
```

- validate the email

```
const email = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$/

const result_email = email.test("dev2003@gmail.com")

console.log(result_email);
```

- check strting of the string contains 'the' or not

```
// /i -- case insensitive

const start = /^the/i;

const result_start = start.test("theasdba");

console.log(result_start);
```

- check ending of the string contains 'he' or not

```
const end = /he$/;

const result_end = end.test("shdfjshe");

console.log(result_end);
```

- check the string contain 3 consecutive digits or not

```
const digit = /\d{3}/;
const result_digit = digit.test("jdjsd223");

console.log(result_digit);
```

- return the number with 1 to 4 with inclusive

```
const text = "123456789";

const result_text = text.match(/[1-4]/g);

console.log(result_text);
```

CallBack Function – Call the another function after one function is completely executes.

```
$('#p1').click(function(){

    $('#p1').hide(1000,function(){

        $('#p2').click();

    })

})

$('#p2').click(function(){

    $('#p2').hide(1000,function(){

        $('#p3').click();

    })

})

$('#p3').click(function(){

    $('#p3').hide(1000,function(){

        alert("Callback is done")

    })

})
```

Deferred and Promise

```
  function asyncOperation() {
  var deferred = $.Deferred();


  //asynchronous operation using setTimeout
  setTimeout(function() {
    let data = {
      fname : "Dev",
      lname : "Nakum",
    }
    var success = Math.random() > 0.5;


    if (success) {
      console.log("Operation successful!!!");
      deferred.resolve(data);
    } else {
      console.log("Operation failed!");
      deferred.reject("Operation failed!");
    }
  }, 2000);
  return deferred.promise();
}
```

```
// Use the promise returned by asyncOperation
var promise = asyncOperation();


promise.then(function(data) {
  console.log("Success:", data);
  return data;
}).then(function(result){
  console.log(result);
})
.catch(function(error) {
  console.log("Error:", error);
});
```

AJAX

- AJAX stands for Asynchronous JavaScript and XML
- Ajax enables a web application user to interact with a web page without the interruption of constant web page reloading.
- Examples of applications using AJAX: Google Maps, Gmail, Youtube, and Facebook tabs.

```
$('#btnSubmit').click(function(e){

    e.preventDefault();

    let todo= $("#todo").val();

    let userData ;


    // getUserTodo retrun the promise

    let getUserTodo = ()=>{

        return $.ajax({

            url:`https://jsonplaceholder.typicode.com/todos/${todo}`,

            method:"get",

            success:function(result){

                console.log("data is successfully get");

            },

            error:function(error){

                console.log(error);

            }

        }).promise();

    }


    // whenever required to the data use below code
```

```
        getUserTodo()
          .then((result)=>{
            handleData(result);           // send the data to the another function
for further manipulation
          })
          .catch((err)=>{
            console.log(err);
          })

        const handleData = (result)=>{
          console.log(result);        // store the data into database or whatever
        }
      })
```

HTTP Request Method

- **GET** → Retrieve the data from API
- **POST** → Insert the data into API
- **PUT** → Update the entire collection
- **DELETE** → Delete the data into API
- **PATCH** → Update the data only specific parameter