

# **JavaScript Object Model (ECMA)**

## **Programming In SharePoint 2016 And**

## **Office 365**

*This free book is provided by courtesy of [C# Corner](#) and Mindcracker Network and its authors. Feel free to share this book with your friends and co-workers. Please do not reproduce, republish, edit or copy this book.*

**Priyaranjan K S**

**SharePoint developer  
C# Corner MVP**

## Table of Contents

I.	Introduction .....	1
A.	JSOM Working.....	1
B.	Script on Demand.....	2
C.	JSOM Structure .....	3
D.	Exception Handling .....	6
II.	Working with Content Type .....	10
A.	Add Site Column to Site Content Type.....	10
B.	Create a List Content Type.....	11
C.	Create Site Content Type .....	13
D.	List All List Content Type .....	14
E.	List All List Content Types .....	15
III.	Working with Document Library.....	17
A.	Create a Document Library .....	17
B.	Enable Minor Versions.....	18
C.	Enable Major Versions .....	19
D.	Get Base template ID .....	20
E.	Get List of Event Receivers.....	21
F.	Get Form URLs .....	23
G.	Hide Document Library .....	24
H.	Enable Document Check out.....	25
I.	Add Library to Quick Launch .....	26
J.	Set Image URL .....	28
IV.	Working with Fields .....	30
A.	Create Different Types of Text Fields.....	30
B.	Create Text, Boolean and URL fields.....	31
C.	Create Number, Percentage and User fields .....	33
D.	Get Field Type .....	34
E.	Create LookUp and DateTime Fields.....	35
F.	Assign Values to List Item .....	37
G.	Get Field Values of List Item .....	38
H.	Create Choice Fields.....	40

I.	Update Field Properties .....	41
J.	Delete Field .....	43
V.	Working with Files.....	44
A.	Check Out File .....	44
B.	Check In File .....	45
C.	Delete File .....	46
D.	Delete Attachment File .....	47
E.	List all Attachment Files .....	49
F.	Get File Properties (Author, Modified By) .....	50
G.	Get All Versions.....	51
H.	Get All Checked Out Files.....	53
I.	Get Checked Out User.....	54
J.	Get Major Version.....	55
K.	Get Minor Version.....	57
L.	Publish File .....	58
M.	Restore File Version .....	59
N.	Delete Version by Label .....	60
O.	Delete All Versions.....	61
P.	Discard Checkout .....	62
Q.	Get All Files from Library.....	64
R.	Create Attachment Folder .....	65
S.	Add File to Document Library .....	67
T.	Add File to List as Attachment .....	69
VI.	Working with Folder .....	72
A.	Break Inheritance.....	72
B.	Create Folder.....	73
C.	Edit Folder Name .....	75
D.	Get All Folders.....	76
E.	Get Files from Folder.....	78
F.	Get Folder Properties.....	79
VII.	Working with Group.....	81
A.	Add Role.....	81
B.	Add User.....	82

C.	Edit Membership.....	84
D.	Get User's Group Membership Properties .....	85
E.	Get User Group Properties .....	86
F.	Delete Group.....	87
G.	Get Site Groups.....	89
H.	Get Group Users.....	90
I.	Remove User from Group .....	92
J.	Set Approver Email .....	93
K.	Set User as Owner of Group .....	94
L.	Set a Group as Owner of Another Group.....	95
VIII.	Working with List .....	97
A.	Allow Attachments.....	97
B.	Allow Folder Creation .....	98
C.	Allow Management of Content Types .....	100
D.	Break Inheritance.....	101
E.	Create List .....	102
F.	Delete List .....	103
G.	Set Draft Item Security.....	105
H.	Get Base Template.....	106
I.	Get List Properties.....	107
J.	Get All Lists.....	108
K.	Set List as Crawlable.....	110
L.	Get Entity Type Full Name .....	111
M.	Set Validation Formula .....	113
N.	Enable List Versioning .....	114
O.	Reset Inheritance .....	116
P.	Update List .....	117
IX.	Working with List Items .....	119
A.	Create List Item .....	119
B.	Batch Create List Item.....	120
C.	Retrieve List Item Conditionally.....	122
D.	Retrieve All List Items.....	123
E.	Delete List Items .....	125

X.	Working with Navigation .....	127
A.	Add Top Navigation Nodes .....	127
B.	Add Quick Launch Nodes .....	128
C.	Delete Quick Launch Navigation Nodes.....	130
D.	Delete Top Navigation Nodes .....	132
E.	Get Child Nodes .....	134
F.	Get Quick Launch Nodes.....	135
G.	Get Top Navigation Nodes .....	137
H.	Update Quick Launch Nodes.....	139
XI.	Working with Document Set.....	141
A.	Create a Document Set.....	141
B.	Get All Files from Document Set.....	143
XII.	Working with Publishing Page .....	146
A.	Create Publishing Page.....	146
B.	Set Publishing Page Content .....	148
C.	Get Publishing Page Content .....	150
XIII.	Working with Property Bag.....	152
A.	Create Property Bag Value.....	152
B.	Read Property Bag Value .....	153
XIV.	Working with Roles .....	155
A.	Get all Roles .....	155
B.	Create Role Definition.....	156
C.	Update Role Definition.....	158
D.	Delete Base Permissions .....	160
E.	Remove Role Definition from User .....	162
F.	Remove Role Definition from Group .....	164
G.	Add Role Definition to User .....	166
H.	Add Role Definition to Group.....	169
I.	Remove Role Definition .....	170
XV.	Working with Views .....	172
A.	Create View.....	172
B.	Get All List Views.....	174
C.	Update List Views.....	175

D.	Get Fields from List View .....	177
E.	Set JSLink in List View.....	178
F.	Delete View.....	180
XVI.	Working with Web .....	182
A.	Add Language.....	182
B.	Get Current Locale .....	183
C.	Get Time Zone.....	184
D.	Get Web Language.....	185
E.	Get Specific Web Properties .....	187
F.	Get All Web Properties .....	188
G.	Remove Language .....	189
H.	Apply Theme .....	191
I.	Get Subwebs .....	192
J.	Get Features List .....	194
K.	Get Server Date and Time.....	195
L.	Get Supported Languages.....	196
M.	Get Web Templates .....	197
XVII.	Working with Users.....	199
A.	Check if User has Full Permissions .....	199
B.	Check if User is Present in Group.....	200
C.	Get All Users in the Web.....	202
D.	Get Current User Properties .....	203
E.	Get User's Groups .....	205
F.	Get User Information .....	206
XVIII.	Working with Search.....	208
A.	Display Managed Property in Results .....	208
B.	Filter by Managed Property .....	210
C.	Filter by Result Source .....	212
D.	Add Multiple Filters.....	213
E.	Refinement Filters by Filetype .....	215
F.	Refinement Filters by Date Time .....	217
XIX.	Working with Social Features .....	219
A.	Follow Document .....	219

B.	Followed Users.....	220
C.	Follow Site.....	222
D.	Follow User .....	223
E.	Get Followed Documents.....	224
F.	Get Followed Sites .....	226
G.	Get Followed Users .....	227
H.	Get Following Status .....	228
I.	Stop Following Document.....	230
J.	Stop Following Site.....	232
K.	Stop Following User .....	234
XX.	Working with User Profile.....	236
A.	Get User Profile.....	236
B.	Get Single User Profile Properties.....	237
C.	Get Multiple User Profile Properties of a User .....	239
D.	Get User Profile Properties for Multiple Users.....	240
E.	Set Multi Value User Profile Property.....	242
F.	Set Current User Properties .....	244
XXI.	Working with Modal Dialog .....	247
A.	Call Out menu .....	247
B.	Modal Dialog.....	248
XXII.	Working with Taxonomy.....	250
A.	Create TermGroup .....	250
B.	Create Termset .....	252
C.	Create Term .....	254
D.	Create Child Term .....	256
E.	Create Label .....	258
F.	Copy Term.....	260
G.	Get All Labels.....	262
H.	Get All Term Store.....	264
I.	Get Default Term Store .....	265
J.	Get All Groups.....	267
K.	Get Terms from Single Level .....	269
L.	Get Termsets from Group .....	270

M.	Get Terms from Termsets .....	272
N.	Move Term.....	273
O.	Move Termset.....	276
P.	Move Term as Child Term .....	278
Q.	Deprecate Term .....	281
R.	Re-enable Deprecated Term .....	283
S.	Delete Label .....	284
T.	Delete Terms.....	286
U.	Delete Termset.....	289
V.	Delete Group.....	291
XXIII.	Working with Custom User action .....	294
A.	Add Custom Action to ECB menu.....	294
B.	Add Custom Action to Site Settings .....	296
C.	Add Custom Action to Ribbon.....	298
D.	Add Custom Action to Site Actions .....	300
E.	Get Available List Custom Actions .....	302
F.	Edit Custom Action.....	304
G.	Delete Custom Action .....	306
XXIV.	Summary .....	308

## About the Author

**Priyaranjan KS** is a Senior SharePoint Consultant engaged in architecting, designing, and developing solutions in SharePoint and Office 365. He has been working on SharePoint for over 7 years and has worked on SharePoint 2007 through SharePoint 2016. He is a certified Scrum Master as well as a Microsoft certified Solutions Developer (SharePoint Apps).

He is a C# Corner MVP and frequently collaborates with them in the field of SharePoint. In case you need any SharePoint help, you can either find him [here](#) or drop a [mail](#) to him.

## Target Audience

The users reading this book needn't to have in-depth working knowledge of SharePoint. Even without expert knowledge, administrators and developers can set up a SharePoint 2016 farm in Azure as this book provides a step by step guide with screenshots.



**Priyaranjan KS**

**SharePoint developer  
C# Corner MVP**

## **Acknowledgement**

I would like to take this opportunity to thank Mahesh Chand, Praveen Kumar and Dinesh Beniwal for facilitating the knowledge sharing platform - [C# Corner](#). The C# Corner team has motivated and created a foothold for people like me to step on.

I would also like to thank Vijai Anand, Jean Paul, Destin Joy, Dhananjay Kumar, and Shivprasad Koirala for being the motivational factor behind writing this book and for leaving the footprint for others to follow.

Thanks a ton to Renju Raj and Jyothi NV for the contributions, guidance, motivation and for doing the technical review of this book.

## I. Introduction

Since the entry of SharePoint into the Content Management Field, developers had to interact with it using the server side object model. Though Out of the box web services were available, yet it had its own limited development capability. With the introduction of client side object model, things became much easier. Now, developers could interact with SharePoint remotely. Client side object model comes in 3 flavors:

- .Net managed client side object model (CSOM)
- JavaScript object model (JSOM)
- Silverlight client side object model

.Net client side object model is used to access SharePoint remotely from a console application, and provides strongly-typed representation of SharePoint objects, using the client library Microsoft.SharePoint.Client.dll .

JavaScript object model, on the other hand, is used to access and interact with SharePoint from within the client browser in an asynchronous fashion. We can make use of Content Editor Web part or SharePoint apps to directly communicate with SharePoint by making use of JSOM. JSOM particularly makes use of the SP.js library to handshake with SharePoint server.

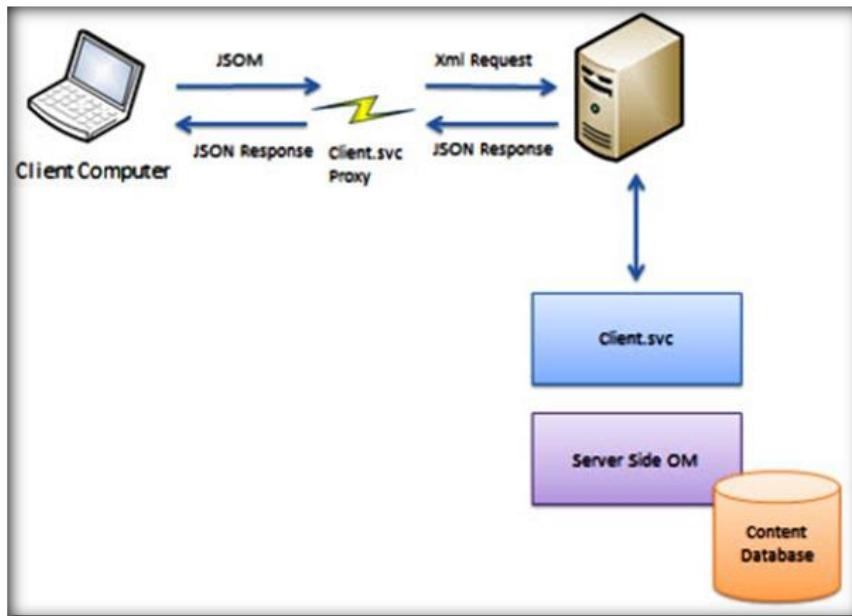
Silverlight variant has become more or less obsolete as Silverlight is losing its importance these days.

### A. JSOM Working

Just like any other API, JSOM is dependent on 3 main client side libraries. These are referenced via SharePoint Master Pages. However, if we are using a non-SharePoint custom master page, ensure we have the references to the below libraries:

- MicrosoftAjax.js – Usually loaded by ScriptResource.axd
- SP.Runtime.js – Loaded from the LAYOUTS folder
- SP.JS – Loaded from the LAYOUTS folder

In order to perform an operation using JSOM, the request goes from the client browser to the REST service named **client.svc** residing in the SharePoint Server. This is an asynchronous call, because after the request has been made, the code will continue to run without waiting for the result. When the request returns, one of the success/failure call back methods that are registered will be called to take care of the returned data.



Client.svc service is the WCF service that is responsible for performing all of the operations requested by JSOM. The service is located at `/_vti_bin/client.svc` on the SharePoint Server. The WCF service endpoint append `'/_vti_bin/client.svc/ProcessQuery'` to process the request that is coming from the client browser and requires a formatted XML Request as per the protocol specifications. So, the request coming from the client browser will be xml formatted and the return type will be a JSON response.

## B. Script on Demand

JavaScript object model extensively makes use of script on demand framework (SP.SOD). It means that the script won't be loaded until it is explicitly requested. Starting from SharePoint 2010 the scripts will be loaded on to the page only as required. Multiple methods are available in SP.SOD framework to implement On Demand scripting. We will look into two main commonly used functions which are utilized extensively in JSOM implementations.

### SP.SOD.executeOrDelayUntilScriptLoaded

`SP.SOD.executeOrDelayUntilScriptLoaded(functionName, scriptFileName)` schedules an asynchronous callback function (`functionName`) which will be called when the script(`scriptFileName`) has finished loading to the page. This method ideally delays the execution of the JSOM code until the dependent scripts has been loaded. 'SP.js' is the primary resource file required for JSOM. If the JSOM code executes prior to the loading of SP.js library, we will get object reference errors. So, to make sure JSOM is executed after SP.js is loaded on to the page, we make use of `SP.SOD.executeOrDelayUntilScriptLoaded` as :

### Usage :

```
SP.SOD.executeOrDelayUntilScriptLoaded(getWeb, 'SP.js');

function getWeb() {

    // Get the current client context and Web instance.
    var clientContext = new SP.ClientContext.get_current();
    var oWeb = clientContext.get_web();
}
```

One of the limitations of **SP.SOD.executeOrDelayUntilScriptLoaded** is that if the script file is not loaded, the JSOM code block that is waiting to be executed will wait endlessly and may not even be executed at all. In order to circumvent this issue, SP.SOD has another useful function **SP.SOD.executeFunc**.

### SP.SOD.executeFunc :

ExecuteFunc method call provides the flexibility to call an On Demand Script and then run another function once the script has finished loading. It resolves the drawback with ‘ExecuteorDelayUntilScriptLoaded (It cannot trigger an On Demand Script).

### Usage :

```
SP.SOD.executeFunc('sp.js', 'SP.ClientContext', retrieveListItems);
```

ExecuteFunc method takes three parameters:

- Sp.js – The script file that will be invoked and loaded to the page. It can be any file.
- SP.ClientContext – The main object within the script file that will be used. It can be kept null.
- retrieveListItems – The function to run after the script files are loaded to the page.

## C. JSOM Structure

Let's look at the structure of JSOM implementation and see how the control flows through the code. Given below is a simple script to retrieve list items from SharePoint.

```
$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', retrieveListItems);
});

var collListItem ;
function retrieveListItems() {
```

```

//Get client context,web and Listobject
var clientContext = new SP.ClientContext();
var oWeb = clientContext.get_web();
var oList = oWeb.get_lists().getByTitle('Employees');

//Get list item collection
collListItem = oList.getItems(SP.CamlQuery.createAllItemsQuery());

//Load the client context and execute the batch
clientContext.load(collListItem);
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);

}

function QuerySuccess() {

    //Create enumerator object and loop through the collection
    var listItemEnumerator = collListItem.getEnumerator();
    while (listItemEnumerator.moveNext()) {
        var oListItem = listItemEnumerator.get_current().get_item('Name');
        console.log("Name : " + oListItem );
    }
}

function QueryFailure() {
    console.log('Request failed'+ args.get_message());
}

```

## Code Walkthrough:

1. In order to work with JSOM, get the current context by calling ‘SP.ClientContext’ or ‘SP.ClientContext.get\_Current’ method. Since JSOM is running in the context of the browser we cannot impersonate the user. The only way to change the user context is to run the client browser in the required user’s credential.

```
var clientContext = new SP.ClientContext();
```

2. Once the context object is created, make a call to get the current web .

```
var oWeb = clientContext.get_web();
```

3. The get\_lists() method of the web object returns all the list collection present in the current web. Applying getByTitle () on the list collection fetches the specific Employees list.

```
var oList = oWeb.get_lists().getByTitle('Employees');
```

4. Now, call ‘getItems’ method on the list object to return all the items

```
collListItem = oList.getItems(SP.CamlQuery.createAllItemsQuery());
```

5. The context object provides the load method which is responsible for loading objects and collections from the server. Whatever object is passed into the load method is populated and can be used in the subsequent call back function.

```
clientContext.load(collListItem);
```

6. If you want to work with the child properties of an object, it must be loaded by calling ‘clientcontext.load(object)’ followed by the ‘executequeryasync’ call. Otherwise, you will be greeted with an exception, as shown below:

*“The property or field has not been initialized. It has not been requested or the request has not been executed. It may need to be explicitly requested.”*

7. JSON provides the flexibility to execute the requests in batches. Thus, each request to create a web object, creates a list from the web object that won’t be sent as separate requests to the server. Instead, all the requests will be batched together and sent to client.svc upon calling the executequeryasync call. In the above example, we are making calls on the context, web, and list objects but have not really instantiated the objects. Only when the call to ExecuteQueryAsync method is made, an XML request is generated and then sent to the SharePoint server’s client.svc service for processing.

As part of ExecuteQueryAsync, you have to provide two callback functions – a success and a failure call back. These methods specify how to proceed when the request is processed.

```
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
```

8. Depending on the Success/ Failure of the executeQueryAsync call, the control will be passed to the appropriate call back function. From within the function, we can then access properties of the loaded object. For instance, in the above example, the loaded object is the list item collection which can be iterated by instantiating an enumerator object as shown below

```
var listItemEnumerator = collListItem.getEnumerator();
while (listItemEnumerator.moveNext()) {
    var oListItem = listItemEnumerator.get_current().get_item('Name');
    console.log("Name : " + oListItem);
}
```

JavaScript object model is an asynchronous programming model that enables communication with SharePoint in the client side. As with any programming constructs JavaScript object model also has exception handling techniques. JavaScript provides native try/catch to handle exceptions. Though JSOM is based on JavaScript, using Try/Catch will not really do exception handling in JSOM. This is because of the asynchronous nature of JSOM. In JSOM, all the operations are send as a batch to ‘client.svc’ WCF service in the server that would process the request. Hence, native try/catch cannot keep track of the errors. Instead, we will make use of JSOM constructs to handle errors.

#### **D. Exception Handling**

SharePoint provides an exception handling mechanism similar to JavaScript try/catch in the format startTry() and startCatch(). Its implementation is similar to try/catch.

The main object that handles the exception in JSOM Is called ‘ExceptionHandlingScope’. It provides the below methods to perform exception handling.

- startScope
- startTry
- startCatch
- startFinally

Let's see how it works:

#### **Internal Implementation**

With any JSOM implementation, we create the starting object which is client context. After that, we create an exception handling scope object using the client context. Once the exception handling scope object is created, we call its startScope method. This method indicates that the entire code within the block will be eligible for exception handling.

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function () {
    SP.SOD.executeOrDelayUntilScriptLoaded(CheckandCreateList, "sp.js");
});

function CheckandCreateList() {
    //Get the client context and web object
}
```

```

var clientContext = new SP.ClientContext.get_current();
var oWeb = clientContext.get_web();

//Create exception handling scope and start the scope
var exceptionScope = new SP.ExceptionHandlingScope(clientContext);
var startScope = exceptionScope.startScope();

//Instantiate Try Scope
var tryScope = exceptionScope.startTry();

//Add the code to update the list that does not exist which will cause error.
var oList = clientContext.get_web().get_lists().getByTitle("ExceptionList");
oList.set_description("Updated Description");
oList.update();

//Dispose the scope
tryScope.dispose();

//Instantiate Catch Scope
var catchScope = exceptionScope.startCatch();

//Within the catch scope write the JSOM code to create new list
var newListCreateInfo = new SP.ListCreationInformation();
newListCreateInfo.set_title("ExceptionList");
newListCreateInfo.set_description("Catch Updated Description");
newListCreateInfo.set_templateType(SP.ListTemplateType.genericList);
clientContext.get_web().get_lists().add(newListCreateInfo);

//Dispose the Catch Scope
catchScope.dispose();

//Instantiate the Finally Scope
var finallyScope = exceptionScope.startFinally();

//Update the newly created List
var oList = clientContext.get_web().get_lists().getByTitle("ExceptionList");
oList.set_description("Finally Updated Description");
oList.update();

//Dispose finallyScope
finallyScope.dispose();

// Dispose startScope
startScope.dispose();

//Execute the batch
clientContext.executeQueryAsync(Success,Failure);
}

function Success() {
    console.log("The list has been created successfully.");
}

function Failure(sender, args) {
    console.log('Request failed - ' + args.get_message());
}

</script>

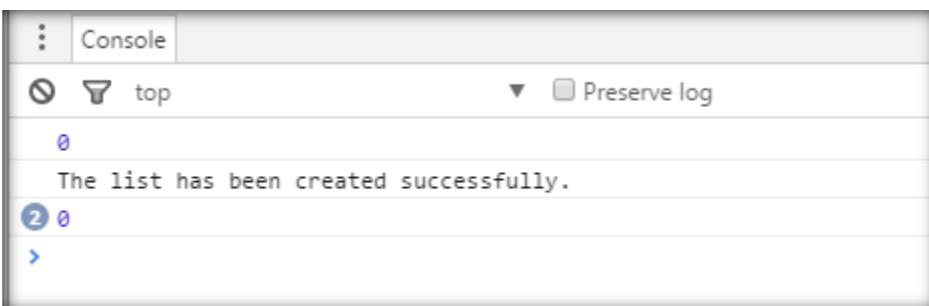
```

Once the exception handling scope starts, we can initiate the try block using the startTry method. Any code written within the startTry method will be checked for exceptions and in the event of an exception, the control will be passed to the exception handling block, the catch block. The code block that began with startTry will be ended using the dispose method. The exception handling block that will be called in case of any error in the try block is the catch block which is instantiated using startCatch method. Within this block we can add the code that will resolve the issue that happened in the try block. Similar to try block, catch block will also be ended using the dispose method.

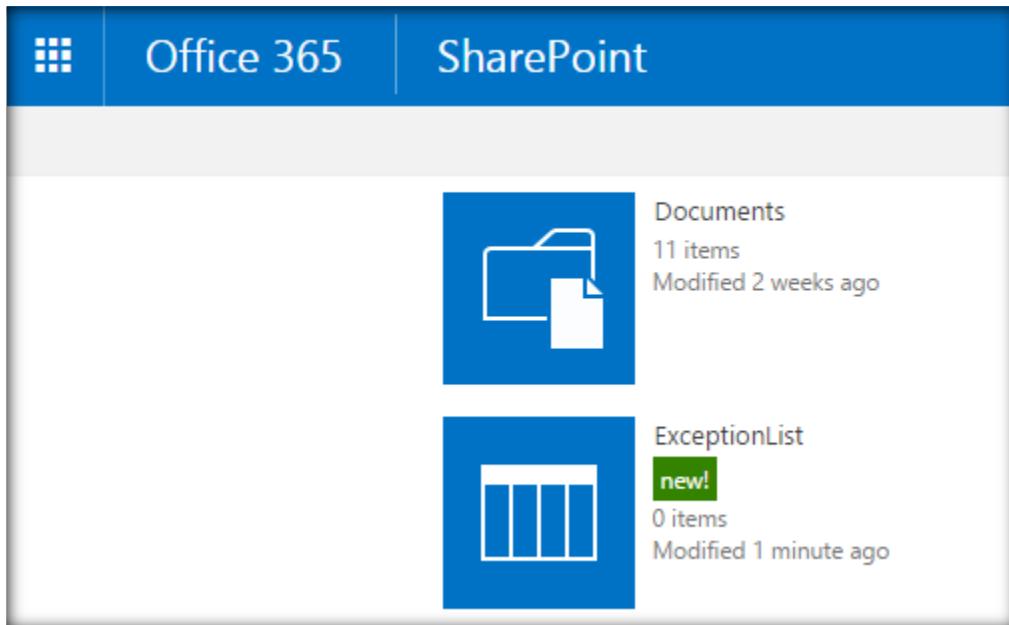
The last code block of the exception handling paradigm is the finally block which is instantiated using the startFinally () method. This block will have all the code that needs to be run regardless of if an error occurs. The dispose method can be called to indicate the end of the finally block.

We can see how exception handling paradigm can be used in a real word scenario. Let's say, we are trying to update a list that does not exist in SharePoint environment. This will ideally throw an exception as the object is not present. We will write the erroneous list update code in try block and handle the exception in the catch block by creating the list. In the finally block, we will again try to update the list. This time, the list will be updated without any exceptions.

### **Output:**



```
Console
top
▼ Preserve log
0
The list has been created successfully.
2 0
>
```



Let's see how to program against SharePoint, using JavaScript Object model. We will cover the operations against various SharePoint objects using JSOM. The code fragments contain the descriptive explanation which is given as inline comments. These code fragments can be used directly inside the SharePoint Hosted Apps or Content Editor Webparts. They have been tested in both SharePoint 2016 and Office 365 (SharePoint Online).

## II. Working with Content Type

### A. Add Site Column to Site Content Type

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to the Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', addSiteColumn);
});

var clientContext,oSiteCol,oSiteContentTypes;
function addSiteColumn() {

    //Get Client Context and Web Object
    clientContext = new SP.ClientContext.get_current();
    var oWeb = clientContext.get_web();

    //Get Site Column object and Site Content Type object
    oSiteCol = oWeb.get_fields().getByInternalNameOrTitle('Defect Count');
    oSiteContentTypes=oWeb.get_contentTypes();

    //Load Site Column and Site Content Type to Client Context
    clientContext.load(oSiteCol);
    clientContext.load(oSiteContentTypes);

    // Execute Batch
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);

}

function QuerySuccess() {

    //Get Site Content Type enumerator and loop through it
    var oSiteCTEnumerator = oSiteContentTypes.getEnumerator();
    var currentCT;
    while(oSiteCTEnumerator.moveNext()) {
        currentCT=oSiteCTEnumerator.get_current();

        //Check if current CT is the desired CT to which the Site Column has to be
        added
        if(currentCT.get_name() == "Defect Tracker") {
```

```

//Add Site Column to CT
var newSiteCol = new SP.FieldLinkCreationInformation();
newSiteCol.set_field(oSiteCol);
currentCT.get_fieldLinks().add(newSiteCol);
currentCT.update(true);

//Load client context and execute batch
clientContext.load(currentCT);
clientContext.executeQueryAsync(SecondQuerySuccess, SecondQueryFailure);
}

}

function QueryFailure(sender,args) {
    console.log('Request failed - '+args.get_message());
}

function SecondQuerySuccess() {
    console.log("Site column has been added to Site Content Type");
}

function SecondQueryFailure(sender,args) {
    console.log('Request failed - '+args.get_message());
}
</script>

```

## Output:

Errors	Multiple lines of text
Defect Count	Single line of text
■ Add from existing site columns	

The Site column ‘Defect Count’ has been added to the Site Content Type.

## B. Create a List Content Type

### Steps:

- Add the Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

```

```

SP.SOD.executeFunc('sp.js', 'SP.ClientContext', addListCT);

});

function addListCT() {

    //Create client context,Web and List object
    clientContext = new SP.ClientContext.get_current();
    var oWeb = clientContext.get_web();
    var oList=oWeb.get_lists().getByTitle("Employees");

    //Get List Content Types Collection and List Item CT ID
    oListContentTypeCollection = oList.get_contentTypes();
    var oListCT = oListContentTypeCollection.getById("0x0101");

    //Create Content Type Creation Information and add it to the CT Collection
    var newListCT = new SP.ContentTypeCreationInformation();
    newListCT.set_name('Defect Tracker CT');
    newListCT.set_parentContentType(oListCT);
    newListCT.set_description('Defect Tracker Content Type at List Level');
    oListContentTypeCollection.add(newListCT);

    //Load Client Context and Execute batch
    clientContext.load(oListContentTypeCollection);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);

}

function QuerySuccess() {
    console.log("List Content Type added successfully");
}

function QueryFailure(sender,args) {
    console.log('Request failed - ' + args.get_message());
}

</script>

```

## Output:

**Content Types**

This list is configured to allow multiple content types. Use content types to specify the information types are currently available in this list:

Content Type	Visible on New Button
Item	✓
Defect Tracker CT	✓

## C. Create Site Content Type

### Steps:

- Add Content Editor web part (CEWP) to the SharePoint page
- Save the below script as a text file and upload it to Site Assets page
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', addSiteCT);
});

function addSiteCT() {
    //Get Client Context and Web object
    clientContext = new SP.ClientContext.get_current();
    var oWeb = clientContext.get_web();

    //Get Site CT Collection and Parent Site CT Object
    oSiteContentTypeCollection = oWeb.get_contentTypes();
    var oSiteCT = oSiteContentTypeCollection.getById("0x0101");

    //Create Site CT and add to the Site CT Collection
    var newSiteCT = new SP.ContentTypeCreationInformation();
    newSiteCT.set_name('Defect Tracker CT');
    newSiteCT.set_parentContentType(oSiteCT);
    newSiteCT.set_group("Custom Content Types");
    newSiteCT.set_description('This is a Defect Tracker Content Type');
    oSiteContentTypeCollection.add(newSiteCT);

    //Load Client Context and Execute the batch
    clientContext.load(oSiteContentTypeCollection);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    console.log("Site Content Type added successfully");
}

function QueryFailure(sender,args) {
    console.log('Request failed - ' + args.get_message());
}
</script>

```

### **Output:**

Custom Content Types	
Custom Content Type	Item
Defect Tracker CT	Document

### **D. List All List Content Type**

#### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### **Script:**

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', listAllListCT);
});

var oListContentTypes;
function listAllListCT() {
    //Get Client Context and Web object
    var clientContext = new SP.ClientContext.get_current();
    oWeb = clientContext.get_web();

    //Get the List and Content Type Collection
    var oList = oWeb.get_lists().getByTitle('DemoDocLibrary');
    oListContentTypes=oList.get_contentTypes();

    //Load Client Context and Execute the batch
    clientContext.load(oListContentTypes);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    //Get List CT enumerator and loop through the collection
    var oListCTEnumerator = oListContentTypes.getEnumerator();
}
```

```

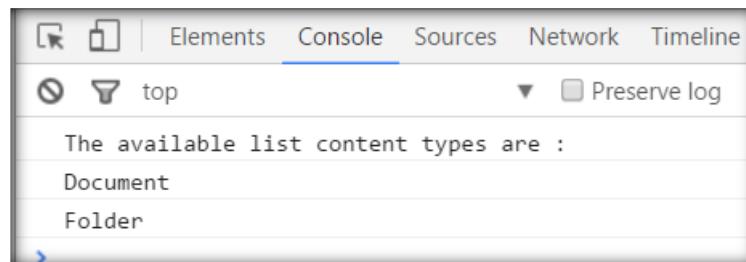
console.log("The available list content types are : ");
while (oListCTEnumerator.moveNext()) {
    console.log(oListCTEnumerator.get_current().get_name() + '\n' );
}
}

function QueryFailure(sender,args) {
    console.log('Request failed - ' + args.get_message());
}

</script>

```

## Output:



## E. List All List Content Types

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', listAllListCT);
});

var oWebContentTypes;
function listAllListCT() {

    //Get Client Context,Root Web of Site,Site CT Collection Objects
    var clientContext = new SP.ClientContext.get_current();
    oRootWeb = clientContext.get_site().get_rootWeb();
    oWebContentTypes=oRootWeb.get_contentTypes();
}

```

```

//Load Client Context and Execute the batch
clientContext.load(oWebContentTypes);
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

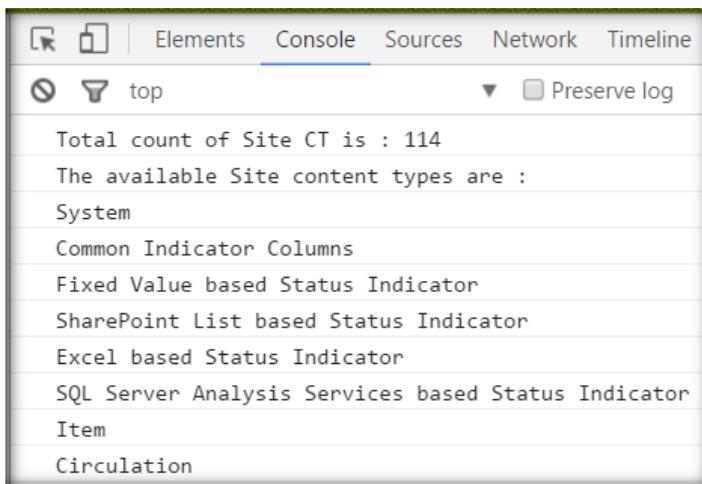
    //Get Web Enumerator and loop through the CT Collection
    var oWebCTEEnumerator = oWebContentTypes.getEnumerator();
    console.log("Total count of Site CT is : "+oWebContentTypes.get_count()+"\n");
    console.log("The available Site content types are : ");
    while (oWebCTEEnumerator.moveNext()) {
        console.log(oWebCTEEnumerator.get_current().get_name()+"\n" );
    }
}

function QueryFailure(sender,args) {
    console.log('Request failed - ' + args.get_message());
}

</script>

```

## Output:



Total count of Site CT is : 114  
The available Site content types are :  
System  
Common Indicator Columns  
Fixed Value based Status Indicator  
SharePoint List based Status Indicator  
Excel based Status Indicator  
SQL Server Analysis Services based Status Indicator  
Item  
Circulation

### III. Working with Document Library

#### A. Create a Document Library

##### Steps:

- Add Content Editor web part(CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

##### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', createDocLib);
});

function createDocLib() {

    //Get Client Context and Web Object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Create Library Creation Information and add it to List Collection
    var libraryCreationInfo = new SP.ListCreationInformation();
    libraryCreationInfo.set_title('Demo Library');
    libraryCreationInfo.set_description('This is a Demo Library');
    libraryCreationInfo.set_templateType(SP.ListTemplateType.documentLibrary);
    var oListColl = oWeb.get_lists().add(libraryCreationInfo);

    //Load Client Context and Execute batch
    clientContext.load(oListColl );
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

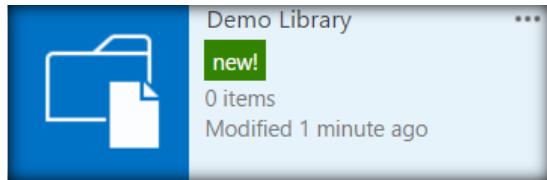
function QuerySuccess() {
    console.log("Document Library created successfully.");
}

function QueryFailure(sender,args) {
    console.log('Request failed - ' + args.get_message());
}

</script>

```

## Output:



## B. Enable Minor Versions

### Steps:

- Add Content Editor web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', enableMinorVersions);
});
function enableMinorVersions() {

    //Get Client Context and Web Object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get Document Library , enable minor versions and version limit.
    var oDocLib = oWeb.get_lists().getByTitle('Demo Library1');
    oDocLib.set_enableMinorVersions(true)
    oDocLib.set_majorWithMinorVersionsLimit('200');
    oDocLib.update();

    //Load Client Context and Execute batch
    clientContext.load(oDocLib);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    console.log("Minor Version has been enabled for the Library.");
}

function QueryFailure(sender,args) {
    console.log('Request failed - '+ args.get_message());
}</script>
```

## **Output:**

Document Version History

Specify whether a version is created each time you edit a file in this document library. [Learn about versions.](#)

Create a version each time you edit a file in this document library?

No versioning

Create major versions  
Example: 1, 2, 3, 4

Create major and minor (draft) versions  
Example: 1.0, 1.1, 1.2, 2.0

Optionally limit the number of versions to retain:

Keep the following number of major versions:  
500

Keep drafts for the following number of major versions:  
200

## **C. Enable Major Versions**

### **Steps:**

- Add Content Editor web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### **Script:**

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', enableMajorVersion);
});

function enableMajorVersion() {

    //Get Client Context and Web object .
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get Document Library , Enable major version and set version limit
    oDocLib = oWeb.get_lists().getByTitle('Demo Library1');
    oDocLib.set_enableVersioning(true);
    oDocLib.set_majorVersionLimit('100');
    oDocLib.update();
}
```

```

//Load Client Context and Execute batch
clientContext.load(oDocLib);
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    console.log("Major Version enabled for the document library");
}

function QueryFailure(sender,args) {
    console.log('Request failed - ' + args.get_message());
}

</script>

```

## **Output:**

Document Version History

Specify whether a version is created each time you edit a file in this document library. [Learn about versions.](#)

Create a version each time you edit a file in this document library?

No versioning

Create major versions  
Example: 1, 2, 3, 4

Create major and minor (draft) versions  
Example: 1.0, 1.1, 1.2, 2.0

Optionally limit the number of versions to retain:

Keep the following number of major versions:

Keep drafts for the following number of major versions:

## **D. Get Base template ID**

### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### **Script:**

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getBaseTemplateID);
});

var oDocLib;
function getBaseTemplateID() {

    //Get Client Context and Web object.
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get Document Library
    oDocLib = oWeb.get_lists().getByTitle('Demo Library1');

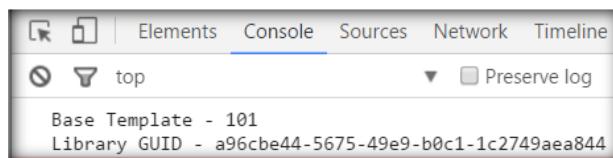
    //Load Client Context and Execute batch
    clientContext.load(oDocLib);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    //Fetch the base template ID
    console.log("Base Template - " + oDocLib.get_baseTemplate() + "\nLibrary GUID - " + oDocLib.get_id());
}

function QueryFailure(sender,args) {
    console.log('Request failed - ' + args.get_message());
}
</script>

```

## Output:



## E. Get List of Event Receivers

### Steps:

- Add Content Editor web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getEventReceivers);
});

var docLib,docLIBER;
function getEventReceivers() {

    //Get Client Context and Web object.
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get Document Library and Event Receivers collection
    docLib = oWeb.get_lists().getByTitle('Demo Library1');
    docLIBER=docLib.get_eventReceivers();

    //Load Client Context and Execute batch
    clientContext.load(docLIBER);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

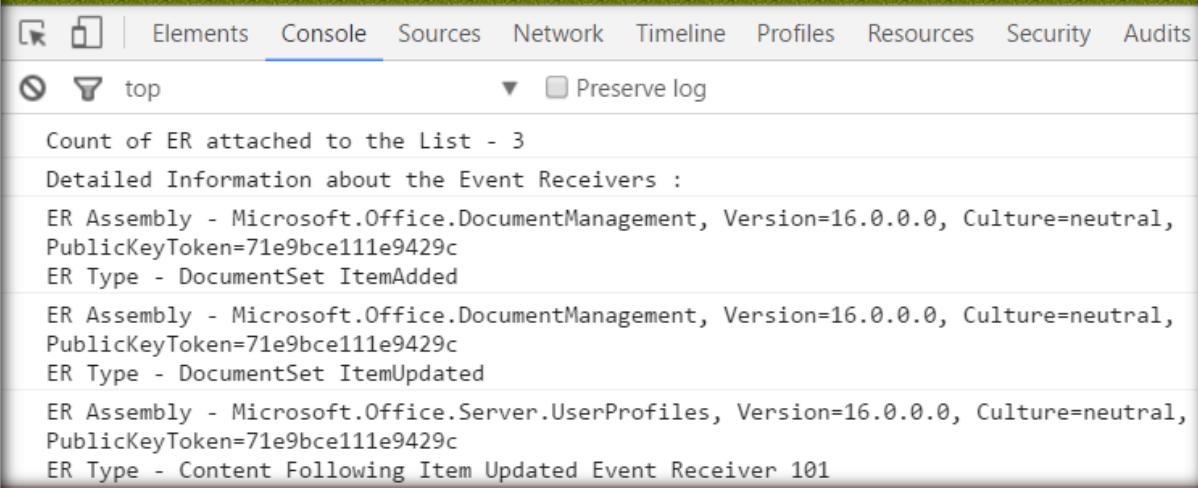
function QuerySuccess() {

    //Get Event Receivers Enumerator and loop through each ER
    var EREnominator = docLIBER.getEnumerator();
    console.log("Count of ER attached to the List - "+docLIBER.get_count()+'\n');
    console.log("Detailed Information about the Event Receivers : "+'\n');
    while (EREnominator.moveNext()) {
        var currentER = EREnominator.get_current();
        console.log("ER Assembly - "+currentER.get_receiverAssembly() + "\nER Type
- " + currentER.get_receiverName());
    }
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}
</script>

```

## Output:



```

Count of ER attached to the List - 3
Detailed Information about the Event Receivers :
ER Assembly - Microsoft.Office.DocumentManagement, Version=16.0.0.0, Culture=neutral,
PublicKeyToken=71e9bce111e9429c
ER Type - DocumentSet ItemAdded
ER Assembly - Microsoft.Office.DocumentManagement, Version=16.0.0.0, Culture=neutral,
PublicKeyToken=71e9bce111e9429c
ER Type - DocumentSet ItemUpdated
ER Assembly - Microsoft.Office.Server.UserProfiles, Version=16.0.0.0, Culture=neutral,
PublicKeyToken=71e9bce111e9429c
ER Type - Content Following Item Updated Event Receiver 101

```

## F. Get Form URLs

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getFormURLs);
});

var oDocLib;
function getFormURLs() {

    //Get Client Context,Web and Document Library object.
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();
    oDocLib = oWeb.get_lists().getByTitle('Demo Library');

    //Load Client Context and Execute batch
    clientContext.load(oDocLib,'DefaultNewFormUrl','DefaultEditFormUrl','DefaultDispla

```

© 2016 C# CORNER.

SHARE THIS DOCUMENT AS IT IS. PLEASE DO NOT REPRODUCE, REPUBLISH, CHANGE OR COPY.

```

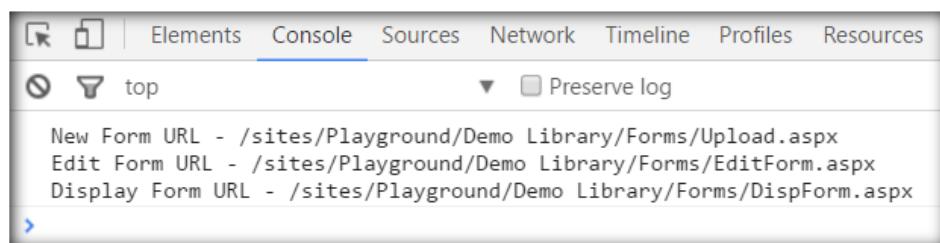
yFormUrl');
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    //Get the form URLs
    console.log("New Form URL - "+oDocLib.get_defaultNewFormUrl()+"\nEdit Form URL
- "+oDocLib.get_defaultEditFormUrl()+"\nDisplay Form URL -
"+oDocLib.get_defaultDisplayFormUrl());
}

function QueryFailure(sender,args) {
    console.log('Request failed - '+ args.get_message());
}
</script>

```

## **Output:**



## **G. Hide Document Library**

### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### **Script:**

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', hideDocLibrary);
});

function hideDocLibrary() {
    //Get Client Context and Web object.

```

```

var clientContext = new SP.ClientContext();
var oWeb = clientContext.get_web();

var oDocLib = oWeb.get_lists().getByTitle('Demo Library');
oDocLib.set_hidden(true);
oDocLib.update();

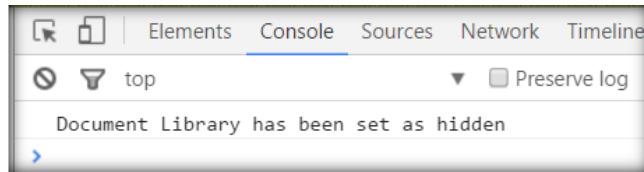
//Load Client Context and Execute batch
clientContext.load(oDocLib);
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    console.log('Document Library has been set as hidden');
}

function QueryFailure(sender,args) {
    console.log('Request failed - ' + args.get_message());
}
</script>

```

## Output:



## H. Enable Document Check out

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', setCheckOut);
});

function setCheckOut() {

```

```

//Get Client Context and Web object.
var clientContext = new SP.ClientContext();
var oWeb = clientContext.get_web();

//Get Document Library and set force check out to true
oDocLib = oWeb.get_lists().getByTitle('DestLibrary');
oDocLib.set_forceCheckout(true);
oDocLib.update();

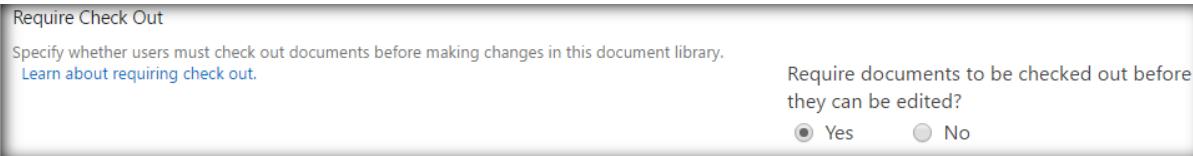
//Load Client Context and Execute batch
clientContext.load(oDocLib);
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    console.log('Force Check out has been enabled');
}

function QueryFailure(sender,args) {
    console.log('Request failed - ' + args.get_message());
}
</script>

```

## **Output:**



## **I. Add Library to Quick Launch**

### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### **Script:**

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', addToQuickLaunch);
});

```

```

function addToQuickLaunch() {
    //Get Client Context and Web object.
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get the list object and add it to quick launch
    oDocLib = oWeb.get_lists().getByTitle('Employees');
    oDocLib.set_onQuickLaunch(true);
    oDocLib.update();
    clientContext.load(oDocLib);

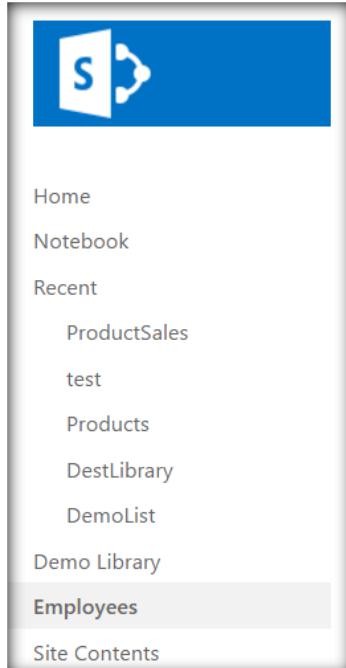
    //Execute the batch
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    console.log('List has been added to the Quick Launch');
}

function QueryFailure(sender,args) {
    console.log('Request failed - ' + args.get_message());
}
</script>

```

## Output:



## J. Set Image URL

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', setImage);
});

function setImage() {

    //Get Client Context and Web object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

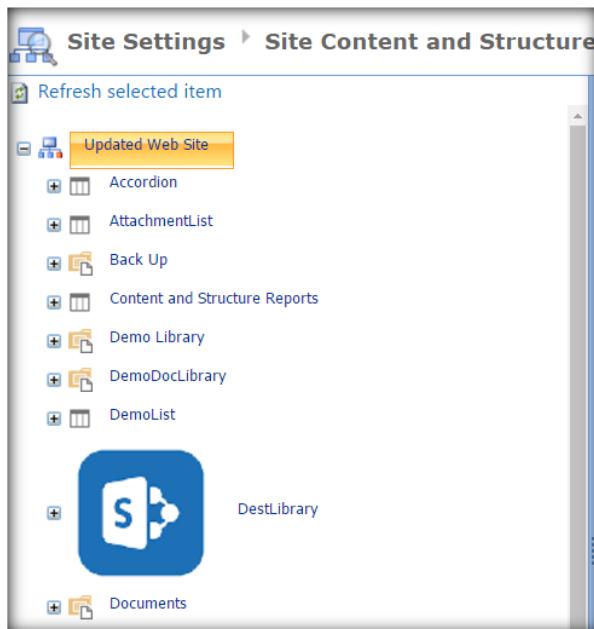
    //Get the Document Library and set an image Url.
    oDocLib = oWeb.get_lists().getByTitle('Demo Library');
    oDocLib.set_imageUrl('https://TestSite/sites/test/SiteAssets/SP2016.png');
    oDocLib.update();

    //Load Client Context and Execute batch
    clientContext.load(oDocLib);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    console.log('Image has been updated');
}

function QueryFailure(sender,args) {
    console.log('Request failed - '+ args.get_message());
}
</script>

```

**Output:**

## IV. Working with Fields

### A. Create Different Types of Text Fields

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', createFields);
});

var oListItem;
function createFields() {

    //Get Client Context and Web object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get List and Field Collection object
    var oList = oWeb.get_lists().getByTitle('Test List');
    var fieldColl = oList.get_fields();

    //Add Plain Text field to the Field Collection
    var plainTextField = fieldColl.addFieldAsXml('<Field Type="Note"'
DisplayName="Employee Address" Name="EmployeeAddress" Required="False"
NumLines="10" RichText="FALSE" AppendOnly="TRUE" />', true,
SP.AddFieldOptions.addToDefaultContentType);
    plainTextField.set_description("This is a Plain multi line field");
    plainTextField.update();

    //Add Rich Text field to the Field Collection
    var richTextField = fieldColl.addFieldAsXml('<Field Type="Note"'
DisplayName="Employee History" Name="EmployeeHistory" Required="False"
NumLines="12" RichText="TRUE" AppendOnly="TRUE" />', true,
SP.AddFieldOptions.addToDefaultContentType);
    richTextField.set_description("This is a Rich Text multi line field");
    richTextField.update();
}

```

```

//Add Enhanced Text field to the Field Collection
var enhancedTextField = fieldColl.addFieldAsXml('<Field Type="Note"
DisplayName="Employee Skillset" Name="EmployeeSkillSet" Required="FALSE"
NumLines="8" RestrictedMode="TRUE" RichText="TRUE" RichTextMode="FullHtml"
AppendOnly="TRUE" />', true, SP.AddFieldOptions.addToDefaultContentType);
enhancedTextField.set_description("This is an Enhanced multi line field");
enhancedTextField.update();

//Execute batch . Loading the client context is not necessary.
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    console.log("Fields created successfully.");
}

function QueryFailure(sender,args) {
    console.log('Request failed - ' + args.get_message());
}
</script>

```

## Output:

Employee Address	Multiple lines of text
Employee History	Multiple lines of text
Employee Skillset	Multiple lines of text

## B. Create Text, Boolean and URL fields

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', createFields);
}

```

```

    });

function createFields() {

    //Get Client Context and Web object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get List and Field Collection object
    var oList = oWeb.get_lists().getByTitle('Test List');
    var fieldColl = oList.get_fields();

    //Add Text field to the Field Collection
    var textField = clientContext.castTo(
        fieldColl.addFieldAsXml('<Field Type="Text" DisplayName="Employee Name" Name="EmployeeName" Required="True" />', true,
        SP.AddFieldOptions.addToDefaultContentType),
        SP.FieldText);
    textField.set_description("This is a text field");
    textField.update();

    //Add Boolean field to the Field Collection
    var booleanField = fieldColl.addFieldAsXml('<Field Type="Boolean" DisplayName="Retired" Name="Retired" ><Default>0</Default></Field>', true,
        SP.AddFieldOptions.addToDefaultContentType);
    booleanField.set_description("This is a boolean field");
    booleanField.update();

    //Add Image Field to the field Collection
    var imageField = fieldColl.addFieldAsXml('<Field Type="URL" DisplayName="Employee Image" Name="EmployeeImage" Required="False" Format="Image" />', true,
        SP.AddFieldOptions.addToDefaultContentType);
    imageField.set_description("This is an image field");
    imageField.update();

    //Add URL Field to the field Collection
    var hyperLinkField = fieldColl.addFieldAsXml('<Field Type="URL" DisplayName="Employee Blog URL" Name="EmployeeBlogURL" Required="False" Format="Hyperlink" />', true,
        SP.AddFieldOptions.addToDefaultContentType);
    hyperLinkField.set_description("This is a hyperlink field");
    hyperLinkField.update();

    //Execute batch . Loading the client context is not necessary.
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    console.log("Fields created successfully.");
}

function QueryFailure(sender,args) {
    console.log('Request failed - ' + args.get_message());
}
</script>

```

### **Output:**

Employee Name	Single line of text
<u>Retired</u>	Yes/No
Employee Image	Hyperlink or Picture
Employee Blog URL	Hyperlink or Picture

### **C. Create Number, Percentage and User fields**

#### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### **Script:**

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', createFields);
});

function createFields() {

    //Get Client Context and Web object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get List and Field Collection object
    var oList = oWeb.get_lists().getByTitle('Test List');
    var fieldColl = oList.get_fields();

    //Add Number field to the Field Collection
    var numberField = fieldColl.addFieldAsXml('<Field Type="Number" 
DisplayName="Age" Name="Age" Required="False" />', true,
SP.AddFieldOptions.addToDefaultContentType) ;
    numberField.set_description("This is a number field");
    numberField.update();

    //Add Percentage field to the Field Collection
    var percentageField = fieldColl.addFieldAsXml('<Field Type="Number" 

```

```

DisplayName="Training Completion" Name="TrainingCompletion" Percentage="TRUE"
Required="False" />, true, SP.AddFieldOptions.addToDefaultContentType) ;
percentageField.set_description("This is a percentage field");
percentageField.update();

//Add User Field to the collection
var userField = fieldColl.addFieldAsXml('<Field Type="User"
DisplayName="Manager" Name="Manager" Required="False" Format="Hyperlink" />',
true, SP.AddFieldOptions.addToDefaultContentType) ;
userField.set_description("This is an user field");
userField.update();

//Execute batch . Loading the client context is not necessary.
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    console.log("Fields created successfully.");
}

function QueryFailure(sender,args) {
    console.log('Request failed - ' + args.get_message());
}
</script>

```

### **Output:**

Age	Number
Training Completion	Number
Manager	Person or Group

### **D. Get Field Type**

#### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### **Script:**

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

```

```

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getFieldtype);
});

var oField;
function getFieldtype() {

    //Get Client Context and Web object
    var clientContext = new SP.ClientContext();
    var oWebsite = clientContext.get_web();

    //Get the List and the Field object.
    var oList = oWebsite.get_lists().getByTitle('Test List');
    oField = oList.get_fields().getByTitle("Employee Image");

    //Load client context and execute the batch
    clientContext.load(oField);
    clientContext.executeQueryAsync(Success, Failure);
}

function Success() {
    console.log('Field Type : ' + oField.get_typeAsString());
}

function Failure(sender,args) {
    console.log('Request failed with error message - ' + args.get_message() + ' .
Stack Trace - ' + args.get_stackTrace());
}
</script>

```



## E. Create LookUp and DateTime Fields

### Steps:

- Add Content Editor web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

## Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', createFields);
});

function createFields() {

    //Get Client Context and Web object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get List and Field Collection object
    var oList = oWeb.get_lists().getByTitle('Test List');
    var fieldColl = oList.get_fields();

    //Add Look up field to the Field Collection
    var lookupField = fieldColl.addFieldAsXml('<Field Type="Lookup"'
DisplayName="Employee Department" Name="EmpDepartment" Required="False"
List="{E1B92017-0001-475E-A978-2275B12F3FDA}" ShowField="Department"
RelationshipDeleteBehavior="None" />', true,
SP.AddFieldOptions.addToDefaultContentType) ;
    lookupField.set_description("This is a lookup field");
    lookupField.update();

    //Add Date Only field to the Field Collection
    var dateField = fieldColl.addFieldAsXml('<Field Type="DateTime"'
DisplayName="DOB" Name="DOB" Format="DateOnly" Required="False" />', true,
SP.AddFieldOptions.addToDefaultContentType) ;
    dateField.set_description("This is a date field");
    dateField.update();

    //Add Date Time field to the Field Collection
    var dateTimeField = fieldColl.addFieldAsXml('<Field Type="DateTime"'
DisplayName="Joining Date" Name="JoiningDate" Format="DateTime" Required="False"
/>', true, SP.AddFieldOptions.addToDefaultContentType) ;
    dateTimeField.set_description("This is a DateTime field");
    dateTimeField.update();

    //Execute batch . Loading the client context is not necessary.
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    console.log("Fields created successfully.");
}

function QueryFailure(sender,args) {
    console.log('Request failed - ' + args.get_message());
}
</script>

```

## Output:

Employee Department	Lookup
DOB	Date and Time
Joining Date	Date and Time

## F. Assign Values to List Item

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', AssignFieldValues);
});

function AssignFieldValues() {

//Get Client Context,Web and List object
var clientContext = new SP.ClientContext();
var oWebsite = clientContext.get_web();
var oList = oWebsite.get_lists().getByTitle('Test List');

//Create List Item Information and update the list item
var oListItemCreationInformation= new SP.ListItemCreationInformation();
oListItem= oList.addItem(oListItemCreationInformation);
oListItem.set_item('Employee_x0020_Name','Priyaranjan');
oListItem.set_item('Retired','0');
oListItem.set_item('Employee_x0020_Address','Kochi,Kerala');
oListItem.set_item('Employee_x0020_Skillset','SharePoint');
oListItem.set_item('Age','28');
oListItem.set_item('DOB','3/30/2016');
oListItem.update();
}

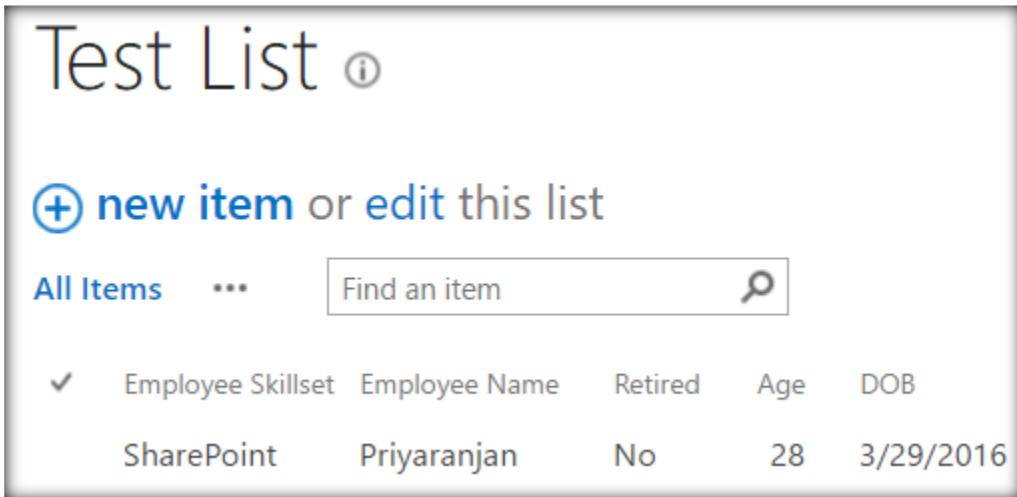
```

```
//Load the client context and Execute the batch
clientContext.load(oListItem);
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    console.log('List Item created and populated with values');
}

function QueryFailure(sender,args) {
    console.log('Request failed with error message - ' + args.get_message() + ' .
Stack Trace - ' + args.get_stackTrace());
}
</script>
```

### Output:



The screenshot shows a SharePoint list titled "Test List". At the top, there is a button labeled "⊕ new item or edit this list". Below the button, there is a search bar with the placeholder "Find an item" and a magnifying glass icon. Underneath the search bar, there is a table with the following data:

Employee Skillset	Employee Name	Retired	Age	DOB
SharePoint	Priyaranjan	No	28	3/29/2016

### G. Get Field Values of List Item

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

```

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', GetFieldValues);
});

var oListItem;
function GetFieldValues() {

    //Get Client Context and Web object
    var clientContext = new SP.ClientContext();
    var oWebsite = clientContext.get_web();

    //Get specific List item .
    oListItem = oWebsite.get_lists().getByTitle('Test List').getItemById('1');

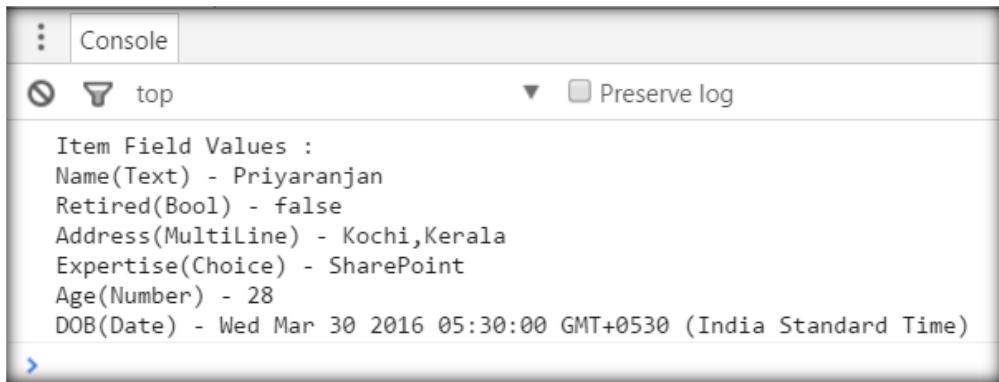
    //Load client context and execute the batch
    clientContext.load(oListItem);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    console.log('Item Field Values :\nName(Text) - 
'+oListItem.get_item("Employee_x0020_Name")+'\nRetired(Bool) - 
'+oListItem.get_item("Retired")+'\nAddress(MultiLine) - 
'+oListItem.get_item("Employee_x0020_Address")+'\nExpertise(Choice) - 
'+oListItem.get_item("Employee_x0020_Skillset") + '\nAge(Number) - ' 
+oListItem.get_item("Age")+' \nDOB(Date) - '+oListItem.get_item("DOB"));
}

function QueryFailure(sender,args) {
    console.log('Request failed with error message - ' + args.get_message() + ' . 
Stack Trace - ' + args.get_stackTrace());
}
</script>

```

## Output:



The screenshot shows a browser's developer tools console window titled "Console". The output displays the field values for a SharePoint list item, structured as key-value pairs separated by colons. The fields include Name, Retired, Address, Expertise, Age, and DOB.

```

Item Field Values :
Name(Text) - Priyaranjan
Retired(Bool) - false
Address(MultiLine) - Kochi,Kerala
Expertise(Choice) - SharePoint
Age(Number) - 28
DOB(Date) - Wed Mar 30 2016 05:30:00 GMT+0530 (India Standard Time)

```

## H. Create Choice Fields

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', createFields);
});

function createFields() {

    //Get Client Context and Web object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get List and Field Collection object
    var oList = oWeb.get_lists().getByTitle('Test List');
    var fieldColl = oList.get_fields();

    //Add Choice DropDown field to the Field Collection
    var choiceDropDownField = clientContext.castTo(
        oList.get_fields().addFieldAsXml('<Field Type="Choice"'
DisplayName="Expertise" Name="Expertise" Format="Dropdown" />', true,
SP.AddFieldOptions.addToStringDefaultContentType),
        SP.FieldChoice);
    var availableDropChoices = Array("SharePoint", "jQuery", "SQL Server");
    choiceDropDownField.set_choices(availableDropChoices);
    choiceDropDownField.update();

    //Add Choice Radio Button field to the Field Collection
    var choiceRadioButtonField = clientContext.castTo(
        oList.get_fields().addFieldAsXml('<Field Type="Choice"'
DisplayName="Experience" Name="Experience" Format="RadioButtons" />', true,
SP.AddFieldOptions.addToStringDefaultContentType),
        SP.FieldChoice);
    var availableRadioChoices = Array("<5", "5-8", "8-12");
    choiceRadioButtonField.set_choices(availableRadioChoices);
    choiceRadioButtonField.update();

    //Add Choice CheckBox field to the Field Collection
    var choiceCheckBoxField = clientContext.castTo(
        oList.get_fields().addFieldAsXml('<Field Type="MultiChoice"'
DisplayName="Preferred Location" Name="PreferredLocation" />', true,

```

```

SP.AddFieldOptions.addToDefaultContentType),
    SP.FieldMultiChoice);
var availableCheckBoxChoices = Array("London", "Kerala", "Dubai");
choiceCheckBoxField.set_choices(availableCheckBoxChoices);
choiceCheckBoxField.update();

//Execute batch . Loading the client context is not necessary.
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    console.log("Fields created successfully.");
}

function QueryFailure(sender,args) {
    console.log('Request failed - ' + args.get_message());
}
</script>

```

## **Output:**

Expertise	Choice
Experience	Choice
Preferred Location	Choice

## **I. Update Field Properties**

### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### **Script:**

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', UpdateFieldProperties);
});

function UpdateFieldProperties() {

```

```

//Get Client Context,Web and List object
var clientContext = new SP.ClientContext();
var oWebsite = clientContext.get_web();
var oList = oWebsite.get_lists().getByTitle('Test List');

//Get the field and update the property
var field = oList.get_fields().getByTitle("Manager");
var managerField = clientContext.castTo(field, SP.FieldUser);
managerField.set_lookupField("Office");
managerField.update();

//Execute the batch
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    console.log('Field Property updated.');
}

function QueryFailure(sender,args) {
    console.log('Request failed with error message - ' + args.get_message() + ' .
Stack Trace - ' + args.get_stackTrace());
}
</script>

```

## Output:

Allow selection of:

People Only  People and Groups

Choose from:

All Users  
 SharePoint Group:  
 Administrator Group ▾

Show field:

Office ▾

## J. Delete Field

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', DeleteField);
});

function DeleteField() {
    //Get Client Context,Web and List object
    var clientContext = new SP.ClientContext();
    var oWebsite = clientContext.get_web();
    var oList = oWebsite.get_lists().getByTitle('Test List');

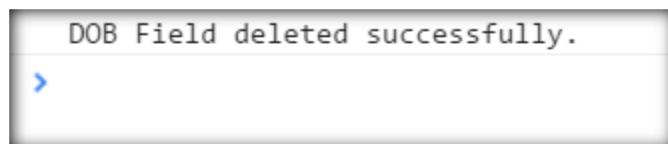
    //Get the field obejct and delete it
    var field = oList.get_fields().getByTitle("DOB");
    field.deleteObject();

    //Execute the batch . Loading client context is not necessary.
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    console.log('DOB Field deleted successfully.');
}

function QueryFailure(sender,args) {
    console.log('Request failed with error message - '+ args.get_message()+' .
Stack Trace - '+ args.get_stackTrace());
}
</script>
```

### Output:



## V. Working with Files

### A. Check Out File

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', checkOutFile);
});

function checkOutFile() {

    //Get the clientcontext,web and File object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();
    var oFile=oWeb.getFileByServerRelativeUrl('/sites/Playground/Shared
Documents/TestDoc.txt');

    //Set the check out property of the file object
    oFile.checkOut();

    //Load the client context the execute the batch
    clientContext.load(oFile);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);

}

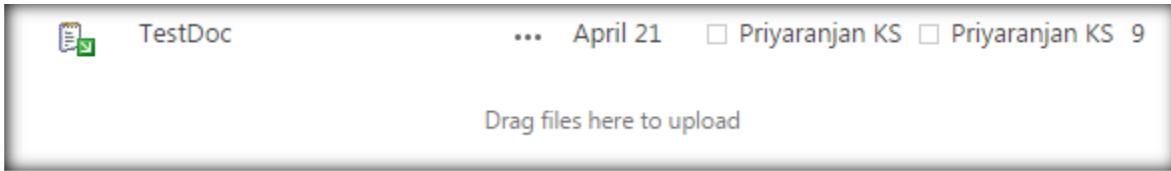
function QuerySuccess() {
    console.log("File checked out.");
}

function QueryFailure(sender,args) {
    console.log('Request failed - '+args.get_message());
}

</script>

```

## **Output:**



## **B. Check In File**

### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### **Script:**

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', checkInFile);
});

function checkInFile() {
    //Get the clientcontext,web and File object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();
    var oFile=oWeb.getFileByServerRelativeUrl('/sites/Playground/Shared
Documents/TestDoc.txt');

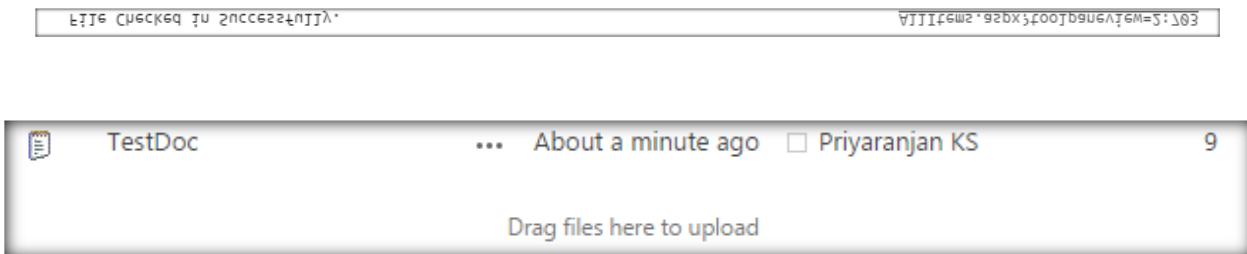
    //Set the check in property of the file object
    oFile.checkIn();

    //Load the client context the execute the batch
    clientContext.load(oFile);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);}

function QuerySuccess() {
    console.log("File Checked in Successfully.");
}

function QueryFailure(sender,args) {
    alert('Request failed - '+args.get_message());
}
</script>
```

## **Output:**



The screenshot shows a SharePoint document library interface. At the top, there is a message box with the text "File checked in successfully." To the right of the message box is a timestamp "10/10/2015 10:57 AM". Below the message box, there is a list item for a file named "TestDoc". The list item includes the file name, a "..." button, a timestamp "About a minute ago", a user name "Priyaranjan KS", and a number "9" indicating the count of items in the list. Below the list item, there is a placeholder text "Drag files here to upload".

## **C. Delete File**

### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### **Script:**

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', deleteFile);
});

var clientContext, collFiles;

function deleteFile() {

    //Get the clientcontext,web and List object
    clientContext = new SP.ClientContext();
    var oWebsite = clientContext.get_web();
    var oList = oWebsite.get_lists().getByTitle('Documents');

    //Set the caml query to fetch the file object
    var camlQuery = new SP.CamlQuery();
    camlQuery.set_viewXml('<View><Query><Where><Eq><FieldRef
Name=\'FileLeafRef\'/>' + '<Value
Type=\'File\'>TestDoc.txt</Value></Eq></Where></Query></View>');
    collFiles = oList.getItems(camlQuery);
}
```

```

//Load the client context the execute the batch
clientContext.load(collFiles);
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    //Get File collection,loop through each file and delete it.
    var itemCount = collFiles.get_count();
    for (var i = itemCount - 1; i >= 0; i--) {
        var oFile = collFiles.itemAt(i);
        oFile.deleteObject();
    };
    //Execute the batch
    clientContext.executeQueryAsync(deleteSucceeded, deleteFailed);
}

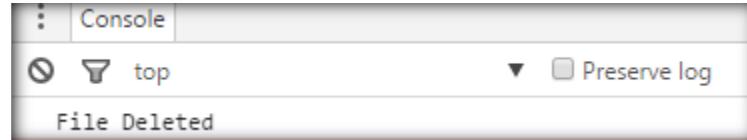
function QueryFailure(sender,args) {
    console.log('Request failed - '+args.get_message());
}

function deleteSucceeded()
{
    console.log('File Deleted')
}

function deleteFailed(sender,args)
{
    console.log('Request failed - '+args.get_message());
}
</script>

```

### Output:



### **D. Delete Attachment File**

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', deleteFileAttachment);
});

var attachmentFiles, clientContext;

function deleteFileAttachment() {

    //Get the clientcontext and web object
    clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get attachment folder for the list item and the attachment files
    var
attachmentFolder=oWeb.getFolderByServerRelativeUrl('Lists/Custom/Attachments/1');
    attachmentFiles= attachmentFolder.get_files();

    //Load clientcontext and execute the batch
    clientContext.load(attachmentFiles);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

    var deleteAttachment;
    //Loop through the attachments and delete the attachments conditionally
    var attachmentEnumerator = attachmentFiles.getEnumerator();
    while (attachmentEnumerator.moveNext()) {
        var fileName = attachmentEnumerator.get_current().get_name();
        if(fileName=="Fields.docx"){
            deleteAttachment=attachmentEnumerator.get_current();
        }
    }
    deleteAttachment.deleteObject();

    //Execute the batch
    clientContext.executeQueryAsync(SecondQuerySuccess, SecondQueryFailure);
}

function QueryFailure(sender,args) {
    console.log('Request failed - ' + args.get_message());
}

function SecondQuerySuccess() {
    console.log("Attachment Deleted.");
}

function SecondQueryFailure(sender,args) {
    console.log('Request failed - ' + args.get_message());
}
</script>

```

## Output:



## E. List all Attachment Files

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getListAttachments);
});

var attachmentFiles;
function getListAttachments() {

    //get client context and web
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get the attachment folder for the list item and get the attachment
    collection
    var
attachmentFolder=oWeb.getFolderByServerRelativeUrl('Lists/Custom/Attachments/1');
attachmentFiles= attachmentFolder.get_files();

    //Load the client context and execute the batch
    clientContext.load(attachmentFiles);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    // Get the attachment collection and loop through it
    var attachmentEnumerator = attachmentFiles.getEnumerator();
```

```

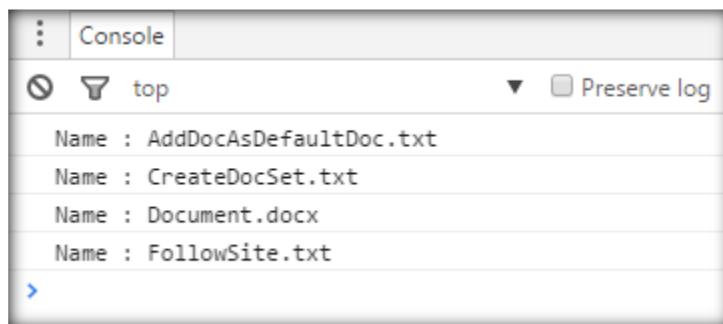
while (attachmentEnumerator.moveNext()) {
    console.log(attachmentEnumerator.get_current().get_name());
}

function QueryFailure(sender,args) {
    console.log('Request failed - '+args.get_message());
}

</script>

```

### **Output:**



### **F. Get File Properties (Author, Modified By)**

#### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### **Script:**

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getAuthor);
});

var author,modifiedBy,oFile ;

```

```

function getAuthor() {

    //Get the clientcontext,web,List and File object
    clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();
    var oList = oWeb.get_lists().getByTitle('Demo Library');
    oFile=oWeb.getFileByServerRelativeUrl('/sites/Playground/Demo
Library/Document.docx');

    //Get the modified and author values
    modifiedBy = oFile.get_modifiedBy();
    author= oFile.get_author();

    //Load the client context and execute the batch
    clientContext.load(modifiedBy);
    clientContext.load(author);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);

}

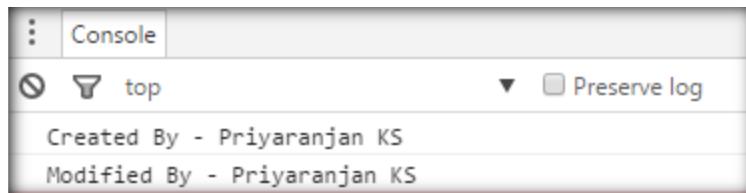
function QuerySuccess() {
    console.log("Created By - " + author.get_title());
    console.log("Modified By - " + modifiedBy.get_title());
}

function QueryFailure(sender,args) {
    alert('Request failed - '+args.get_message());
}

</script>

```

## Output:



## G. Get All Versions

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

## Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getVersions);
});

var collFiles,versionCollection,clientContext ;
function getVersions() {

    //Get the clientcontext,web and list object
    clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();
    var oList = oWeb.get_lists().getByTitle('Documents');

    //Get the file and version collection
    var oFile=oWeb.getFileByServerRelativeUrl('/sites/Playground/Demo
Library/Document.docx');
    versionCollection =oFile.get_versions();

    //Load the client context and execute the batch
    clientContext.load(versionCollection);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);

}

function QuerySuccess() {

    //Get the version collections and loop through each object
    var versionEnumerator = versionCollection.getEnumerator();

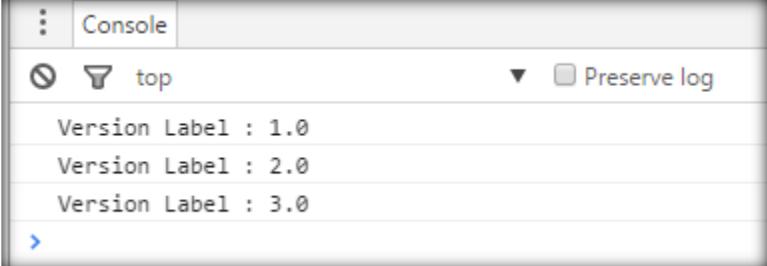
    while (versionEnumerator.moveNext()) {
        var oFile= versionEnumerator.getCurrent();
        console.log("Version Label : " + oFile.getVersionLabel());
    }
}

function QueryFailure(sender,args) {
    console.log('Request failed - '+args.getMessage());
}

</script>

```

## **Output:**



```
Console
Version Label : 1.0
Version Label : 2.0
Version Label : 3.0
```

## **H. Get All Checked Out Files**

### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### **Script:**

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getCheckedOutFiles);
});

var collFiles;
function getCheckedOutFiles() {

    //Get the clientcontext,web and list object
    clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();
    var oList = oWeb.get_lists().getByTitle('Demo Library');

    //Get the checked out file collection using CAML query
    var camlQuery = new SP.CamlQuery();
    camlQuery.set_viewXml("<View><Query><Where><IsNotNull><FieldRef
Name='CheckoutUser' /></IsNotNull></Where></Query></View>");
    collFiles = oList.getItems(camlQuery);
}
```

```

//Load the client context and execute the batch
clientContext.load(collFiles, 'Include(DisplayName)');
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);

}

function QuerySuccess() {

//Get the checked out file collection and loop through it
var filesEnumerator = collFiles.getEnumerator();

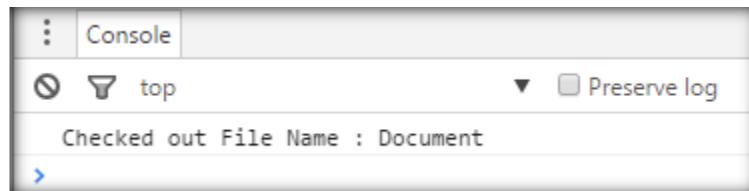
while (filesEnumerator.moveNext()) {
    var oFile= filesEnumerator.get_current();
    console.log("Checked out File Name : " + oFile.get_displayName());
}

}

function QueryFailure(sender,args) {
    console.log('Request failed - '+args.get_message());
}
</script>

```

## Output:



## I. Get Checked Out User

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getCheckedOutUser);
}

```

```

};

var oFile, checkedOutUser, newclientContext; // 
function getCheckedOutUser() {

    //Get the clientcontext and web object
    clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get the checkedout file
    oFile=oWeb.getFileByServerRelativeUrl('/sites/Playground/Demo
Library/document.docx');

    //Load the client context and execute the batch
    clientContext.load(oFile,"CheckedOutByUser","CheckOutType","Name");
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);

}

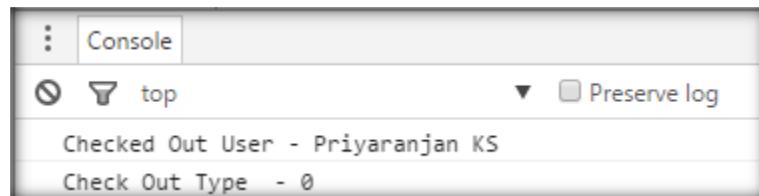
function QuerySuccess() {

    //Get the checked out user value
    checkedOutUser= oFile.get_checkedOutByUser();
    console.log("Checked Out User - " + checkedOutUser.get_title());
    console.log("Check Out Type - " + oFile.get_checkOutType());
}

function QueryFailure(sender,args) {
    console.log('Request failed - '+args.get_message());
}
</script>

```

## Output:



## J. Get Major Version

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getMajorVersion);
});

function getMajorVersion() {

    //Get client context and web
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get List and File object
    var oList = oWeb.get_lists().getByTitle('Demo Library');
    oFile=oWeb.getFileByServerRelativeUrl('/sites/Playground/Demo
Library/Document.docx');

    //Load client context and execute the batch
    clientContext.load(oFile);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);

}

function QuerySuccess() {

    //Get the major version
    var majorVersion =oFile.get_majorVersion();
    console.log("Major Version - "+ majorVersion );

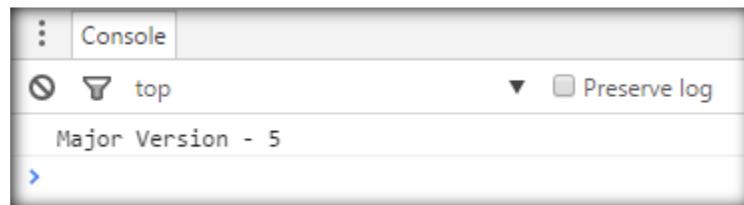
}

function QueryFailure(sender,args) {
    console.log('Request failed - '+args.get_message());
}

</script>

```

### Output:



## K. Get Minor Version

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getMinorVersion);
});

function getMinorVersion() {

    //Get client context and web
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get List and File object
    var oList = oWeb.get_lists().getByTitle('Demo Library');
    oFile=oWeb.getFileByServerRelativeUrl('/sites/Playground/Demo
Library/Document.docx');

    //Load client context and execute the batch
    clientContext.load(oFile);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);

}

function QuerySuccess() {

    //Get the minor version
    var minorVersion =oFile.get_minorVersion();
    console.log("Minor Version - "+ minorVersion );

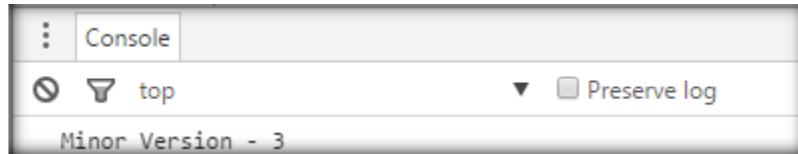
}

function QueryFailure(sender,args) {
    console.log('Request failed - '+args.get_message());
}

</script>

```

## Output:



## L. Publish File

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', publishFile);
});

var oFile;
function publishFile() {

    //Get client context and web
    clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get List and File object
    var oList = oWeb.get_lists().getByTitle('Demo Library');
    oFile=oWeb.getFileByServerRelativeUrl('/sites/Playground/Demo
Library/Document.docx');

    //Publish the file and execute the batch
    oFile.publish();
    clientContext.load(oFile);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);

}

function QuerySuccess() {
    //Get major version
}

```

```

var majorVersion = oFile.get_majorVersion();
console.log("Major Version - "+ majorVersion );

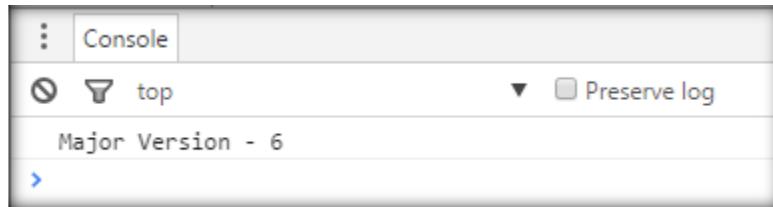
}

function QueryFailure(sender,args) {
    console.log('Request failed - '+args.get_message());
}

</script>

```

### Output:



## M. Restore File Version

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', restoreFile);
});

var oFile;
function restoreFile() {

    //Get client context and web
    clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get File object and restore a specific version
    oFile=oWeb.getFileByServerRelativeUrl('/sites/Playground/Demo
Library/Document.docx');
    oFile.get_versions().restoreByLabel("4.0");
}

```

```

//Load Client Context and execute the batch
clientContext.load(oFile);
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);

}

function QuerySuccess() {

    //Get major version
    var majorVersion =oFile.get_majorVersion();
    console.log("Version Restored . New Major Version - "+ majorVersion );

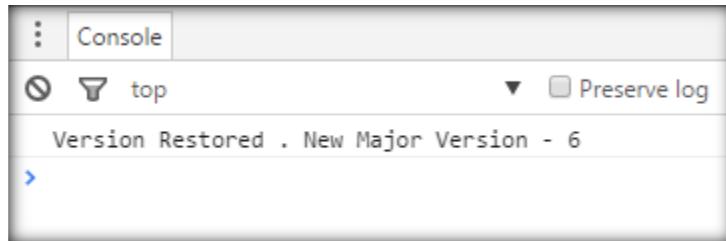
}

function QueryFailure(sender,args) {
    console.log('Request failed - '+args.get_message());
}

</script>

```

### Output:



### **N. Delete Version by Label**

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', deleteByLabel);
});


```

```

var oFile;
function deleteByLabel() {

    //Get client context and web
    clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get List and and File object
    var oList = oWeb.get_lists().getByTitle('Demo Library');
    oFile=oWeb.getFileByServerRelativeUrl('/sites/Playground/Demo
Library/Document.docx');

    //Delete file version and execute the batch
    oFile.get_versions().deleteByLabel("4.0");
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);

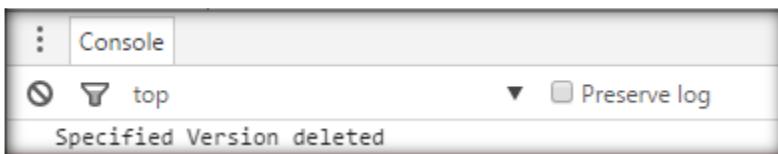
}

function QuerySuccess() {
    console.log("Specified Version deleted");
}

function QueryFailure() {
    console.log('Request failed');
}

</script>

```



5.1	5/13/2016 3:28 AM	<input type="checkbox"/> Priyaranjan KS	19.2 KB
3.0	5/13/2016 12:10 AM	<input type="checkbox"/> Priyaranjan KS	19.2 KB

## O. Delete All Versions

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', deleteAllVersions);
});

var oFile;
function deleteAllVersions() {

    //Get client context and web
    clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get file object,delete all versions and execute the batch
    oFile=oWeb.getFileByServerRelativeUrl('/sites/Playground/Demo
Library/Document.docx');
    oFile.get_versions().deleteAll();
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);

}

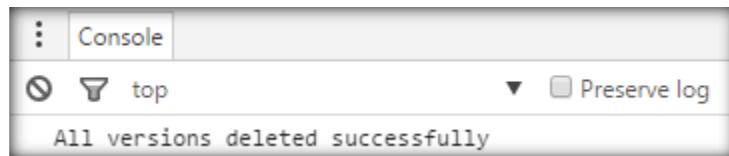
function QuerySuccess() {
    console.log("All versions deleted successfully");
}

function QueryFailure(sender,args) {
    console.log('Request failed - '+args.get_message());
}

</script>

```

### Output:



### P. Discard Checkout

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', discardCheckOut);
});

function discardCheckOut() {

    //Get client context and web object
    clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get the file object and discard check out
    var oFile=oWeb.getFileByServerRelativeUrl('/sites/Playground/Demo
Library/Document.docx');
    oFile.undoCheckOut();

    //Load Client Context and execute the batch
    clientContext.load(oFile);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);

}

function QuerySuccess() {
    console.log("Check Out Discarded.");
}

function QueryFailure(sender,args) {
    console.log('Request failed - '+args.get_message());
}

</script>

```

### Output:



## Q. Get All Files from Library

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getFiles);
});

var collFiles,clientContext ;
function getFiles() {

    //Get the clientcontext,web and List object
    clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();
    var oList = oWeb.get_lists().getByTitle('Demo Library');

    //Use Caml to get all the files from library
    var camlQuery = new SP.CamlQuery();
    camlQuery.set_viewXml('<View
Scope=\'RecursiveAll\'><Query><Where><Eq><FieldRef Name=\'FSObjType\' /><Value
Type=\'Integer\'>0</Value></Eq></Where></Query></View>');
    collFiles = oList.getItems(camlQuery);

    //Load clientcontext and execute the batch
    clientContext.load(collFiles);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);

}

function QuerySuccess() {

    //Get the file collection and loop through it
    var filesItemEnumerator = collFiles.getEnumerator();

    while (filesItemEnumerator.moveNext()) {
        var oFile= filesItemEnumerator.get_current().get_item('FileLeafRef');
        console.log("Name : " + oFile );
    }
}

```

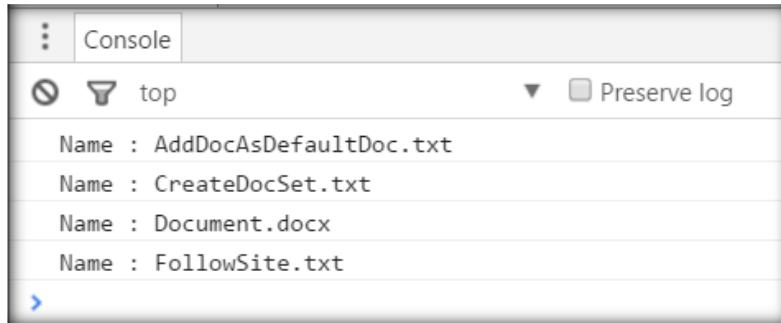
```

function QueryFailure(sender,args) {
    console.log('Request failed - ' + args.get_message());
}

</script>

```

### **Output:**



### **R. Create Attachment Folder**

#### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### **Script:**

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

var fileInput ;
$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', checkAttachmentFolder);
});
var clientContext,oListItem,oWeb;
function checkAttachmentFolder() {

    //Get Client Context and Web object.
    clientContext = new SP.ClientContext();
    oWeb = clientContext.get_web();

    //Get list and Attachment folder where the attachment of a particular list
item is stored.
}

```

```

var oList = oWeb.get_lists().getByTitle('NewList');
oListItem = oList.getItemById(1);

//Load client context and execute the batch
clientContext.load(oWeb);
clientContext.load(oListItem);
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    if(!oListItem.get_fieldValues()['Attachments']) {
        //Get attachment folder and attachment folder url
        var attachmentsFolder =
oWeb.getFolderByServerRelativeUrl('Lists/NewList/Attachments');
        var
attachmentsFolderURL=oWeb.get_serverRelativeUrl()+'Lists/NewList/Attachments';

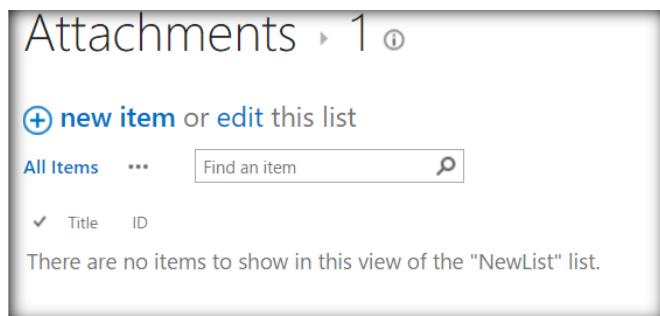
        //Create Attachment Folder
        attachmentsFolder = attachmentsFolder.get_folders().add('_' + '1');
        attachmentsFolder.moveTo(attachmentsFolderURL + '/' + '1');

        //Load Client Context and execute the batch
        clientContext.load(attachmentsFolder);
        clientContext.executeQueryAsync(SecondQuerySuccess, SecondQueryFailure);
    }
}

function QueryFailure(sender,args) {
    console.log('Request failed with error message - '+ args.get_message()+' .
Stack Trace - '+ args.get_stackTrace());
}
function SecondQuerySuccess(sender,args) {
    console.log('Attachment folder has been created.');
}
function SecondQueryFailure(sender,args) {
    console.log('Request failed with error message - '+ args.get_message()+' .
Stack Trace - '+ args.get_stackTrace());
}
</script>

```

## Output:



Attachments → 1 ⓘ

**⊕ new item or edit this list**

All Items    ...    Find an item

✓ Title    ID

There are no items to show in this view of the "NewList" list.

## S. Add File to Document Library

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<html>
<head>

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

var fileInput ;
$(document).ready(function() {
    fileInput = $("#getFile");
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', registerClick);
});

function registerClick()
{
    //Register File Upload Click Event
    $("#addFileButton").click(readFile);
}

var arrayBuffer ;
function readFile()
{
    //Get File Input Control and read th file name
    var element = document.getElementById("getFile");
    var file = element.files[0];
    var parts = element.value.split("\\\\");
    var fileName = parts[parts.length - 1];

    //Read File contents using file reader
    var reader = new FileReader();
    reader.onload = function (e) {
        uploadFile(e.target.result, fileName);
    }
    reader.onerror = function (e) {
        alert(e.target.error);
    }
    reader.readAsArrayBuffer(file);
}

var attachmentFiles;

```

```

function uploadFile(arrayBuffer,fileName) {
    //Get Client Context,Web and List object.
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();
    var oList = oWeb.get_lists().getByTitle('Documents');

    //Convert the file contents into base64 data
    var bytes = new Uint8Array(arrayBuffer);
    var i, length, out = '';
    for (i = 0, length = bytes.length; i < length; i += 1) {
        out += String.fromCharCode(bytes[i]);
    }
    var base64 = btoa(out);

    //Create FileCreationInformation object using the read file data
    var createInfo = new SP.FileCreationInformation();
    createInfo.set_content(base64);
    createInfo.set_url(fileName);

    //Add the file to the library
    var uploadedDocument = oList.get_rootFolder().get_files().add(createInfo)

    //Load client context and execute the batch
    clientContext.load(uploadedDocument);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

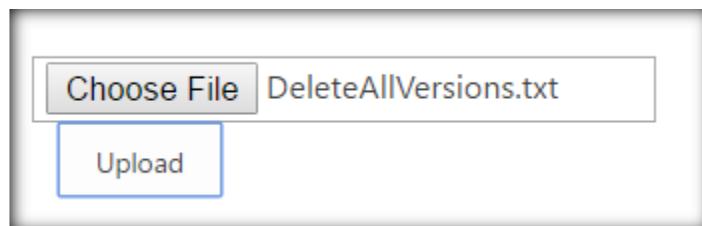
function QuerySuccess() {
    console.log('File Uploaded Successfully.');
}

function QueryFailure(sender,args) {
    console.log('Request failed with error message - '+ args.get_message() + ' .
Stack Trace - ' + args.get_stackTrace());
}

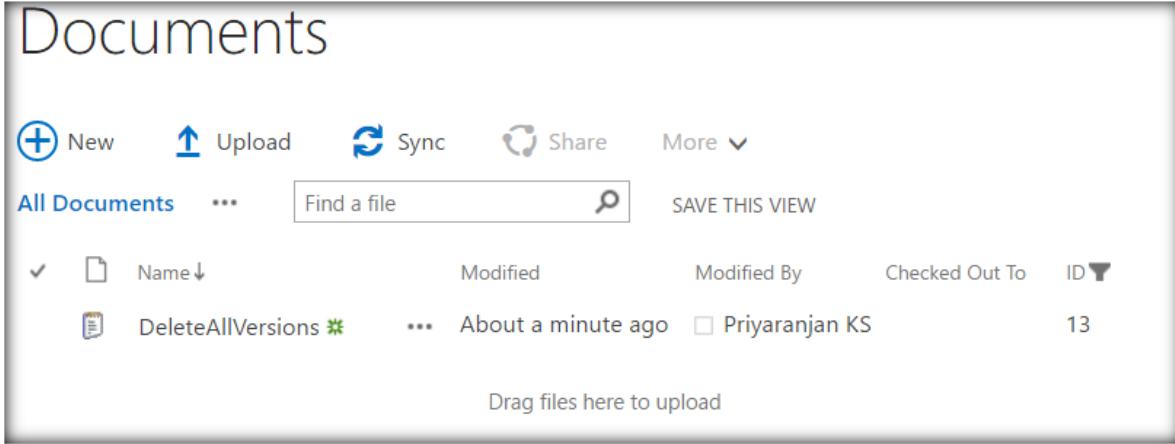
</script>
</head>
<body>
<input id="getFile" type="file"/><br />
    <input id="addFileButton" type="button" value="Upload" />
</body>
<html>

```

## Output:



Select the file and click on Upload. The new file will be uploaded to the document library, as shown below.



## T. Add File to List as Attachment

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<html>
<head>

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

var fileInput ;
$(document).ready(function() {
    fileInput = $("#getFile");
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', registerClick);
});

function registerClick()
{
```

```

//Register File Upload Click Event
$("#addFileButton").click(readFile);

}

var arrayBuffer ;
function readFile()
{
    //Get File Input Control and read th file name
    var element = document.getElementById("getFile");
    var file = element.files[0];
    var parts = element.value.split("\\\\");
    var fileName = parts[parts.length - 1];

    //Read File contents using file reader
    var reader = new FileReader();
    reader.onload = function (e) {
        uploadFile(e.target.result, fileName);
    }
    reader.onerror = function (e) {
        alert(e.target.error);
    }
    reader.readAsArrayBuffer(file);
}

var attachmentFiles,clientContext,createInfo;
function uploadFile(arrayBuffer,fileName) {

    //Get Client Context and Web object.
    clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get list and Attachment folder where the attachment of a particular
    list item is stored.
    var oList = oWeb.get_lists().getByTitle('NewList');
    var
attachmentFolder=oWeb.getFolderByServerRelativeUrl('Lists/NewList/Attachments
/1');

    //Convert the file contents into base64 data
    var bytes = new Uint8Array(arrayBuffer);
    var i, length, out = '';
    for (i = 0, length = bytes.length; i < length; i += 1) {
        out += String.fromCharCode(bytes[i]);
    }
    var base64 = btoa(out);

    //Create FileCreationInformation object using the read file data
    createInfo = new SP.FileCreationInformation();
    createInfo.set_content(base64);
    createInfo.set_url(fileName);

    //Add the file to the list item
    attachmentFiles = attachmentFolder.get_files().add(createInfo);
}

```

```

//Load client context and execute the batch
clientContext.load(oList);
clientContext.load(attachmentFiles);
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

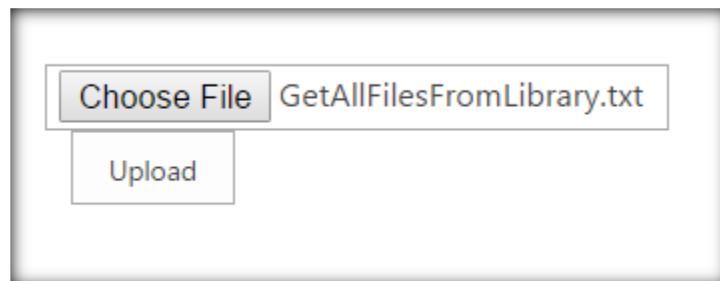
function QuerySuccess() {
    console.log("Attachment added successfully ");
}

function QueryFailure(sender,args) {
    console.log('Request failed with error message - '+ args.get_message()+' . Stack Trace - '+ args.get_stackTrace());
}

</script>
</head>
<body>
<input id="getFile" type="file"/><br />
    <input id="addFileButton" type="button" value="Upload" />
</body>
<html>

```

### Output:



Title * <input type="text" value="TestItem"/>	Attachments <a href="#">GetAllFilesFromLibrary.txt</a> <input type="button" value="Delete"/>
Created at 5/28/2016 10:39 PM by <input type="checkbox"/> Priyaranjan KS Last modified at 5/28/2016 11:29 PM by <input type="checkbox"/> Priyaranjan KS	
<input type="button" value="Save"/> <input type="button" value="Cancel"/>	

## VI. Working with Folder

### A. Break Inheritance

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', breakInheritance);
});

var oFolder, clientContext;
function breakInheritance() {

    //Get client context.web and folder object
    clientContext = SP.ClientContext.get_current();
    var web = clientContext.get_web();
    oFolder = web.getFolderByServerRelativeUrl("Demo Library/Demo Folder");

    //Load Client Context and execute the batch
    clientContext.load(oFolder, 'ListItemAllFields');
    clientContext.executeQueryAsync(Success, Failure);

}

function Success()
{
    //Get Folder instance and break inheritance
    var folderInstance = oFolder.get_listItemAllFields();
    folderInstance.breakRoleInheritance(false, false);

    //Execute the batch
    clientContext.executeQueryAsync(OnSuccess, onFailure);
}

function Failure(sender, args) {
    console.log("Request failed with error message - " + args.get_message());
}

```

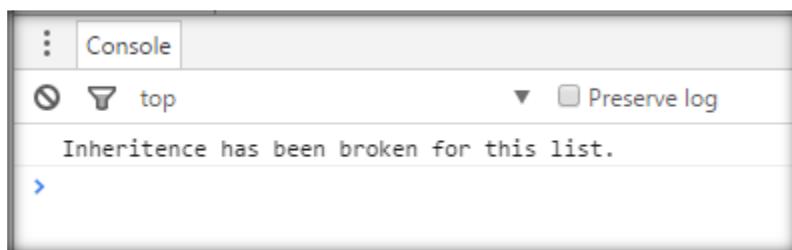
```

function OnSuccess() {
    console.log("Inheritance has been broken for this list. ");
}

function onFailure(sender, args) {
    console.log('Request failed. ' + args.get_message() + '\n' +
args.get_stackTrace());
}
</script>

```

### **Output:**



	Type	Permission Levels
<input type="checkbox"/> Priyaranjan KS	User	Full Control

### **B. Create Folder**

#### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### **Script:**

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', createFolder);
});

var oListItem;
function createFolder() {

    //Get Client Context , web and list object
    var clientContext = new SP.ClientContext();
    var oWebsite = clientContext.get_web();
    var oList = oWebsite.get_lists().getByTitle('Demo Library');

    //Create Folder using SP.FileSystemObjectType.folder type
    var oListItemCreationInformation= new SP.ListItemCreationInformation();

    oListItemCreationInformation.set_underlyingObjectType(SP.FileSystemObjectType.folder);
    oListItemCreationInformation.set_leafName("NewFolder");
    oListItem= oList.addItem(oListItemCreationInformation);
    oListItem.update();

    //Load Client Context and execute the batch
    clientContext.load(oListItem);
    clientContext.executeQueryAsync(Success, Failure);
}

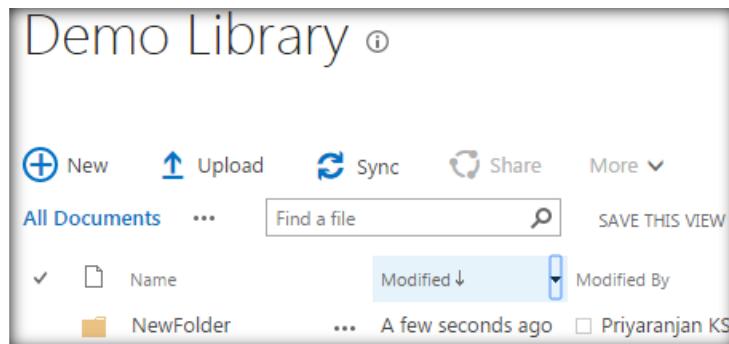
function Success() {
    console.log('Title: ' + oListItem.get_item('FileLeafRef'));
}

function Failure(sender,args) {
    console.log('Request failed with error message - ' + args.get_message()+' .
Stack Trace - '+ args.get_stackTrace());
}

</script>

```

## Output:



## C. Edit Folder Name

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', editFolderName);
});

var oListItem, collListItem, clientContext;
function editFolderName() {

    //Get client context,web and list object
    clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();
    var oList = oWeb.get_lists().getByTitle('Demo Library');

    //Get the Folder collection using caml query
    var camlQuery = new SP.CamlQuery();
    camlQuery.set_viewXml("<View><Query><Where><And><Eq><FieldRef
Name='FileLeafRef' /><Value Type='Text'>NewFolder</Value></Eq><Eq><FieldRef
Name='FSObjType' /><Value
Type='Text'>1</Value></Eq></And></Where></Query></View>");
    collListItem = oList.getItems(camlQuery)

    //Load client context and execute the batch
    clientContext.load(collListItem);
    clientContext.executeQueryAsync(Success, Failure);
}

function Success() {

    //Get folder collection and loop through it
    var listItemEnumerator = collListItem.getEnumerator();
    var oListItem;
    while (listItemEnumerator.moveNext()) {
        //get the list item and change the folder name
        oListItem = listItemEnumerator.get_current();
        oListItem.set_item('FileLeafRef', 'ChangedFolderName');
        oListItem.update();
    }
}

```

```

//Load the client context and execute the batch
clientContext.load(oListItem);
clientContext.executeQueryAsync(CallSuccess, CallFailure);

}

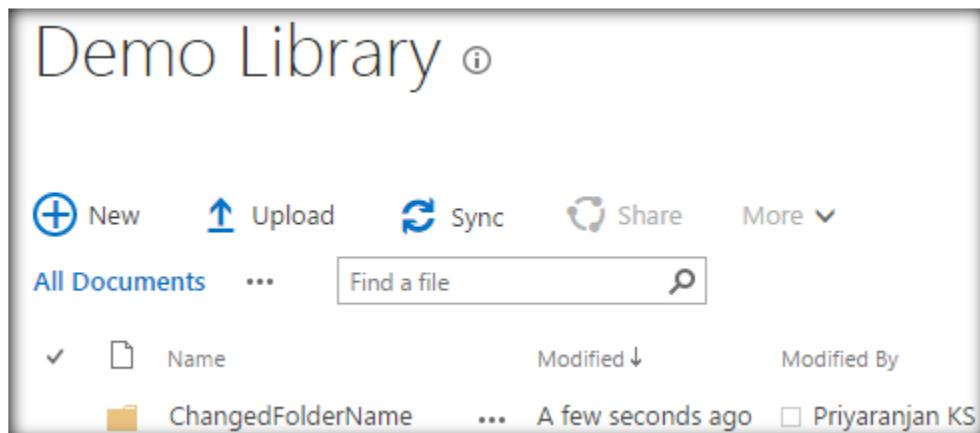
function Failure(sender,args) {
    console.log('Request failed with error message - '+ args.get_message()+' .
Stack Trace - '+ args.get_stackTrace());
}

function CallSuccess() {
    console.log("Folder renamed successfully.");
}
function CallFailure() {
    console.log('Request failed with error message - '+ args.get_message()+' .
Stack Trace - '+ args.get_stackTrace());
}

</script>

```

### Output:



### D. Get All Folders

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

## Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getAllFolders);
});

var oListItem, collListItem, clientContext;
function getAllFolders() {

    //Get client context,web and list object
    clientContext = new SP.ClientContext();
    var oWeb= clientContext.get_web();
    var oList = oWeb.get_lists().getByTitle('Demo Library');

    //use caml query to get the folder collection
    var camlQuery = new SP.CamlQuery();
    camlQuery.set_viewXml("<View Scope='RecursiveAll'><Query><Where><Eq><FieldRef
Name='FSObjType' /><Value Type='Text'>1</Value></Eq></Where></Query></View>");
    camlQuery.set_folderServerRelativeUrl('/Sites/Playground/Demo
Library/NewFolder');
    collListItem = oList.getItems(camlQuery)

    //Load the client context and execute the batch
    clientContext.load(collListItem);
    clientContext.executeQueryAsync(Success, Failure);
}

function Success() {

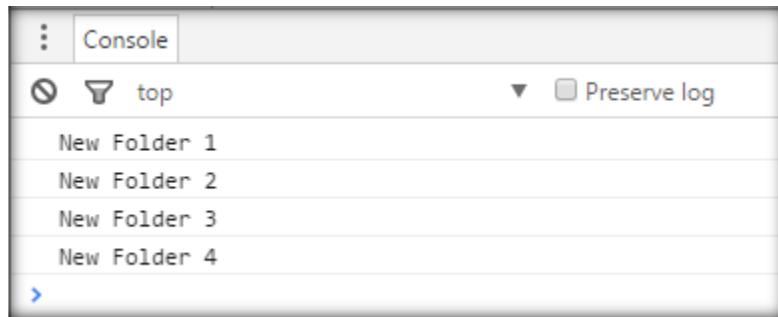
    //get the folder collection and loop through it
    var listItemEnumerator = collListItem.getEnumerator();
    var oListItem;
    while (listItemEnumerator.moveNext()) {
        oListItem = listItemEnumerator.get_current();
        console.log(oListItem.get_item('FileLeafRef'));
    }
}

function Failure(sender,args) {
    console.log('Request failed with error message - ' + args.get_message() + '
Stack Trace - ' + args.get_stackTrace());
}

</script>

```

## Output:



## E. Get Files from Folder

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getFilesFromFolder);
});

var files;
function getFilesFromFolder() {

    //Get client context and web object
    var clientContext = SP.ClientContext.get_current();
    var web = clientContext.get_web();

    //Get folder and files
    folder = web.getFolderByServerRelativeUrl("Demo Library/NewFolder");
    files = folder.get_files();

    //Load client context and execute the batch
    clientContext.load(files);
    clientContext.executeQueryAsync(Success, Failure);
}

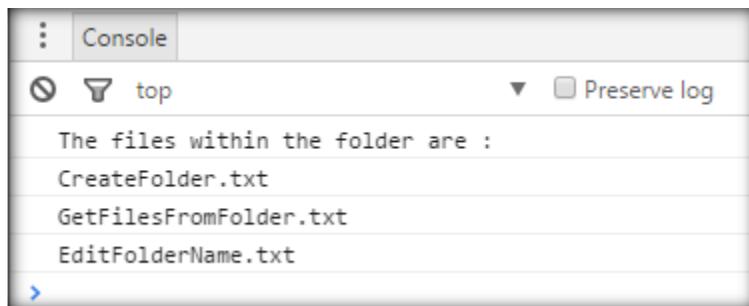
function Success()
{
}

```

```
//Get files collection and loop through it
var listItemEnumerator = files.getEnumerator();
console.log("The files within the folder are :\n");
while (listItemEnumerator.moveNext()) {
    var fileName= listItemEnumerator.get_current().get_name();
    console.log(fileName);
}

function Failure(sender, args) {
    console.log("Request failed with error message - " + args.get_message());
}
</script>
```

### Output:



### F. Get Folder Properties

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getFolderProperties);
});

var oFolder;
function getFolderProperties() {
    //Get client context and web object
```

```

var clientContext = new SP.ClientContext();
var oWebsite = clientContext.get_web();

//Get List and folder object
var oList = oWebsite.get_lists().getByTitle('Demo Library');
oFolder = oWebsite.getFolderByServerRelativeUrl('/sites/playground/Demo
Library/NewFolder');

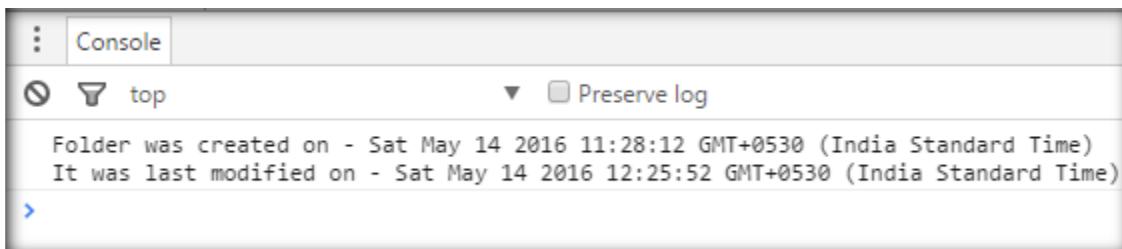
//Get the client context and execute the batch
clientContext.load(oFolder);
clientContext.executeQueryAsync(Success, Failure);
}

function Success() {
    console.log("Folder was created on - "+ oFolder.get_timeCreated()+'\n' +"It
was last modified on - " + oFolder.get_timeLastModified());
}

function Failure(sender,args) {
    console.log('Request failed with error message - '+ args.get_message()+' .
Stack Trace - '+ args.get_stackTrace());
}
</script>

```

## Output:



The screenshot shows the browser's developer tools Console tab. It has a header with 'Console' and other buttons like 'Preserve log'. Below the header, there are controls for 'clear' and 'top'. The main area displays two lines of log output in green text:

```

Folder was created on - Sat May 14 2016 11:28:12 GMT+0530 (India Standard Time)
It was last modified on - Sat May 14 2016 12:25:52 GMT+0530 (India Standard Time)

```

## VII. Working with Group

### A. Add Role

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', addRole);
});

var oSubWebs;
function addRole() {

    //Get client context and web object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();
    //Get the group to be assigned the role
    var oGroup= oWeb.get_siteGroups().getByName("HR Group");

    //Get Role definition and role definition binding collection
    var oRoleDef = oWeb.get_roleDefinitions().getByName('Contribute');
    var oBindingColl =
SP.RoleDefinitionBindingCollection.newObject(clientContext);

    // Add the role to the collection.
    oBindingColl.add(oRoleDef);

    // Get the RoleAssignmentCollection for the target web.
    var oCurrentRoleAssignments = oWeb.get_roleAssignments();

    // assign the group to the new RoleDefinitionBindingCollection.
    var roleAssignmentContribute = oCurrentRoleAssignments.add(oGroup,
oBindingColl);

    //Load the client context and execute the batch
    clientContext.load(oGroup);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

```

```

}

function QuerySuccess() {
    console.log("Role definition added Successfully.");
}

function QueryFailure(sender,args) {
    console.log('Request failed - '+args.get_message());
}

</script>

```

### Output:

Updated Web Site

## View Site Collection Permissions: HR Group



Use this page to view the permission assignments that this SharePoint group has in this site collection. In addition to the listed URLs, this group has access to any sites, lists, or items that inherit permissions from these URLs.

URL	Permission Level
<a href="https://ctsplayground.sharepoint.com/sites/Playground">https://ctsplayground.sharepoint.com/sites/Playground</a>	Design, Contribute

### B. Add User

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', addUser);
});

var oGroupUsers;
function addUser() {

```

```

//Get client context and web
var clientContext = new SP.ClientContext();
var oWeb = clientContext.get_web();

//Get group collection and specific group
var groupCollection = oWeb.get_siteGroups();
var oGroup = groupCollection.getByName("HR Group");

//Get user object . In SharePoint 2016 use 'Domain\\UserName' as the format
//In SharePoint Online the user id has to be in the format
i:0#.f|membership|username@tenant.onmicrosoft.com
var oUser =
clientContext.get_web().ensureUser('i:0#.f|membership|Priyaranjan@CTSPPlayGround.on
microsoft.com');

//Get all SP Users in SP Group
var oUserCollection = oGroup.get_users();

//Add User to Group
var addedUser = oUserCollection.addUser(oUser);

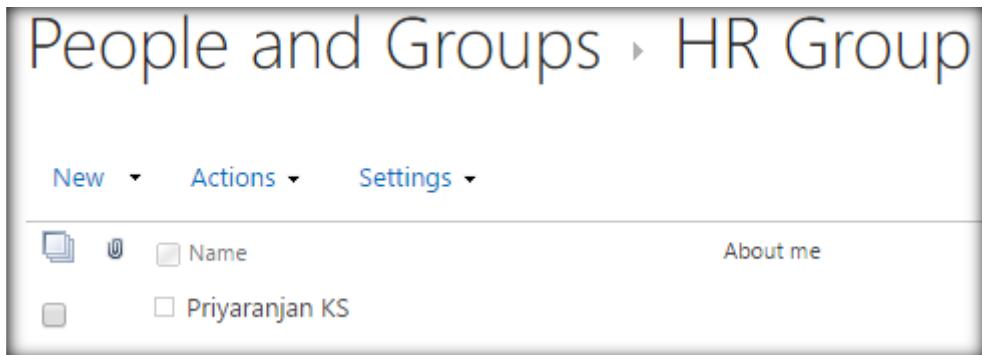
//Load client context and execute the batch
clientContext.load(addedUser);
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    console.log("User added Successfully.");
}

function QueryFailure(sender,args) {
    console.log('Request failed - '+args.get_message());
}
</script>

```

### Output:



The screenshot shows the SharePoint 'People and Groups' page for the 'HR Group'. At the top, there's a navigation bar with 'New', 'Actions', and 'Settings'. Below the navigation, there's a table-like structure with columns for icons, names, and 'About me'. The first row shows a user with a name field containing 'Priyaranjan KS'. The second row shows a user with a name field containing 'About me'. The third row shows a user with a name field containing 'Priyaranjan KS', which is highlighted in blue, indicating it's the currently selected item.

## C. Edit Membership

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', editMembership);
});

function editMembership() {

    //Get client context and web
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get group collection and specific group
    var groupCollection = oWeb.get_siteGroups();
    var oGroup = groupCollection.getByName("HR Group");

    //Allow members to edit membership
    oGroup.set_allowMembersEditMembership(true);
    oGroup.set_onlyAllowMembersViewMembership(false);
    oGroup.update();

    //Load client context and execute the batch
    clientContext.load(oGroup);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    console.log("Member can now edit membership.");
}

function QueryFailure(sender,args) {
    alert('Request failed - '+args.get_message());
}

</script>

```

## **Output:**

<b>Group Settings</b> Specify who has permission to see the list of group members and who has permission to add and remove members from the group.	Who can view the membership of the group? <input type="radio"/> Group Members <input checked="" type="radio"/> Everyone
Who can edit the membership of the group? <input type="radio"/> Group Owner <input checked="" type="radio"/> Group Members	

## **D. Get User's Group Membership Properties**

### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### **Script:**

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
  SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getUsersEditPermissions);
});

var oGroup;
function getUsersEditPermissions() {

  //Get client context and web
  var clientContext = new SP.ClientContext();
  var oWeb = clientContext.get_web();

  //Get group collection and specific group
  var groupCollection = oWeb.get_siteGroups();
  oGroup = groupCollection.getByName("HR Group");

  //Load client context and execute the batch
  clientContext.load(oGroup, "CanCurrentUserEditMembership", "CanCurrentUserManageGroup", "CanCurrentUserViewMembership");
  clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
  console.log("Can Current User Edit Membership - "+oGroup.get_canCurrentUserEditMembership());
  console.log("Can Current User Manage Group - "

```

```

"+oGroup.get_canCurrentUserManageGroup());
    console.log("Can Current User View Membership -
"+oGroup.get_canCurrentUserViewMembership());

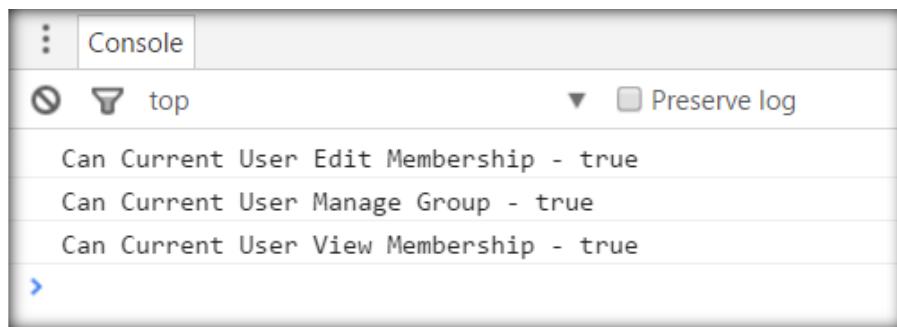
}

function QueryFailure(sender,args) {
    alert('Request failed -      '+ args.get_message());
}

</script>

```

### **Output:**



The screenshot shows the SharePoint browser console with the 'Console' tab selected. It displays three log entries:

- Can Current User Edit Membership - true
- Can Current User Manage Group - true
- Can Current User View Membership - true

### **E. Get User Group Properties**

#### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### **Script:**

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getGroupProperties);
});

var oGroup;
function getGroupProperties() {
    //Get client context and web object

```

```

var clientContext = new SP.ClientContext();
var oWeb = clientContext.get_web();

//Get group collection and specific group
var groupCollection = oWeb.get_siteGroups();
oGroup = groupCollection.getByName("HR Group");

//Load client context and execute the batch
clientContext.load(oGroup,"AllowMembersEditMembership","AllowRequestToJoinLeave",
"AutoAcceptRequestToJoinLeave","OnlyAllowMembersViewMembership","RequestToJoinLeave
EmailSetting");
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

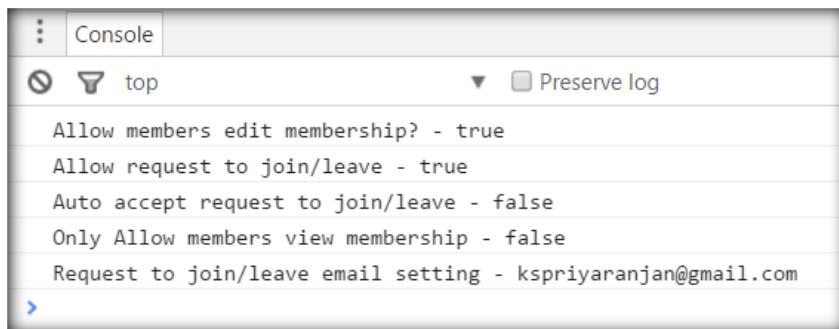
function QuerySuccess() {
    console.log("Allow members edit membership? - 
"+oGroup.get_allowMembersEditMembership());
    console.log("Allow request to join/leave - 
"+oGroup.get_allowRequestToJoinLeave());
    console.log("Auto accept request to join/leave - 
"+oGroup.get_autoAcceptRequestToJoinLeave());
    console.log("Only Allow members view membership - 
"+oGroup.get_onlyAllowMembersViewMembership());
    console.log("Request to join/leave email setting - 
"+oGroup.get_requestToJoinLeaveEmailSetting());
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## Output:



The screenshot shows a browser's developer tools Console tab. The output window displays the following log messages:

```

Allow members edit membership? - true
Allow request to join/leave - true
Auto accept request to join/leave - false
Only Allow members view membership - false
Request to join/leave email setting - kspriyaranjan@gmail.com

```

## F. Delete Group

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', deleteGroup);
});

var clientContext,oGroupCollection,oGroup;
function deleteGroup() {

    //Get client context and web
    clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get group collection and specific group
    oGroupCollection = oWeb.get_siteGroups();
    oGroup = oGroupCollection.getByName("Admin Group");

    //Load the Client Context and execute the batch
    clientContext.load(oGroupCollection);
    clientContext.load(oGroup);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    //Remove the group from the collection
    oGroupCollection.removeByLoginName(oGroup.get_loginName());
    clientContext.executeQueryAsynchronous(OnQuerySuccess, OnQueryFailure);
}

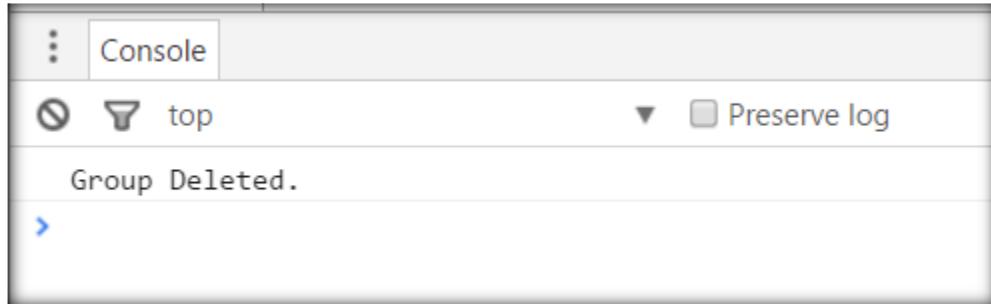
function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

function OnQuerySuccess() {
    console.log("Group Deleted.");
}
function OnQueryFailure(sender,args) {
    console.log('Request failed - '+args.get_message());
}

</script>

```

### Output:



### G. Get Site Groups

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getSiteGroups);
});

var groupCollection;
function getSiteGroups() {

    //Get client context, web and group collection
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();
    groupCollection = oWeb.get_siteGroups();

    //Get client context and execute the batch
    clientContext.load(groupCollection);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

    //Get group collection and iterate through it
    var groupEnumerator = groupCollection.getEnumerator();
    console.log("The site groups available are : \n");
}
```

```

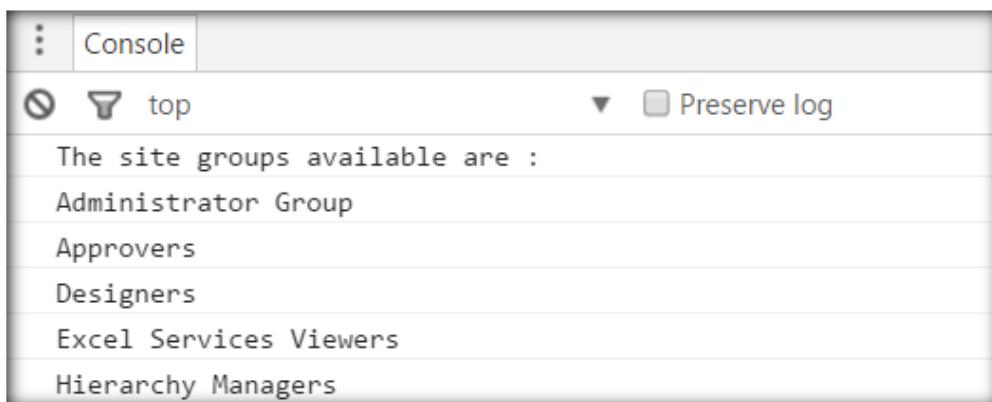
while (groupEnumerator.moveNext()) {
    var group =groupEnumerator.get_current();
    console.log(group.get_title()+"\n");
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

### Output:



```

Console
top
Preserve log
The site groups available are :
Administrator Group
Approvers
Designers
Excel Services Viewers
Hierarchy Managers

```

## H. Get Group Users

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getGroupUsers);
});

var oGroupUsers;

```

```

function getGroupUsers () {
    //Get client context and web object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get group collection and specific group
    var groupCollection = oWeb.get_siteGroups();
    oGroup = groupCollection.getByName("HR Group");

    //Get the users collection
    oGroupUsers = oGroup.get_users();

    //Load the client context and execute the batch
    clientContext.load(oGroupUsers);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

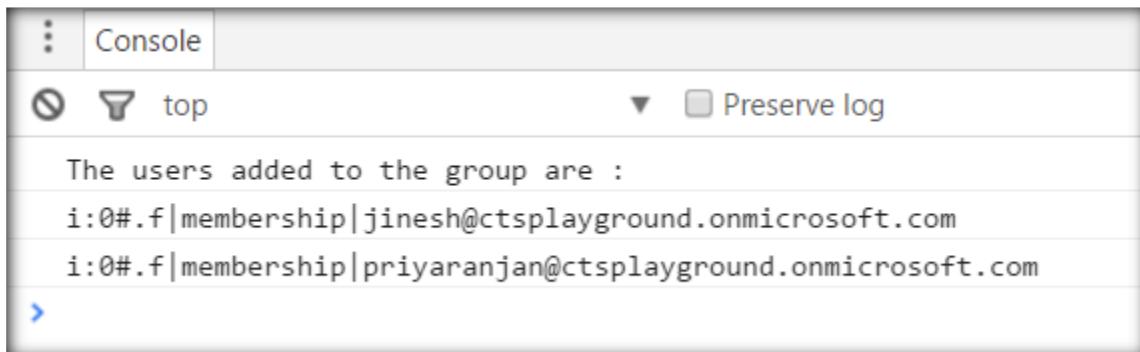
function QuerySuccess() {
    //Get the users collection and loop through it
    var groupUsersEnumerator = oGroupUsers.getEnumerator();
    console.log("The users added to the group are :\n");
    while (groupUsersEnumerator.moveNext()) {
        var groupUser = groupUsersEnumerator.get_current();
        console.log(groupUser.get_loginName());
    }
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## Output:



The screenshot shows a browser's developer tools console window. The title bar says "Console". Below it are filter and preserve log buttons. The main area displays the following text:

```

The users added to the group are :
i:0#.f|membership|jinesh@ctsplayground.onmicrosoft.com
i:0#.f|membership|priyaranjan@ctsplayground.onmicrosoft.com

```

## I. Remove User from Group

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', removeUser);
});

var oGroupUsers;
function removeUser() {
    //Get client context and web object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get group collection and specific group
    var groupCollection = oWeb.get_siteGroups();
    oGroup = groupCollection.getByName("HR Group");

    //Ensure SP User .In SP 2016 User name has to be in the format Domain\Login
    var oUser =
oWeb.ensureUser("i:0#.f|membership|Priyaranjan@CTSPlayGround.onmicrosoft.com");
    oGroup.get_users().remove(oUser);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    console.log("User removed from group.");
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## Output:



The screenshot shows a browser's developer tools console. The title bar says "Console". Below it are filter buttons for "Console" (selected), "Script", and "Network". There are also buttons for "Preserve log" and "Clear log". The main area of the console contains the text "User removed from group.".

## J. Set Approver Email

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', setApproverEmail);
});

var oGroupUsers;
function setApproverEmail() {

    //Get client context and web object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get group collection and specific group
    var groupCollection = oWeb.get_siteGroups();
    oGroup = groupCollection.getByName("HR Group");

    //Assign the approver email
    oGroup.set_requestToJoinLeaveEmailSetting("priyaranjan.ks@gmail.com");
    oGroup.update();

    //Load the client context and execute the batch
    clientContext.load(oGroup);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}


```

```

function QuerySuccess() {
    console.log("Approver email set.");
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

### Output:

Membership Requests

Specify whether to allow users to request membership in this group and allow users to request to leave the group. All requests will be sent to the e-mail address specified. If auto-accept is enabled, users will automatically be added or removed when they make a request.

**Caution:** If you select yes for the Auto-accept requests option, any user requesting access to this group will automatically be added as a member of the group and receive the permission levels associated with the group.

Allow requests to join/leave this group?

Yes       No

Auto-accept requests?

Yes       No

Send membership requests to the following e-mail address:

## K. Set User as Owner of Group

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', setAsOwner);
});

var oGroupUsers;
function setAsOwner() {
    //Get client context and web object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get group collection and specific group
    var groupCollection = oWeb.get_siteGroups();
    oGroup = groupCollection.getByName("HR Group");
}

```

```

//Set as owner
var oUser =
clientContext.get_web().ensureUser('i:0#.f|membership|Priyaranjan@CTSPlayGround.on
microsoft.com');
oGroup.set_owner(oUser);
oGroup.update();

//Load the client context and execute the batch
clientContext.load(oGroup);
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    console.log("Group Owner set.");
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}
</script>

```

### **Output:**

#### Owner

The owner can change anything about the group such as adding and removing members or deleting the group. Only one user or group can be the owner.

#### Group owner:

## L. Set a Group as Owner of Another Group

### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### **Script:**

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', setGroupAsOwner);
});

```

```

var oGroupUsers;
function setGroupAsOwner() {

    //Get client context and web object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get group collection and specific group
    var groupCollection = oWeb.get_siteGroups();
    oGroup = groupCollection.getByName("HR Group");
    //Get the owner group
    var oOwnerGroup = groupCollection.getByName("Approvers");

    //Set the owner group
    oGroup.set_owner(oOwnerGroup);
    oGroup.update();
    //Load the client context and execute the batch
    clientContext.load(oGroup);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    console.log("Group Owner set.");
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## Output:

<p><b>Owner</b></p> <p>The owner can change anything about the group such as adding and removing members or deleting the group. Only one user or group can be the owner.</p>	<p><b>Group owner:</b></p> <input type="text" value="Approvers x"/>
--	---

## VIII. Working with List

### A. Allow Attachments

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', allowAttachment);
});

var oList, clientContext;

function allowAttachment() {

    //Get client context, web and list object
    clientContext = new SP.ClientContext.get_current();
    var oWeb = clientContext.get_web();
    oList= oWeb.get_lists().getByTitle('DemoList');

    //Load the client context and execute the batch
    clientContext.load(oList);
    clientContext.executeQueryAsync(Success,Failure);
}

function Success() {

    //Allow attachment to list
    oList.set_enableAttachments('true');
    oList.update();

    //Load the client context and execute the batch
    clientContext.load(oList);
    clientContext.executeQueryAsync(OnSuccess,OnFailure);
}

function Failure(sender, args) {
    console.log('Request failed. ' + args.get_message() + '\n' +
}

```

```

args.get_stackTrace();
}

function OnSuccess() {
    console.log("Attachments have been enabled for this list");
}

function onFailure(sender, args) {
    console.log('Request failed. ' + args.get_message() + '\n' +
args.get_stackTrace());
}
</script>

```

### Output:

**Attachments**

Specify whether users can attach files to items in this list.

Attachments to list items are:

Enabled

Disabled

## B. Allow Folder Creation

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', allowFolderCreation);
});

var oList, clientContext;

function allowFolderCreation() {
    //Get client context , web and list object

```

```

clientContext = new SP.ClientContext.get_current();
var oWeb = clientContext.get_web();
oList= oWeb.get_lists().getByTitle('DemoList');

//Load the client context and execute the batch
clientContext.load(oList);
clientContext.executeQueryAsync(Success,Failure);
}

function Success() {

    //Enable folder creation
    oList.set_enableFolderCreation('true');
    oList.update();

    //Load the client context and execute the batch
    clientContext.load(oList);
    clientContext.executeQueryAsync(OnSuccess,OnFailure);
}

function Failure(sender, args) {
    console.log('Request failed. ' + args.get_message() + '\n' +
args.get_stackTrace());
}

function OnSuccess() {
    console.log("Folder creation has been enabled for this list");
}

function OnFailure(sender, args) {
    console.log('Request failed. ' + args.get_message() + '\n' +
args.get_stackTrace());
}
</script>

```

## Output:

### Folders

Specify whether the "New Folder" command is available. Changing this setting does not affect existing folders.

Make "New Folder" command available?

Yes     No

## C. Allow Management of Content Types

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', allowManagementCT);
});

var oList, clientContext;
function allowManagementCT() {

    //Get the client context ,web and list object
    clientContext = new SP.ClientContext.get_current();
    var oWeb = clientContext.get_web();
    oList= oWeb.get_lists().getByTitle('DemoList');

    // Load the client context and execute the batch
    clientContext.load(oList );
    clientContext.executeQueryAsync(Success,Failure);
}

function Success(){
    //Enable content type
    oList.set_contentTypesEnabled('true');
    oList.update();

    // Load the client context and execute the batch
    clientContext.load(oList );
    clientContext.executeQueryAsync(OnSuccess,OnFailure);
}

function Failure(sender, args) {
    console.log('Request failed. ' + args.get_message() + '\n' +
args.get_stackTrace());
}

function OnSuccess() {
    console.log("Management of Content Types has been enabled for this list");
}

function OnFailure(sender, args) {
}

```

```

    console.log('Request failed. ' + args.get_message() + '\n' +
args.get_stackTrace());
}
</script>

```

## **Output:**

### Content Types

Specify whether to allow the management of content types on this list. Each content type will appear on the new button and can have a unique set of columns, workflows and other behaviors.

Allow management of content types?

Yes     No

## **D. Break Inheritance**

### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### **Script:**

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', breakInheritance);
});

function breakInheritance() {

    //Get the client context and list object
    var clientContext = new SP.ClientContext.get_current();
    var oList = clientContext.get_web().get_lists().getByTitle('DemoList');

    //Break inheritance
    oList.breakRoleInheritance(true, false);

    //Load the client context and execute the batch
    clientContext.load(oList);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    alert("Inheritance broken successfully!");
}

function QueryFailure(sender, args) {
    alert("There was an error breaking inheritance: " + args.get_message());
}

```

```

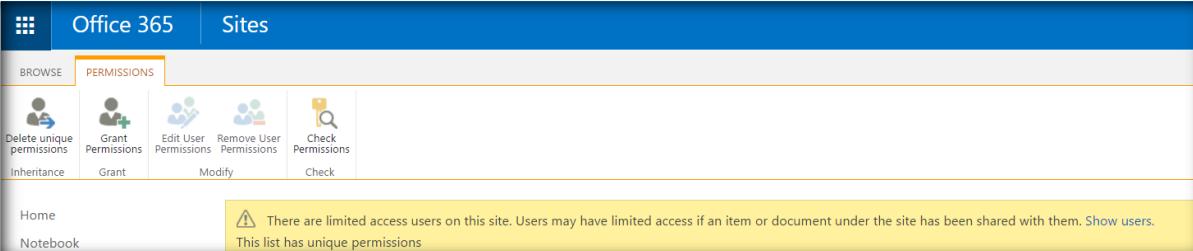
}

function QuerySuccess() {
    console.log("Inheritance has been broken.");
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}
</script>

```

## **Output:**



The screenshot shows the SharePoint Site Permissions page. At the top, there are tabs for 'BROWSE' and 'PERMISSIONS'. The 'PERMISSIONS' tab is selected. Below the tabs, there are five buttons: 'Delete unique permissions', 'Grant Permissions', 'Edit User Permissions', 'Remove User Permissions', and 'Check Permissions'. Underneath these buttons, there are two rows of links: 'Inheritance' (with 'Grant'), 'Grant' (with 'Modify'), and 'Check' (with 'Check'). In the bottom left corner, there are links for 'Home' and 'Notebook'. On the right side, there is a yellow warning message: '⚠ There are limited access users on this site. Users may have limited access if an item or document under the site has been shared with them. [Show users.](#)' and 'This list has unique permissions'.

## **E. Create List**

### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### **Script:**

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', createList);
});

function createList() {

    //Get client context and web object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Create list instance
    var listCreationInfo = new SP.ListCreationInformation();

```

```

listCreationInfo.set_title('Test List');
listCreationInfo.set_description('This is a Demo List');
listCreationInfo.set_templateType(SP.ListTemplateType.genericList);

//Add new list to list collection
var oListColl = oWeb.get_lists().add(listCreationInfo);

//Load the client context and execute the batch
clientContext.load(oListColl );
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

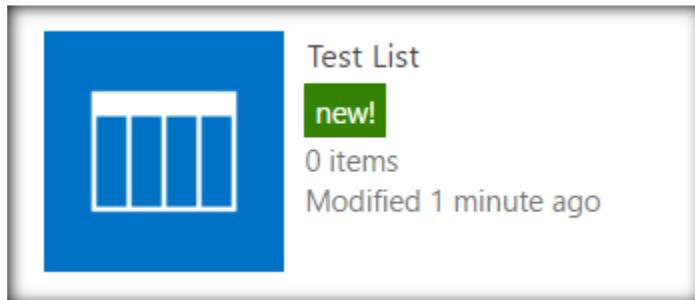
function QuerySuccess() {
    console.log("List has been created.");
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## Output:



## F. Delete List

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', deleteList);
});

function deleteList() {

    //Get the client context and web object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get the list object and delete it
    var oList = oWeb.get_lists().getByTitle('Test List');
    oList.deleteObject();

    //Execute the batch
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

    console.log("List has been deleted.");
}

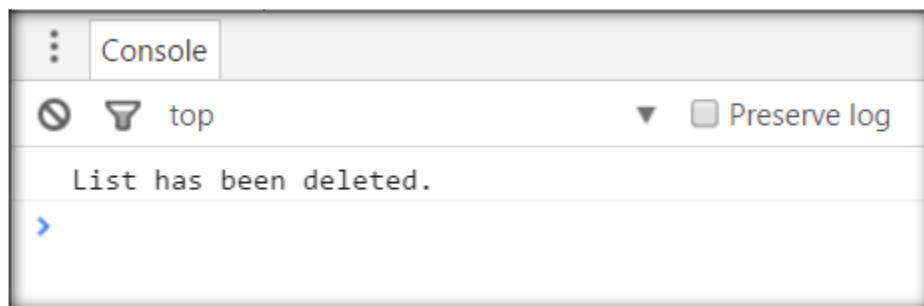
function QueryFailure(sender,args) {

    console.log('Request failed'+ args.get_message());
}

</script>

```

### Output:



## G. Set Draft Item Security

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', draftSecurity);
});

var oList, clientContext;

function draftSecurity() {

    //Get the client context , web and list object
    clientContext = new SP.ClientContext.get_current();
    var oWeb = clientContext.get_web();
    oList= oWeb.get_lists().getByTitle('DemoList');

    //Load the client context and execute the batch
    clientContext.load(oList);
    clientContext.executeQueryAsync(Success,Failure);
}

function Success(){

    //Set the draft security
    oList.set_draftVersionVisibility('2');
    oList.update();

    //Load the client context and execute the batch
    clientContext.load(oList);
    clientContext.executeQueryAsync(OnSuccess,OnFailure);
}

function Failure(sender, args) {
    console.log('Request failed. ' + args.get_message() + '\n' +
args.get_stackTrace());
}

function OnSuccess(){
    console.log("Draft item security have been enabled for this list");
}

```

```

}

function OnFailure(sender, args) {
    console.log('Request failed. ' + args.get_message() + '\n' +
args.get_stackTrace());
}
</script>

```

## **Output:**

### Draft Item Security

Drafts are minor versions or items which have not been approved. Specify which users should be able to view drafts in this list. [Learn about specifying who can view and edit drafts.](#)

Who should see draft items in this list?

- Any user who can read items
- Only users who can edit items
- Only users who can approve items (and the author of the item)

## **H. Get Base Template**

### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### **Script:**

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getBaseTemplate);
});

var oList;
function getBaseTemplate() {

    //Get client context , web and list object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();
    oList = oWeb.get_lists().getByTitle('DemoList');

    //Load the client context and execute the batch
    clientContext.load(oList);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

```

```

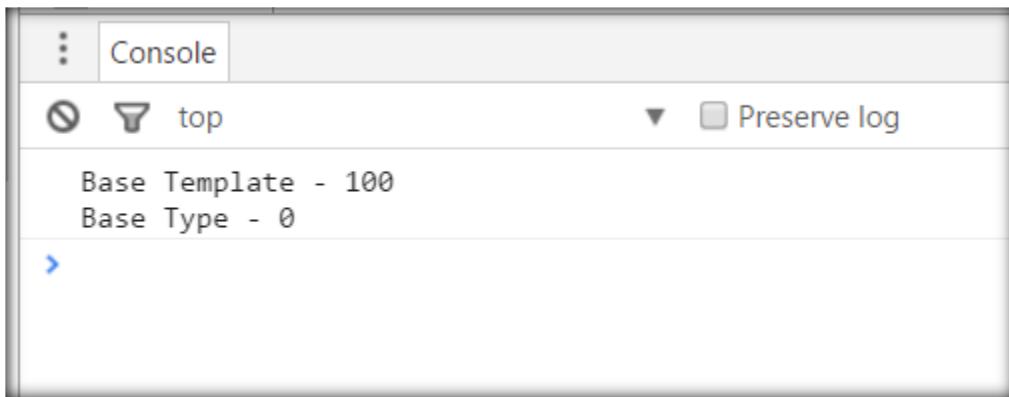
function QuerySuccess() {
    console.log("Base Template - " + oList.get_baseTemplate() + "\nBase Type - "
+oList.get_baseType());
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

### Output:



## I. Get List Properties

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getListProperties);
});

var oList;

```

```

function getListProperties() {
    //Get client context ,web and list object
    var clientContext = new SP.ClientContext.get_current();
    var oWeb = clientContext.get_web();
    oList= oWeb.get_lists().getByTitle('DemoList');

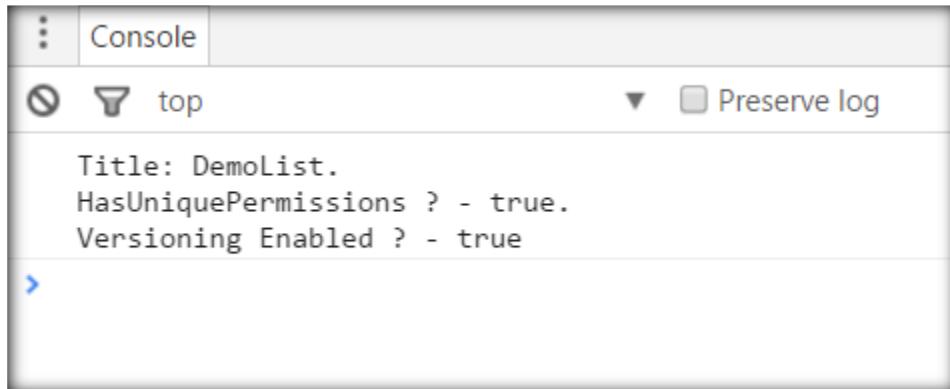
    //Load client context and execute the batch
    clientContext.load(oList,'Title','HasUniqueRoleAssignments','EnableVersioning');
    clientContext.executeQueryAsync(Success,Failure);
}

function Success() {
    console.log(' Title: ' + oList.get_title() + '\n HasUniquePermissions ? - ' +
oList.get_hasUniqueRoleAssignments() + '\n Versioning Enabled ? - ' +
'+oList.get_enableVersioning());
}

function Failure(sender, args) {
    console.log('Request failed. ' + args.get_message() + '\n' +
args.get_stackTrace());
}
</script>

```

## **Output:**



Console

Preserve log

```

Title: DemoList.
HasUniquePermissions ? - true.
Versioning Enabled ? - true

```

## **J. Get All Lists**

### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

## Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', retrieveAllList);
});

var oList;
function retrieveAllList() {

    //Get client context ,web and list collection object
    var clientContext = new SP.ClientContext.get_current();
    var oWeb = clientContext.get_web();
    oList= oWeb.get_lists();

    //Load the client context and execute the batch
    clientContext.load(oList);
    clientContext.executeQueryAsync(Success,Failure);
}

function Success() {

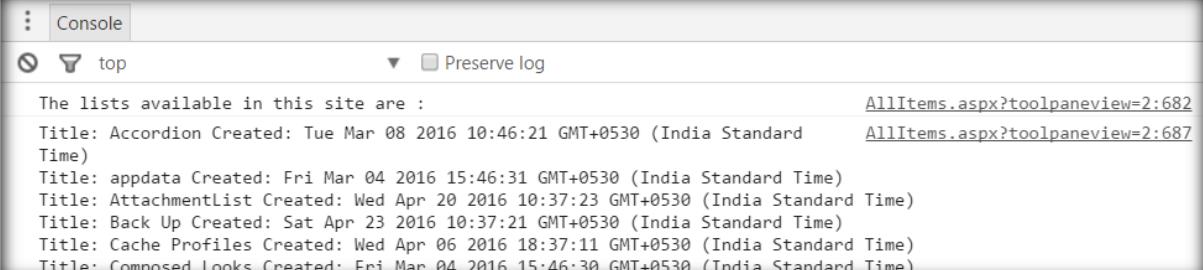
    //Get the list collection and loop through it
    var listInfo = '';
    var listEnumerator = oList.getEnumerator();
    console.log("The lists available in this site are :\n");
    while (listEnumerator.moveNext()) {
        var oCurrentList = listEnumerator.get_current();
        listInfo += 'Title: ' + oCurrentList.get_title() + ' Created: ' +
oCurrentList.get_created().toString() + '\n';
    }
    console.log(listInfo);
}

function Failure(sender, args) {
    console.log('Request failed. ' + args.get_message() + '\n' +
args.get_stackTrace());
}

</script>

```

## Output:



The lists available in this site are :

```

Title: Accordion Created: Tue Mar 08 2016 10:46:21 GMT+0530 (India Standard Time)
Title: appdata Created: Fri Mar 04 2016 15:46:31 GMT+0530 (India Standard Time)
Title: AttachmentList Created: Wed Apr 20 2016 10:37:23 GMT+0530 (India Standard Time)
Title: Back Up Created: Sat Apr 23 2016 10:37:21 GMT+0530 (India Standard Time)
Title: Cache Profiles Created: Wed Apr 06 2016 18:37:11 GMT+0530 (India Standard Time)
Title: Composed Looks Created: Fri Mar 04 2016 15:46:30 GMT+0530 (India Standard Time)

```

## K. Set List as Crawlable

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', setCrawlProperty);
});

var oList, clientContext;

function setCrawlProperty() {

    //Get client context ,web and list object
    clientContext = new SP.ClientContext.get_current();
    var oWeb = clientContext.get_web();
    oList= oWeb.get_lists().getByTitle('DemoList');

    //Load the client context and execute the batch
    clientContext.load(oList,'NoCrawl');
    clientContext.executeQueryAsync(Success,Failure);
}

function Success() {
}

```

```

//Set the no crawl property to true to prevent list from being crawlable
oList.set_noCrawl('true');
oList.update();

//Load the client context and execute the batch
clientContext.load(oList);
clientContext.executeQueryAsync(OnSuccess, onFailure);
}

function Failure(sender, args) {
    console.log('Request failed. ' + args.get_message() + '\n' +
args.get_stackTrace());
}

function OnSuccess() {
    console.log("List crawl setting updated");
}

function onFailure(sender, args) {
    console.log('Request failed. ' + args.get_message() + '\n' +
args.get_stackTrace());
}
</script>

```

## Output:

<p><b>Search</b></p> <p>Specify whether this list should be visible in search results. Users who do not have permission to see these items will not see them in search results, no matter what this setting is.</p>	<p>Allow items from this list to appear in search results?</p> <p><input type="radio"/> Yes    <input checked="" type="radio"/> No</p>
---	--

## L. Get Entity Type Full Name

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getEntityTypeName);
});

var oList;

function getEntityTypeName() {

    //Get client context ,web and list object
    clientContext = new SP.ClientContext.get_current();
    var oWeb = clientContext.get_web();
    oList= oWeb.get_lists().getByTitle('DemoList');

    //Load the client context and execute the batch
    clientContext.load(oList,'ListItemEntityTypeFullName');
    clientContext.executeQueryAsync(Success,Failure);
}

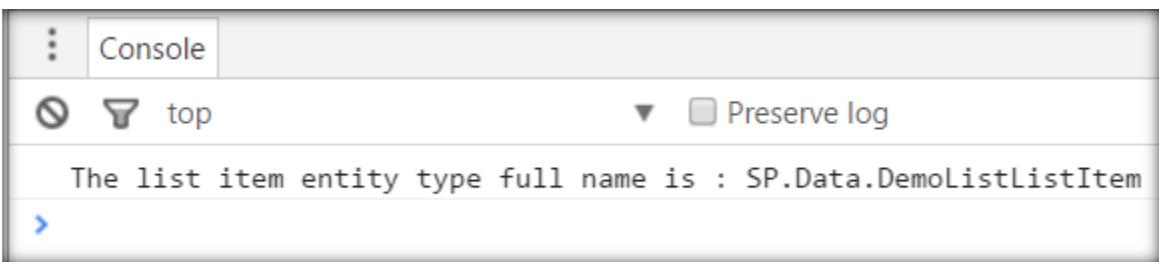
function Success() {

    //Get the entity type full name
    var listItemEntityType = oList.get_listItemEntityTypeFullName();
    console.log("The list item entity type full name is : " + listItemEntityType)
}

function Failure(sender, args) {
    console.log('Request failed. ' + args.get_message() + '\n' +
args.get_stackTrace());
}
</script>

```

### Output:



The screenshot shows a browser's developer tools console window. The title bar says 'Console'. Below it are buttons for 'Clear' (trash can icon), 'Top' (up arrow icon), and 'Preserve log' (checkbox). The main area of the console displays the following text:

```
The list item entity type full name is : SP.Data.DemoListListItem
```

## M. Set Validation Formula

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', SetValidation);
});

var oList, clientContext;

function SetValidation() {

    //Get client context ,web and list object
    clientContext = new SP.ClientContext.get_current();
    var oWeb = clientContext.get_web();
    oList= oWeb.get_lists().getByTitle('DemoList');

    //Load the client context and execute the batch
    clientContext.load(oList);
    clientContext.executeQueryAsync(Success,Failure);
}

function Success() {

    //Set validation formula
    oList.set_validationFormula("[StartDate] < TODAY()");
    oList.set_validationMessage('Validation Failed ! Start Date should be less
than Today.');
    oList.update();

    //Load the client context and execute the batch
    clientContext.load(oList);
    clientContext.executeQueryAsync(OnSuccess,OnFailure);

}

function Failure(sender, args) {
    console.log('Request failed. ' + args.get_message() + '\n' +
args.get_stackTrace());
}

```

```

}

function OnSuccess() {
    console.log("List validation formula has been set for this list");
}

function onFailure(sender, args) {
    console.log('Request failed. ' + args.get_message() + '\n' +
args.get_stackTrace());
}
</script>

```

### **Output:**

Settings ▶ Validation Settings

**Formula**

Specify the formula you want to use to validate data when new items are saved to this list. To pass validation, the formula must evaluate to TRUE. For more information, see Formulas in Help.

Example: =[Discount]<[Cost] will only pass validation if column Discount is less than column Cost.

[Learn more about proper syntax for formulas.](#)

Formula:

=StartDate < TODAY()

Insert Column:

- Age
- Approval Status
- Column1
- Column2
- Column3
- Created
- DOB
- Employee Name
- Expertise
- February

[Add to formula](#)

## **N. Enable List Versioning**

### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

## Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', listVersioning);
});

var oList, clientContext;

function listVersioning() {

    //Get client context ,web and list object
    clientContext = new SP.ClientContext.get_current();
    var oWeb = clientContext.get_web();
    oList= oWeb.get_lists().getByTitle('DemoList');

    //Load the client context and execute the batch
    clientContext.load(oList);
    clientContext.executeQueryAsync(Success,Failure);
}

function Success(){

    //Enable Versioning
    oList.set_enableVersioning('true');
    oList.update();

    //Load the client context and execute the batch
    clientContext.load(oList);
    clientContext.executeQueryAsync(OnSuccess,OnFailure);

}

function Failure(sender, args) {
    console.log('Request failed. ' + args.get_message() + '\n' +
args.get_stackTrace());
}

function OnSuccess() {
    console.log("Versioning have been enabled for this list");
}

function OnFailure(sender, args) {
    console.log('Request failed. ' + args.get_message() + '\n' +
args.get_stackTrace());
}
</script>

```

## **Output:**

Item Version History

Specify whether a version is created each time you edit an item in this list.  
[Learn about versions.](#)

Create a version each time you edit an item in this list?

Yes  No

Optionally limit the number of versions to retain:

Keep the following number of versions:

Keep drafts for the following number of approved versions:

## **O. Reset Inheritance**

### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### **Script:**

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', resetInheritance);
});

var collList, oList, clientContext ;
function resetInheritance() {

    //Get client context ,web and list collection object
    clientContext = new SP.ClientContext();
    var web = clientContext.get_web();
    collList = web.get_lists();

    //Load the client context and execute the batch
    clientContext.load(collList,'Include(HasUniqueRoleAssignments)');
    clientContext.executeQueryAsync(QuerySuccess,QueryFailure);

}

function QuerySuccess() {
```

```

//get list collection object and loop through it
var listEnumerator = collList.getEnumerator();

while (listEnumerator.moveNext()) {
    var oList = listEnumerator.get_current();
    if(oList.get_hasUniqueRoleAssignments()){
        //If list has unique permission start inheriting permission from
parent
        oList.resetRoleInheritance();

        //Load the client context and execute the batch
        clientContext.load(oList);
        clientContext.executeQueryAsync(OnQuerySuccess, OnQueryFailure);
    }
}

function QueryFailure(sender,args) {
    console.log('Request failed - '+args.get_message());
}
function OnQuerySuccess(sender,args) {
    console.log('Request Succeeded');
}
function OnQueryFailure(sender,args) {
    console.log('Request failed - '+args.get_message());
}
</script>

```



## P. Update List

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', updateList);
});

function updateList() {

    //Get client context ,web and list object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();
    var oList = oWeb.get_lists().getByTitle('TestList');

    //Change list title
    oList.set_title('List Name Changed');
    oList.update();

    //Load client context and execute the batch
    clientContext.load(oList);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

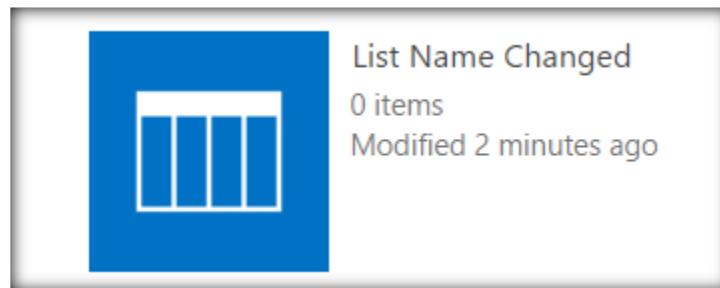
function QuerySuccess() {
    console.log("List title updated.");
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## Output:



## IX. Working with List Items

### A. Create List Item

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', createListItem);
});

var oListItem;
function createListItem() {

//Get client context and web object
var clientContext = new SP.ClientContext();
var oWeb = clientContext.get_web();

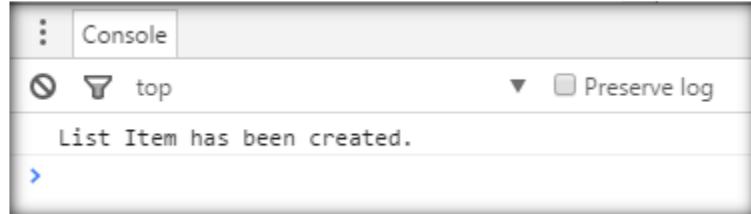
//Get List and create list item using ListItemCreationInformation object
var oList = oWeb.get_lists().getByTitle('Promise');
var oListItemCretionInformation= new SP.ListItemCreationInformation();
oListItem= oList.addItem(oListItemCretionInformation);
oListItem.set_item('Title','List Item Created using JSOM in Office 365');
oListItem.update();

//Load the client context and execute the batch
clientContext.load(oListItem);
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);}

function QuerySuccess() {
    console.log("List Item has been created.");
}

function QueryFailure() {
    console.log('Request failed'+ args.get_message());
}
</script>
```

## Output:



A screenshot of a SharePoint list view. At the top, there's a header with a plus sign icon for 'new item or edit this list', 'All Items', a search bar, and a 'SAVE THIS VIEW' button. Below the header, there are two columns: 'Title' and 'ID'. The first item in the list has the title 'List Item Created using JSOM in Office 365' and the ID '6'.

## B. Batch Create List Item

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext',
batchCreateListItems);
});
```

```

function batchCreateListItems() {
    //Get client context and list object
    var clientContext = SP.ClientContext.get_current();
    var oList = clientContext.get_web().get_lists().getByTitle('Promise');

    //Within the loop create ListItemCreationInformation object and add it to
    client context.
    for(var i = 0; i< 5; i++){
        var itemCreateInfo = new SP.ListItemCreationInformation();
        var oListItem = oList.addItem(itemCreateInfo);
        oListItem.set_item('Title', 'Item Order - ' + i);
        oListItem.update();
        clientContext.load(oListItem);
    }

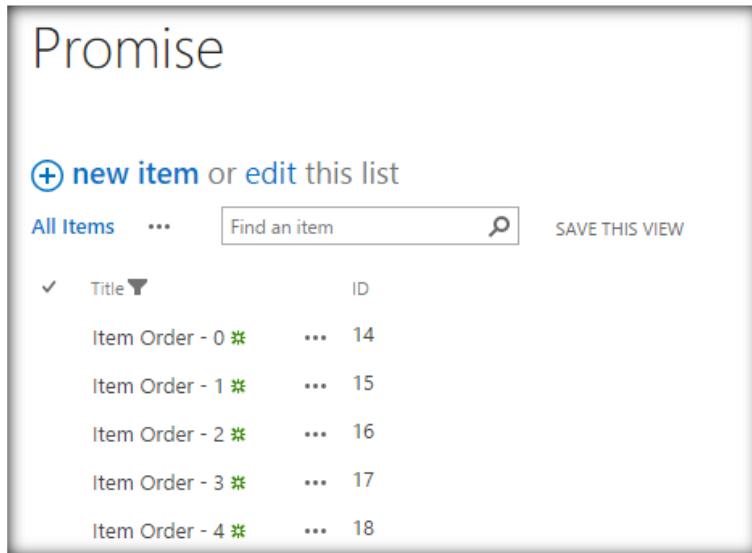
    //Execute the batch
    clientContext.executeQueryAsync(onQuerySucceeded, onQueryFailed);
}

function onQuerySucceeded() {
    console.log("List items have been created.");
}

function onQueryFailed(sender, args) {
    console.log('Request failed'+ args.get_message());
}
</script>

```

### Output:



The screenshot shows a SharePoint list titled "Promise". The list has two columns: "Title" and "ID". There are five items in the list:

Title	ID
Item Order - 0 *	14
Item Order - 1 *	15
Item Order - 2 *	16
Item Order - 3 *	17
Item Order - 4 *	18

## C. Retrieve List Item Conditionally

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', retrieveListItem);
});

var collListItem;
function retrieveListItem() {

    //Get client context and web object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //get the list object and retrieve the list items using caml query
    var oList = oWeb.get_lists().getByTitle('Promise');
    var camlQuery = new SP.CamlQuery();

    //Specify the condition - We are getting items based on a specific title
    value
    camlQuery.set_viewXml("<View><Query><Where><Eq><FieldRef
Name=Title/><Value Type=Text>List Item Created Via
JSOM</Value></Eq></Where></Query></View>");
    collListItem = oList.getItems(camlQuery);

    //Load the client context and execute the batch
    clientContext.load(collListItem);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);

}

function QuerySuccess() {

    //Create enumerator object and loop through the items
    var listItemEnumerator = collListItem.getEnumerator();
    while (listItemEnumerator.moveNext()) {

```

```

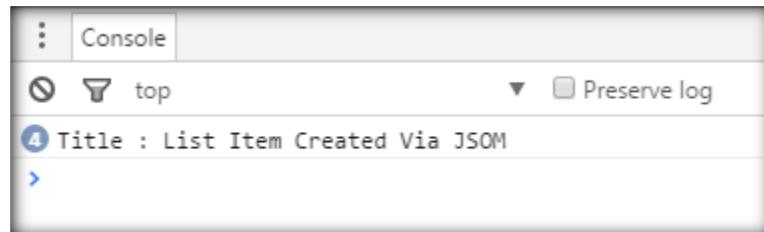
        var oListItem = listItemEnumerator.get_current().get_item('Title');
        console.log("Title : " + oListItem );
    }

function QueryFailure() {
    console.log('Request failed'+ args.get_message());
}

</script>

```

### **Output:**



### **D. Retrieve All List Items**

#### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### **Script:**

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', retrieveListItems);
});

var collListItem ;
function retrieveListItems() {

    //Get client context,web and List object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

```

```

var oList = oWeb.get_lists().getByTitle('Employees');

//Get list item collection
collListItem = oList.getItems(SP.CamlQuery.createAllItemsQuery());

//Load the client context and execute the batch
clientContext.load(collListItem);
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);

}

function QuerySuccess() {

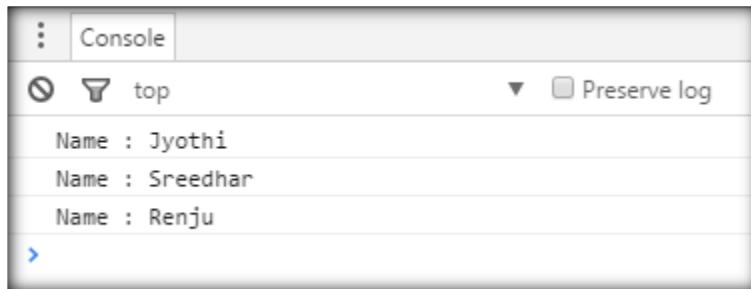
    //Create enumerator object and loop through the collection
    var listItemEnumerator = collListItem.getEnumerator();
    while (listItemEnumerator.moveNext()) {
        var oListItem = listItemEnumerator.get_current().get_item('Name');
        console.log("Name : " + oListItem );
    }
}

function QueryFailure() {
    console.log('Request failed'+ args.get_message());
}
</script>

```

### Output:

Employees				
<a href="#">+ new item or edit this list</a>				
<a href="#">All Items</a>		<a href="#">Find an item</a> <span style="float: right;">🔍</span>		
✓	Title	Name	Address	Employee Number
1	...	Jyothi	Palakkad,Kerala	354,897
2	...	Sreedhar	Ernakulam,Kerala	257,654
3	...	Renju	Kollam,Kerala	354,456



## E. Delete List Items

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', deleteListItem);
});

var collListItem,clientContext ;
function deleteListItem() {

    //Get client context and web object
    clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get the list object and get the list item to delete using caml query
    var oList = oWeb.get_lists().getByTitle('Promise');
    var camlQuery = new SP.CamlQuery();
    camlQuery.set_viewXml('<View><Query><Where><Eq><FieldRef
Name=\'Title\'/>' + '<Value
Type=\'Text\'>Test</Value></Eq></Where></Query></View>');
    collListItem = oList.getItems(camlQuery);

    clientContext.load(collListItem, "Include(Title)");
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}
```

```

}

function QuerySuccess() {
    //Get list item count of the items to be deleted and within the loop, set
    //the object to be deleted
    var itemCount = collListItem.get_count();
    for (var i = itemCount - 1; i >= 0; i--) {
        var oListItem = collListItem.itemAt(i);
        oListItem.deleteObject();
    }

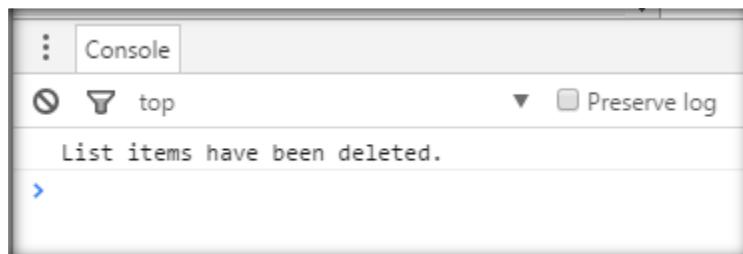
    //Execute the batch which will delete the items
    clientContext.executeQueryAsync(deleteSucceeded, deleteFailed);
}

function QueryFailure() {
    console.log('Request failed'+ args.get_message());
}

function deleteSucceeded()
{
    console.log("List items have been deleted.");
}
function deleteFailed()
{
    console.log('Request failed'+ args.get_message());
}
</script>

```

### Output:



## X. Working with Navigation

### A. Add Top Navigation Nodes

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', addTopNavigation);
});

var oQuickLaunchColl;
function addTopNavigation() {

    //Get the client context,web and top navigation object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();
    oTopNavigationColl = oWeb.get_navigation().get_topNavigationBar();

    //Add a new top navigation object using NavigationNodeCreationInformation
    //object
    var oNavigation = new SP.NavigationNodeCreationInformation();

    oNavigation.set_title("Employee Master List");
    oNavigation.set_url("/sites/playground/Lists/Employees/AllItems.aspx");
    oTopNavigationColl.add(oNavigation);

    //Load the client context and execute the batch
    clientContext.load(oTopNavigationColl);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

    //Get the node count and loop through the collection
}

```

```

var itemCount = oTopNavigationColl.get_count();

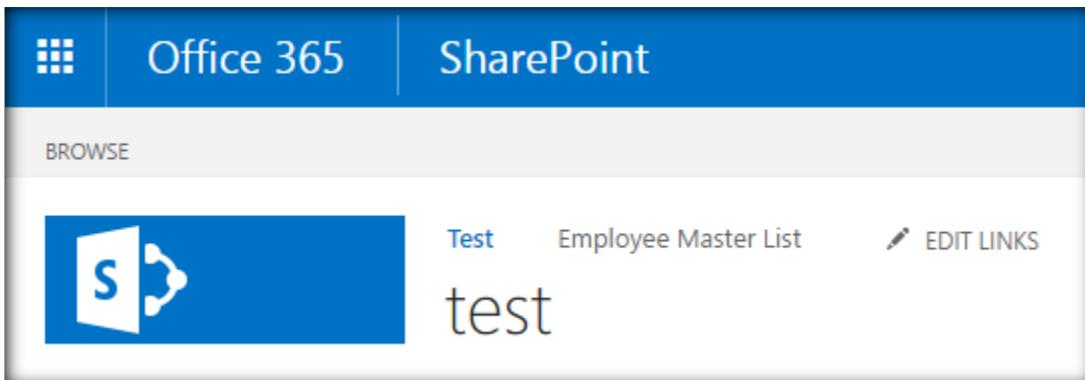
console.log("Top Navigation Details are:");
for (var i = 0; i < itemCount; i++) {
    var title = oTopNavigationColl.get_item(i).get_title();
    console.log(title);
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

### **Output:**



### **B. Add Quick Launch Nodes**

#### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### **Script:**

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

```

```

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', addNavigation);
});

var oQuickLaunchColl;
function addNavigation() {

    //Get client context and web object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get quick launch object and add the navigation
    oQuickLaunchColl =oWeb.get_navigation().get_quickLaunch();

    //Create a navigation node using NavigationCreationInformation
    var oNavigation = new SP.NavigationNodeCreationInformation();
    oNavigation.set_title("Employee Master List");
    oNavigation.set_url("/sites/playground/Lists/Employees/AllItems.aspx");
    oQuickLaunchColl.add(oNavigation);

    //Load the client context and execute the batch
    clientContext.load(oQuickLaunchColl);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

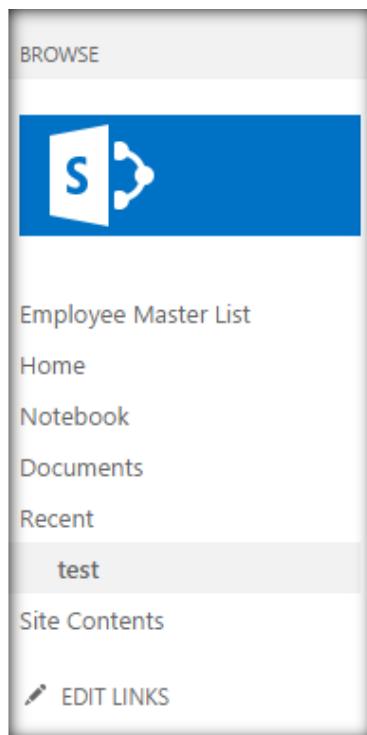
    //Get the item count of navigation nodes and loop through it
    var itemCount = oQuickLaunchColl.get_count();
    console.log("The navigation node details are:");
    for (var i = 0; i < itemCount; i++) {
        var title = oQuickLaunchColl.get_item(i).get_title();
        console.log(title);
    }
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

### Output:



## C. Delete Quick Launch Navigation Nodes

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext',
deleteNavigationQuickLaunch);
});

var oListItem,oQuickLaunchColl,clientContext;
```

```

function deleteNavigationQuickLaunch() {
    //Get the client context,web and quick launch collection object
    clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();
    oQuickLaunchColl = oWeb.get_navigation().get_quickLaunch();

    //load the client context and execute the batch
    clientContext.load(oQuickLaunchColl);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    //Get the item count and mark the node to be deleted
    var itemCount = oQuickLaunchColl.get_count();
    var deletearray= new Array();

    for (var i = 0; i < itemCount; i++) {
        var title = oQuickLaunchColl.get_item(i).get_title();
        if(title=="Employee Master List")
        {
            var oQuickLaunchItem = oQuickLaunchColl.itemAt(i);
            deletearray.push(oQuickLaunchItem);

        }
    }
    //Delete the marked nodes
    for (var i = 0; i < deletearray.length; i++) {
        deletearray[i].deleteObject();
    }

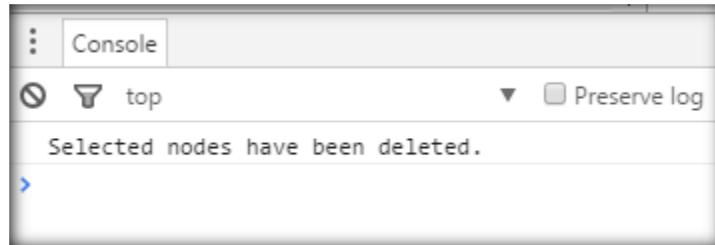
    //Execute the batch
    clientContext.executeQueryAsync(SecondQuerySuccess, SecondQueryFailure);

}
function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}
function SecondQuerySuccess() {
    console.log("Selected nodes have been deleted.");
}
function SecondQueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## Output:



## D. Delete Top Navigation Nodes

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', deleteTopNavigation);
});

var oTopNavigationColl,clientContext;
function deleteTopNavigation() {

    //Get the client context,web and top navigation collection
    clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();
    oTopNavigationColl =oWeb.get_navigation().get_topNavigationBar();

    //Get the client context and execute the batch
    clientContext.load(oTopNavigationColl);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
```

```

//Get the item count and mark the navigation nodes to be deleted
var itemCount = oTopNavigationColl.get_count();
var deletearray= new Array();
for (var i = 0; i < itemCount; i++) {
    var title = oTopNavigationColl.get_item(i).get_title();
    if(title=="Employee Master List")
    {
        var oTopNavigationItem = oTopNavigationColl.itemAt(i);
        deletearray.push(oTopNavigationItem);
    }
}

//Delete the top navigation items
for (var i = 0; i < deletearray.length; i++) {
    deletearray[i].deleteObject();
}
//Execute the batch
clientContext.executeQueryAsync(SecondQuerySuccess, SecondQueryFailure);

}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

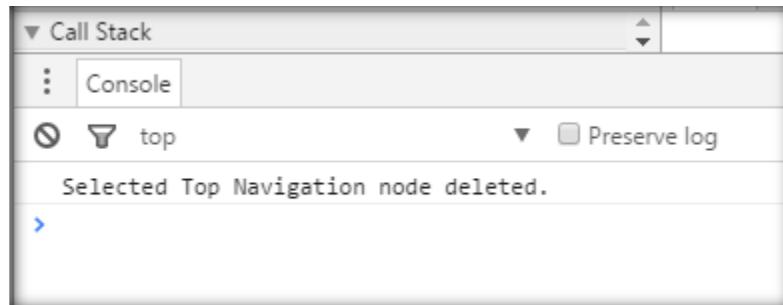
function SecondQuerySuccess() {
    console.log("Selected Top Navigation node deleted.");
}

function SecondQueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## Output:



## E. Get Child Nodes

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getChildNodes);
});

var oListItem,oQuickLaunchColl,clientContext ;
function getChildNodes() {

//Get client context,web and quick launch collection
clientContext = new SP.ClientContext();
var oWeb = clientContext.get_web();
oQuickLaunchColl =oWeb.get_navigation().get_quickLaunch();

//Load the client context and execute the batch
clientContext.load(oQuickLaunchColl);
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

var children;
function QuerySuccess() {

//Get the quick launch node count and loop through it
var itemCount = oQuickLaunchColl.get_count();
for (var i = 0; i < itemCount; i++) {
    var title = oQuickLaunchColl.get_item(i).get_title();
    if(title=="Recent")
    {

        //Get the children nodes and load it to the client context
        children = oQuickLaunchColl.get_item(i).get_children();
        clientContext.load(children);
    }
}
}
```

```

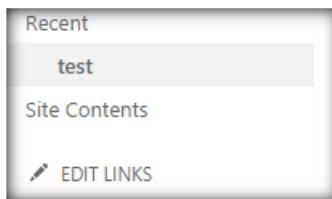
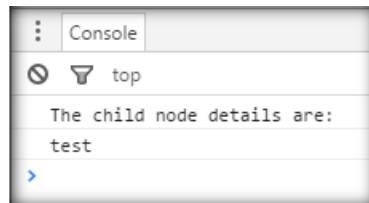
//Execute the batch
clientContext.executeQueryAsync(SecondQuerySuccess, SecondQueryFailure);
}

function QueryFailure() {
    console.log('Request failed'+ args.get_message());
}
function SecondQuerySuccess() {
    var itemCount = children.get_count();

    //Get the child node details
    console.log("The child node details are:");
    for (var i = 0; i < itemCount; i++) {
        var title = children.get_item(i).get_title();
        console.log(title);
    }
}
function SecondQueryFailure() {
    console.log('Request failed'+ args.get_message());
}
</script>

```

### Output:



## F. Get Quick Launch Nodes

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

## Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', quickLaunch);
});


var oQuickLaunchColl;
function quickLaunch() {

    //Get the client context,web and quick launch items.
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();
    oQuickLaunchColl =oWeb.get_navigation().get_quickLaunch();

    //Load the client context and execute the batch
    clientContext.load(oQuickLaunchColl);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

    //Get the quick launch navigation node item count and loop through it to
    get the node details
    var itemCount = oQuickLaunchColl.get_count();
    console.log("The node details are:");
    for (var i = 0; i < itemCount; i++) {
        var title = oQuickLaunchColl.get_item(i).get_title();
        var url = oQuickLaunchColl.get_item(i).get_url();
        console.log(title+ "--- "+url );
    }
}

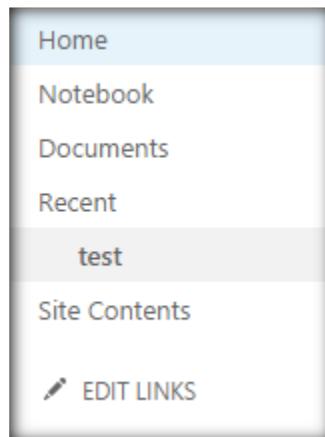
function QueryFailure() {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## Output:

```
Console
top ▼ Preserve log
The node details are:
Home--/sites/Playground/test
Notebook--/sites/Playground/test/_layouts/15/WopiFrame.aspx?sourcedoc={83a4cf41-e607-40e0-b6b6-cbb6a8b65a58}&action=editnew
Documents--/sites/Playground/test/Shared Documents/Forms/AllItems.aspx
Recent--/sites/Playground/test
Site Contents--/sites/Playground/test/_layouts/15/viewlsts.aspx
>
```



## G. Get Top Navigation Nodes

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">
$(document).ready(function() {
```

```

SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getTopNavigation);
});

var oTopNavigationColl;
function getTopNavigation() {

    //Get client context,web and top navigation object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();
    oTopNavigationColl =oWeb.get_navigation().get_topNavigationBar();

    //Load client context and execute the batch
    clientContext.load(oTopNavigationColl);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

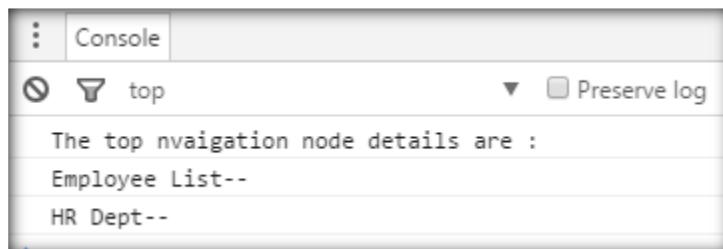
    //Get top navigation node count, loop through it and get the node
    properties.
    var itemCount = oTopNavigationColl.get_count();
    console.log("The top nvaigation node details are :");
    for (var i = 0; i < itemCount; i++) {
        var title = oTopNavigationColl.get_item(i).get_title();
        var url = oTopNavigationColl.get_item(i).get_url();
        console.log(title+"--"+url );
    }
}

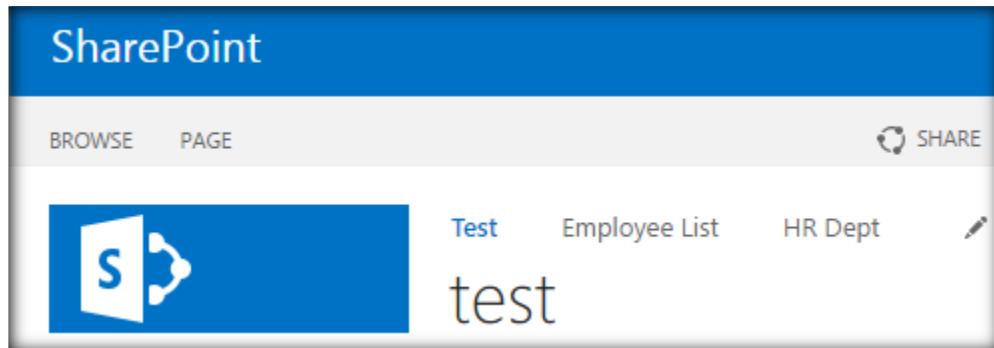
function QueryFailure() {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## Output:





## H. Update Quick Launch Nodes

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', addNavigation);
});

var oListItem,oNavigationColl,clientContext ;
function addNavigation() {

//Get client context,web and quick launch object
clientContext = new SP.ClientContext();
var oWeb = clientContext.get_web();
oNavigationColl =oWeb.get_navigation().get_quickLaunch();

//Launch the client context and execute the batch
clientContext.load(oNavigationColl);
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

var children;
function QuerySuccess() {
```

```

//Get the navigation node item count and loop through it
var itemCount = oNavigationColl.get_count();
for (var i = 0; i < itemCount; i++) {

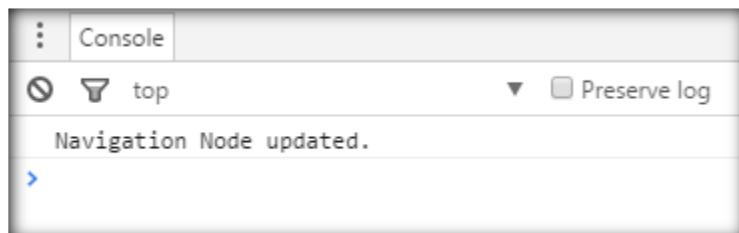
//Check the node title and if it matches update the url property
    var title = oNavigationColl.get_item(i).get_title();
    if(title=="Test List")
    {
        oNode = oNavigationColl.get_item(i);
        oNode.set_url("/sites/playground/Lists/Employees/AllItems.aspx");
        oNode.update();
    }
}

//Execute the batch
clientContext.executeQueryAsync(SecondQuerySuccess, SecondQueryFailure);
}

function QueryFailure() {
    console.log('Request failed'+ args.get_message());
}
function SecondQuerySuccess() {
    console.log("Navigation Node updated.");
}
function SecondQueryFailure() {
    console.log('Request failed'+ args.get_message());
}
</script>

```

### Output:



## XI. Working with Document Set

### A. Create a Document Set

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
        $.getScript(scriptbase + "SP.js", function() {
            $.getScript(scriptbase + "SP.DocumentManagement.js",createDocumentSet);
        });
    });
});

var oLibraryFolder,clientContext,docSetContentType;
function createDocumentSet() {
    //Get the client context,web and library object.
    clientContext = new SP.ClientContext.get_current();
    oWeb = clientContext.get_web();
    var oList = oWeb.get_lists().getByTitle("Demo Library");

    //Load the document library object
    clientContext.load(oList);

    //Get the root folder of the library and load it
    oLibraryFolder = oList.get_rootFolder();
    clientContext.load(oLibraryFolder);

    //Set the content type for the document set and load it
    var documentSetContentTypeID = "0x0120D520";
}

```

```

documentSetContentType =
clientContext.get_site().get_rootWeb().get_contentTypes().getById(documentSetContentTypeID);
clientContext.load(documentSetContentType);

//Execute the batch
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);

}

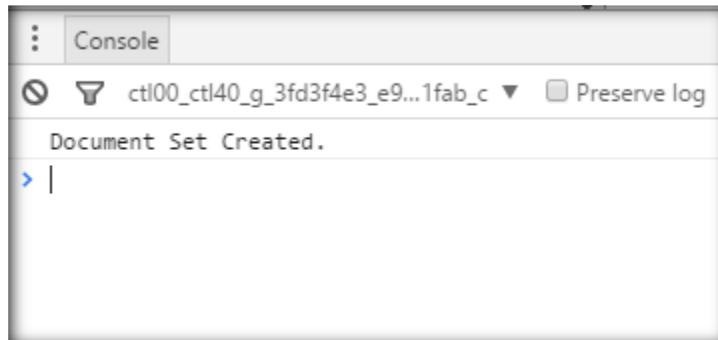
function QuerySuccess() {
    //Set the document set name
    var documentSetName = "Long Term Execution Planning";
    SP.DocumentSet.DocumentSet.create(clientContext, oLibraryFolder, documentSetName,
documentSetContentType.get_id());
    clientContext.executeQueryAsync(SecondQuerySuccess, SecondQueryFailure);
}

function QueryFailure() {
    console.log('Request failed - ' + args.get_message());
}
function SecondQuerySuccess() {
    console.log('Document Set Created.');
}
function SecondQueryFailure(sender,args) {
    console.log('Request failed - ' + args.get_message());
}

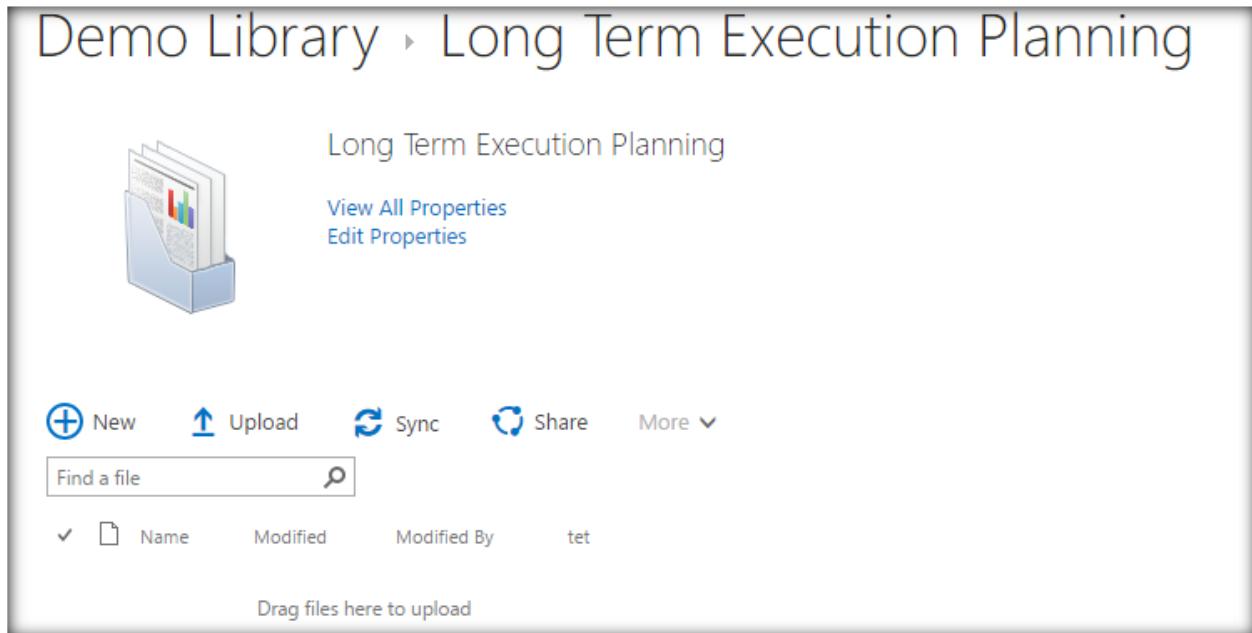
</script>

```

## Output:



Demo Library › Long Term Execution Planning



## B. Get All Files from Document Set

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
        $.getScript(scriptbase + "SP.js", function () {
            $.getScript(scriptbase +
"SP.DocumentManagement.js",createDocumentSet);
    });
});
```

```

        });
    });

});

var docSetFiles;
function createDocumentSet() {

    //Get the client context,web and library object.
    clientContext = new SP.ClientContext.get_current();
    oWeb = clientContext.get_web();
    var oList = oWeb.get_lists().getByTitle("Demo Library");
    clientContext.load(oList);

    //Get the root folder of the library
    oLibraryFolder = oList.get_rootFolder();
    var documentSetFolder ="/sites/Playground/Demo%20Library/Long Term Execution
Planning";

    //Get the document set files using CAML query
    var camlQuery = SP.CamlQuery.createAllItemsQuery();
    camlQuery.set_folderServerRelativeUrl(documentSetFolder);
    docSetFiles = oList.getItems(camlQuery);

    //Load the client context and execute the batch
    clientContext.load(docSetFiles,'Include(File)');
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);

}

function QuerySuccess() {

    //Loop through the document set files and get the display name
    var docSetFilesEnumerator = docSetFiles.getEnumerator();

    while (docSetFilesEnumerator .moveNext()) {
        var oDoc = docSetFilesEnumerator.get_current().get_file();
        console.log("Document Name : " + oDoc.get_name());
    }

}

function QueryFailure() {
    console.log('Request failed - ' + args.get_message());
}

</script>

```

## **Output:**

## Demo Library › Long Term Execution Planning



### Long Term Execution Planning

[View All Properties](#)  
[Edit Properties](#)

New     Upload     Sync     Share    More [▼](#)

Find a file

✓	Name	Modified	Modified By	tet
	Budget Estimate *	... A few seconds ago	<input type="checkbox"/>	Priyaranjan KS
	Execution Test Plan *	... A few seconds ago	<input type="checkbox"/>	Priyaranjan KS
	Paymentreceipt *	... 2 minutes ago	<input type="checkbox"/>	Priyaranjan KS

```
⋮ : Console
    ⚡ top ▼ Preserve log
Document Name : Paymentreceipt.pdf
Document Name : Budget Estimate.docx
Document Name : Execution Test Plan.docx
>
```

## XII. Working with Publishing Page

### A. Create Publishing Page

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl +
"/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
        $.getScript(scriptbase + "SP.js", function () {
            $.getScript(scriptbase + "SP.Publishing.js",
createPublishingPage);

        });
    });
});

var oWeb,clientContext,pageLayoutItem;
function createPublishingPage() {
    //Get the client context,web and list object(Master Page Gallery)
    clientContext = new SP.ClientContext.get_current();
    oWeb = clientContext.get_web();
    var oList = oWeb.get_lists().getByTitle('Master Page Gallery');

    //Get the page layout by ID using which we will create a publishing page
    pageLayoutItem = oList.getItemById(2268);

    //Load the client context and execute the batch
    clientContext.load(oWeb);
}
```

```

clientContext.load(pageLayoutItem);
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    //Create Publishing Page using PublishingPageInformation object
    var newPublishingPage =
SP.Publishing.PublishingWeb.get PublishingWeb(clientContext,oWeb);
    var pageInfo = new SP.Publishing.PublishingPageInformation();
    pageInfo.set_name("New Publishing Page.aspx");
    pageInfo.set_pageLayoutListItemId(pageLayoutItem);
    newPage = newPublishingPage.addPublishingPage(pageInfo);

    //Load the new page object to the client context
    clientContext.load(newPage);
    clientContext.executeQueryAsync(SecondQuerySuccess, SecondQueryFailure);
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

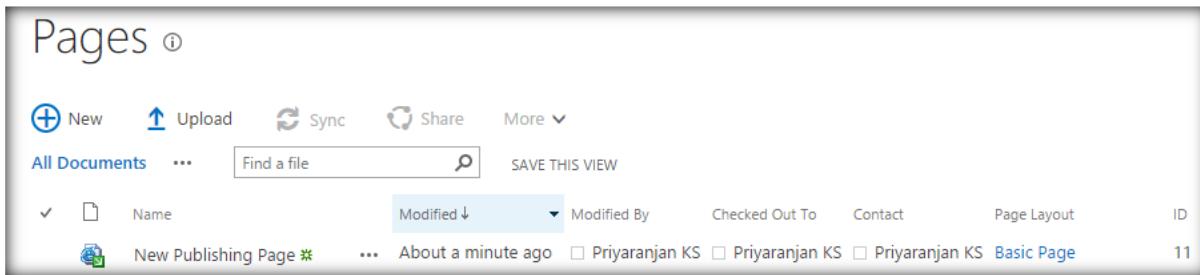
function SecondQuerySuccess(sender,args) {
    console.log("Publishing page created successfully.");
}

function SecondQueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## Output:



The screenshot shows a SharePoint 'Pages' library interface. At the top, there are buttons for 'New', 'Upload', 'Sync', 'Share', and 'More'. Below the header, there's a search bar labeled 'Find a file' and a 'SAVE THIS VIEW' button. The main area displays a table with columns: 'Name', 'Modified', 'Modified By', 'Checked Out To', 'Contact', 'Page Layout', and 'ID'. A single item is listed: 'New Publishing Page' (with a green asterisk icon), modified 'About a minute ago', by 'Priyanjan KS', checked out to 'Priyanjan KS', and has a 'Basic Page' layout. The ID is 11.

## B. Set Publishing Page Content

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {

    var scriptbase = _spPageContextInfo.webServerRelativeUrl +
"/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {

        $.getScript(scriptbase + "SP.js", function(){

            $.getScript(scriptbase + "SP.Publishing.js",
createPublishingPage);

        });
    });
});

var oSite,oWeb,clientContext,pageLayoutItem,oFileItem;
function createPublishingPage() {

    //Get Client Context and Web object
    clientContext = new SP.ClientContext.get_current();
    oWeb = clientContext.get_web();

    //Get the list object for Pages Library and get the page that has to be
updated
    var oListPages = oWeb.get_lists().getByTitle('Pages');
    oFileItem = oListPages.getItemById(11);

    //Load the client context and execute the batch
    clientContext.load(oWeb);
    clientContext.load(oFileItem);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

```

```

function QuerySuccess() {
    //Set the publishing page content and update the page
    oFileItem.set_item("PublishingPageContent", '<span
style="margin:auto;background-color:Yellow; font-size:18px">Office 365
Publishing Page content ...</span>');
    oFileItem.set_item("PublishingPageImage", "<img alt='image'
src='/sites/Playground/SiteAssets/SP.png'>");
    oFileItem.update();

    //Load the client context and execute the batch
    clientContext.load(oFileItem);
    clientContext.executeQueryAsync(SecondQuerySuccess, SecondQueryFailure);
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

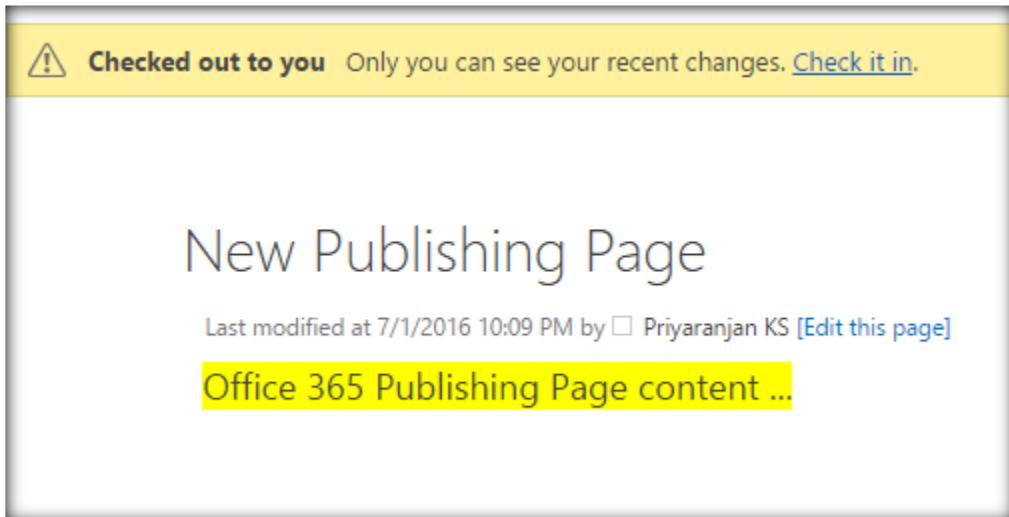
function SecondQuerySuccess() {
    console.log('Publishing Page content has been updated.');
}

function SecondQueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

### Output:



⚠️ Checked out to you Only you can see your recent changes. [Check it in.](#)

## New Publishing Page

Last modified at 7/1/2016 10:09 PM by  Priyaranjan KS [Edit this page]

Office 365 Publishing Page content ...

## C. Get Publishing Page Content

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl +
"/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
        $.getScript(scriptbase + "SP.js", function () {
            $.getScript(scriptbase + "SP.Publishing.js",
createPublishingPage);

        });
    });
});

var oSite,oWeb,clientContext,pageLayoutItem,oFileItem;
function createPublishingPage() {
//Get the client context and web object
clientContext = new SP.ClientContext.get_current();
oWeb = clientContext.get_web();

//Get the pages library and the required page by id
var oListPages = oWeb.get_lists().getByTitle('Pages');
oFileItem = oListPages.getItemById(11);

//Load the client context and execute the batch
clientContext.load(oWeb);
clientContext.load(oFileItem);
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

var content,image ;
function QuerySuccess() {

```

```

//Get the content and image associated with the page
content = oFileItem.get_item("PublishingPageContent");
image = oFileItem.get_item("PublishingPageImage");

//Out the results
console.log("Publishing Page Content - " + content);
console.log("Publishing Page Image - " + image);
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

```

</script>

## **Output:**

Rollup Image


Rollup Image is a site column created by the Publishing feature. It is used on the Page Content in content roll-ups such as the Content By Search web part.

Target Audiences

Publishing feature. It is used to specify audiences to which this page will be targeted.

Hide physical URLs from search

If checked, the physical URL of this page will not appear in search results. Friendly URLs are generated instead.

Page Content

Office 365 Publishing Page content ...

```

Publishing Page Content - <span style="margin-left: auto; background-color: yellow; font-size: 18px;">Office 365 Publishing Page content ...</span>
Publishing Page Image - 
>

```

## XIII. Working with Property Bag

### A. Create Property Bag Value

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">
$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', ModifyPropertyBag);
});

var oList, oPropBag ;
function ModifyPropertyBag() {

    //Get Client Context and Web object.
    clientContext = new SP.ClientContext.get_current();
    var oWeb = clientContext.get_web();

    //Get property bag collection and set the new key value pair
    oPropBag = oWeb.get_allProperties();
    oPropBag.set_item("Updated Name", "Modified List");
    oWeb.update();

    //Execute the batch
    clientContext.executeQueryAsync(Success,Failure); }

function Success(){
    console.log('Property bag value created');
}

function Failure(sender, args) {
    console.log('Request failed. ' + args.get_message() + '\n' +
args.get_stackTrace());
}
</script>

```

## B. Read Property Bag Value

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', ReadPropertyBag);
});

var oList, oPropBag ;

function ReadPropertyBag() {
//Get Client Context and Web object.
clientContext = new SP.ClientContext.get_current();
var oWeb = clientContext.get_web();

//Get property bag collection
oPropBag = oWeb.get_allProperties();

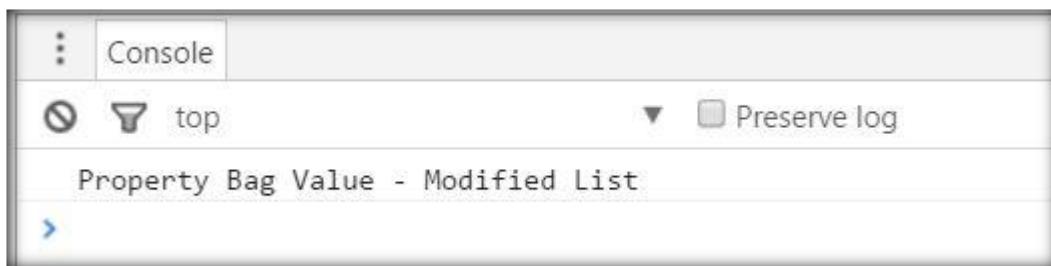
//Load client context and execute the batch
clientContext.load(oPropBag);
clientContext.executeQueryAsync(Success,Failure);
}

function Success() {

//Get specific Property Bag item and display its value
console.log("Property Bag Value - "+oPropBag.get_fieldValues()["Updated
Name"]);
}

function Failure(sender, args) {
    console.log('Request failed. ' + args.get_message() + '\n' +
args.get_stackTrace());
}
</script>

```

**Output:**

The screenshot shows the 'Output' window from Microsoft Visual Studio. The title bar says 'Console'. Below it, there are filter icons: a stop sign, a funnel, and the word 'top'. To the right is a dropdown arrow and a checkbox labeled 'Preserve log'. The main area displays the text 'Property Bag Value - Modified List' followed by a blue chevron icon pointing right.

## XIV. Working with Roles

### A. Get all Roles

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getRoleDef);
});

var roleDefCollection;
function getRoleDef() {

    //Create client context and web object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get role definitions object
    roleDefCollection = oWeb.get_roleDefinitions();

    //Load the client context and execute the batch
    clientContext.load(roleDefCollection);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

    //Instantiate the role definition enumerator collection and loop through
    it
    var roleDefEnumerator = roleDefCollection.getEnumerator();
    console.log("The available role definitions for the web are: ");
    while (roleDefEnumerator.moveNext()) {
        var roleDef = roleDefEnumerator.get_current();
        console.log(roleDef.get_name());
    }
}
```

```

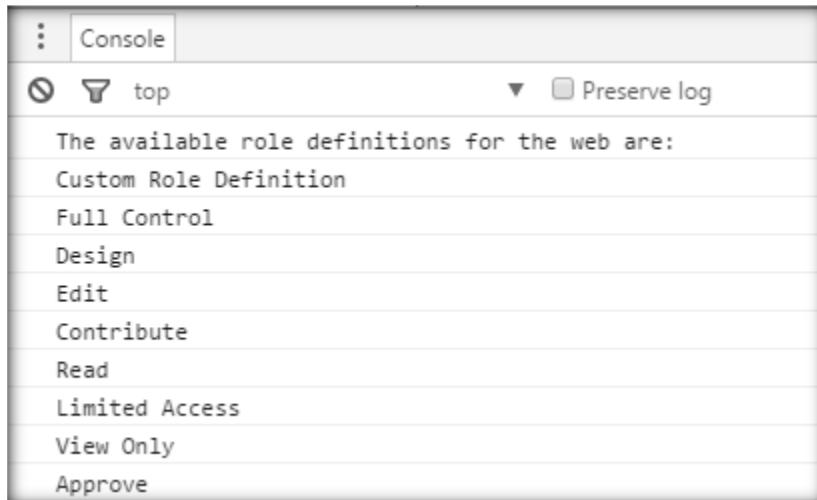
        }
    }

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

### Output:



### B. Create Role Definition

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', createRoleDef);
}

```

```

});;

var roleDefCollection;
function createRoleDef() {

    //Get the client context and site object
    var clientContext = new SP.ClientContext();
    var oSiteColl = clientContext.get_site();

    //Get the web object and create a permissions object
    var oWeb = clientContext.get_web();
    var oPermissions = new SP.BasePermissions();

    //Set the permission masks
    oPermissions.set(SP.PermissionKind.viewListItems);
    oPermissions.set(SP.PermissionKind.addListItems);
    oPermissions.set(SP.PermissionKind.editListItems);
    oPermissions.set(SP.PermissionKind.deleteListItems);

    // Create a new role definition.
    var oRoleDefinitionCreateInfo = new
SP.RoleDefinitionCreationInformation();
    oRoleDefinitionCreateInfo.set_name('Custom Role Definition');
    oRoleDefinitionCreateInfo.set_description('Custom Role Definition to
manage list items');
    oRoleDefinitionCreateInfo.set_basePermissions(oPermissions);
    var roleDefinition =
oSiteColl.get_rootWeb().get_roleDefinitions().add(oRoleDefinitionCreateInfo
);

    //Execute the batch
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

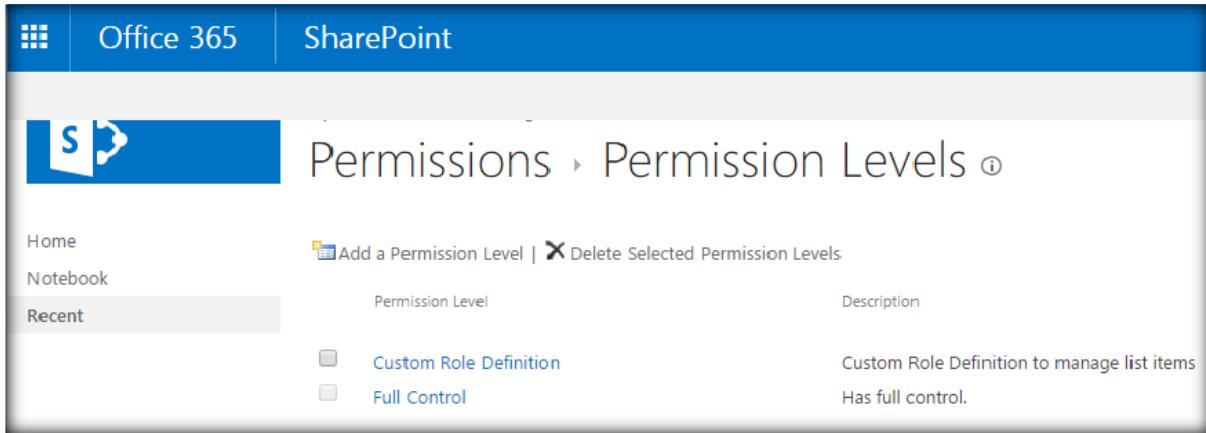
function QuerySuccess() {
    console.log("New Role Definition has been created.");
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## **Output:**



The screenshot shows the SharePoint 'Permissions > Permission Levels' page. The navigation bar at the top includes 'Office 365' and 'SharePoint'. On the left, there's a ribbon with a blue 'S' icon. The main content area has a header 'Permissions > Permission Levels'. Below it, there's a table with columns 'Permission Level' and 'Description'. Two items are listed:

Permission Level	Description
<input type="checkbox"/> Custom Role Definition	Custom Role Definition to manage list items
<input type="checkbox"/> Full Control	Has full control.

## C. Update Role Definition

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', updateRoleDef);
});
var roleDef,clientContext;
function updateRoleDef() {
    //Get client context,web and the existing role definition object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();
    roleDef = oWeb.get_roleDefinitions().getByName("Custom Role Definition");

    //Create a new base permission and assign it to the role definition
    var updatedBasePermissions = new SP.BasePermissions();
    updatedBasePermissions.set(SP.PermissionKind.deleteListItems);
    updatedBasePermissions.set(SP.PermissionKind.createGroups);
    roleDef.set_basePermissions(updatedBasePermissions);
    roleDef.update();
}

```

```

        clientContext.load(roleDef);
        clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
    }

    function QuerySuccess() {
        console.log("Role Definition has been updated.");
    }

    function QueryFailure(sender,args) {
        console.log('Request failed'+ args.get_message());
    }

</script>

```

## **Output:**

Before running the script:

**List Permissions**

- Manage Lists - Create and delete lists, add or remove columns in a list, and add or remove public views of a list.
- Override List Behaviors - Discard or check in a document which is checked out to another user, and change or override settings which allow users to read/edit only their own items
- Add Items - Add items to lists and add documents to document libraries.
- Edit Items - Edit items in lists, edit documents in document libraries, and customize Web Part Pages in document libraries.
- Delete Items - Delete items from a list and documents from a document library.
- View Items - View items in lists and documents in document libraries.

**Site Permissions**

- Manage Permissions - Create and change permission levels on the Web site and assign permissions to users and groups.
- View Web Analytics Data - View reports on Web site usage.
- Create Subsites - Create subsites such as team sites, Meeting Workspace sites, and Document Workspace sites.
- Manage Web Site - Grants the ability to perform all administration tasks for the Web site as well as manage content.
- Add and Customize Pages - Add, change, or delete HTML pages or Web Part Pages, and edit the Web site using a Microsoft SharePoint Foundation-compatible editor.
- Apply Themes and Borders - Apply a theme or borders to the entire Web site.
- Apply Style Sheets - Apply a style sheet (.CSS file) to the Web site.
- Create Groups - Create a group of users that can be used anywhere within the site collection.

After running the script:

**List Permissions**

- Manage Lists - Create and delete lists, add or remove columns in a list, and add or remove public views of a list.
- Override List Behaviors - Discard or check in a document which is checked out to another user, and change or override settings which allow users to read/edit only their own items
- Add Items - Add items to lists and add documents to document libraries.
- Edit Items - Edit items in lists, edit documents in document libraries, and customize Web Part Pages in document libraries.
- Delete Items - Delete items from a list and documents from a document library.
- View Items - View items in lists and documents in document libraries.
- Approve Items - Approve a minor version of a list item or document.

**Site Permissions**

- Manage Permissions - Create and change permission levels on the Web site and assign permissions to users and groups.
- View Web Analytics Data - View reports on Web site usage.
- Create Subsites - Create subsites such as team sites, Meeting Workspace sites, and Document Workspace sites.
- Manage Web Site - Grants the ability to perform all administration tasks for the Web site as well as manage content.
- Add and Customize Pages - Add, change, or delete HTML pages or Web Part Pages, and edit the Web site using a Microsoft SharePoint Foundation-compatible editor.
- Apply Themes and Borders - Apply a theme or borders to the entire Web site.
- Apply Style Sheets - Apply a style sheet (.CSS file) to the Web site.
- Create Groups - Create a group of users that can be used anywhere within the site collection.

**D. Delete Base Permissions****Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

**Script:**

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext',
removeBasePermissions);
});

var oRoleDefinition,clientContext;
function removeBasePermissions() {

    //Get client context and Web object
    clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get the role definition from which the base permission has to be
removed
    oRoleDefinition = oWeb.get_roleDefinitions().getByName("Custom Role
Definition");

    //Load the client context and execute the batch
    clientContext.load(oRoleDefinition);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

    //Get the base permission for the role definition and remove the
particular permission
    var oBasePermissions = oRoleDefinition.get_basePermissions();
    oBasePermissions.clear(SP.PermissionKind.deleteListItems);

    //Update the role definition
    oRoleDefinition.set_basePermissions(oBasePermissions);
    oRoleDefinition.update();

    //Load the client context and execute the batch
    clientContext.load(oRoleDefinition);
    clientContext.executeQueryAsync(SecondQuerySuccess, SecondQueryFailure);
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}
function SecondQuerySuccess() {
    console.log('The role definition has been updated.');
}

function SecondQueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}
</script>
```

## **Output:**

Before running the script:

**List Permissions**

- Manage Lists - Create and delete lists, add or remove columns in a list, and add or remove public views of a list.
- Override List Behaviors - Discard or check in a document which is checked out to another user, and change or override settings which allow users to read/edit only their own items
- Add Items - Add items to lists and add documents to document libraries.
- Edit Items - Edit items in lists, edit documents in document libraries, and customize Web Part Pages in document libraries.
- Delete Items - Delete items from a list and documents from a document library.
- View Items - View items in lists and documents in document libraries.

After running the script:

**List Permissions**

- Manage Lists - Create and delete lists, add or remove columns in a list, and add or remove public views of a list.
- Override List Behaviors - Discard or check in a document which is checked out to another user, and change or override settings which allow users to read/edit only their own items
- Add Items - Add items to lists and add documents to document libraries.
- Edit Items - Edit items in lists, edit documents in document libraries, and customize Web Part Pages in document libraries.
- Delete Items - Delete items from a list and documents from a document library.
- View Items - View items in lists and documents in document libraries.

## **E. Remove Role Definition from User**

### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

## Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', removeRoleFromUser);
});

function removeRoleFromUser() {
    //Get the client context and web object
    var clientContext = new SP.ClientContext.get_current();
    var oWeb = clientContext.get_web();

    //Get the user and role definition object
    var oUser= oWeb.get_currentUser();
    var roleDef = oWeb.get_roleDefinitions().getByName("Custom Role
Definition");

    //Get the role assignments for the current user
    var roleAssignmentColl = oWeb.get_roleAssignments();
    var roleAssignment = roleAssignmentColl.getByPrincipal(oUser)

    //Remove the role from the user
    var roleAssignmentBindingColl =
roleAssignment.get_roleDefinitionBindings();
    roleAssignmentBindingColl.remove(roleDef);
    roleAssignment.importRoleDefinitionBindings(roleAssignmentBindingColl);
    roleAssignment.update();

    //Execute the batch
    clientContext.executeQueryAsync(Succeeded,Failure);
}

function Succeeded() {
    console.log('The role definition has been removed from the user.');
}

function Failure(sender, args) {
    console.log('Request failed'+ args.get_message());
}
</script>

```

## Output:

Before running the script:

## Updated Web Site: Check Permissions

### Check Permissions

To check permissions for a user or group, enter their name or e-mail address.

User/Group:




Permission levels given to Priyanjan KS (i:0#.f|membership|priyanjan@ctsplayground.onmicrosoft.com)

Custom Role      Given directly  
Definition, Limited Access

Full Control,      Given through the "Playground Owners" group.  
Limited Access

After running the script:

## Updated Web Site: Check Permissions

### Check Permissions

To check permissions for a user or group, enter their name or e-mail address.

User/Group:




Permission levels given to Priyanjan KS (i:0#.f|membership|priyanjan@ctsplayground.onmicrosoft.com)

Limited Access      Given directly  
Full Control,      Given through the "Playground Owners" group.  
Limited Access

## F. Remove Role Definition from Group

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

## Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', removeRoleFromGroup);
});

function removeRoleFromGroup() {

    //Get the client context and web object
    var clientContext = new SP.ClientContext.get_current();
    var oWeb = clientContext.get_web();

    //Get the group and the role definition object
    var oGroup = oWeb.get_siteGroups().getByName("HR");
    var roleDef = oWeb.get_roleDefinitions().getByName("Custom Role
Definition");

    //Get the role assignments collection
    var roleAssignmentColl = oWeb.get_roleAssignments();
    var roleAssignment = roleAssignmentColl.getByPrincipal(oGroup)

    //Get the role definition binding from the role assignment and remove the
role definition from it
    var roleAssignmentBindingColl =
roleAssignment.get_roleDefinitionBindings();
    roleAssignmentBindingColl.remove(roleDef);
    roleAssignment.importRoleDefinitionBindings(roleAssignmentBindingColl);
    roleAssignment.update();

    //Execute the batch
    clientContext.executeQueryAsync(Succeeded,Failure);
}

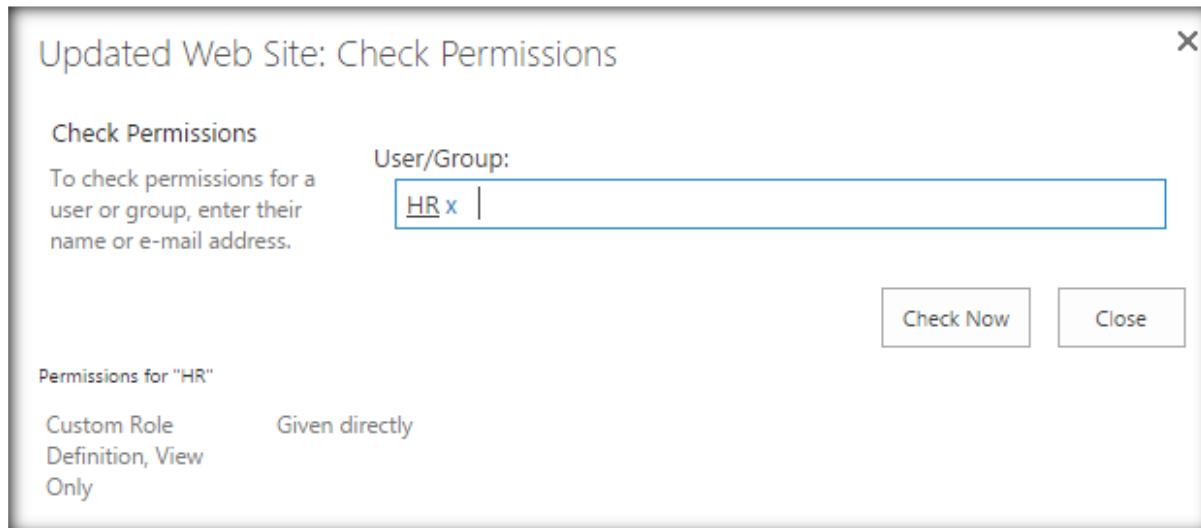
function Succeeded() {
    console.log('The role definition has been removed from the group.');
}

function Failure(sender, args) {
    console.log('Request failed'+ args.get_message());
}
</script>

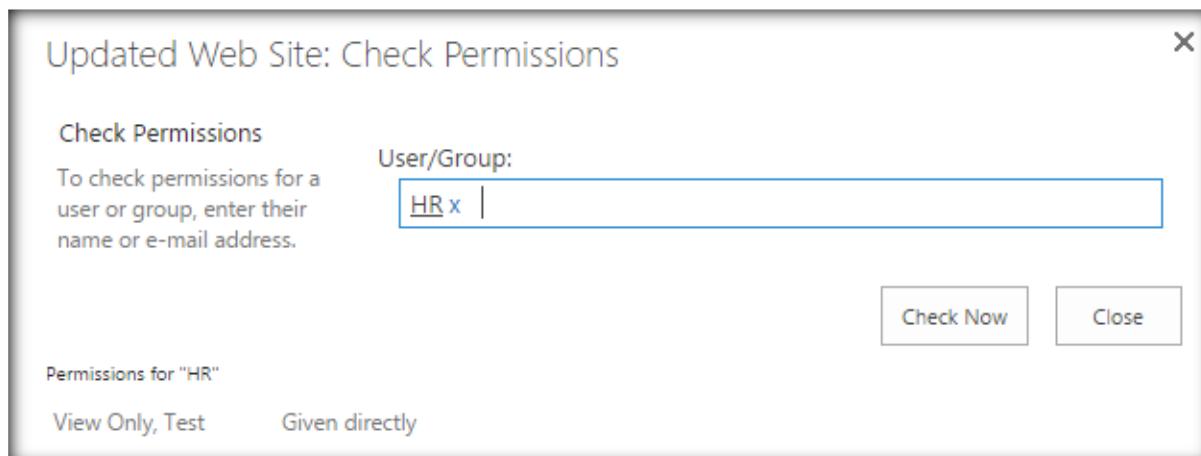
```

## Output:

Before running the script:



After running the script:



## G. Add Role Definition to User

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', addRoleToUser);
});

function addRoleToUser() {

    //Get client context and web object
    var clientContext = new SP.ClientContext.get_current();
    var oWeb = clientContext.get_web();

    //Get current user and role definition object
    var oUser = oWeb.get_currentUser();
    var roleDef = oWeb.get_roleDefinitions().getByName("Custom Role
Definition");
    var collRoleDefinitionBinding =
SP.RoleDefinitionBindingCollection.newObject(clientContext);

    //Add the role definition to the user's role definition collection
    collRoleDefinitionBinding.add(roleDef);
    oWeb.get_roleAssignments().add(oUser, collRoleDefinitionBinding);

    //Execute the batch
    clientContext.executeQueryAsync(Succeeded, Failure);
}

function Succeeded() {
    console.log('The role definition has been added to the user.');
}

function Failure(sender, args) {
    console.log('Request failed'+ args.get_message());
}
</script>

```

### Output:

Before running the script:

### Updated Web Site: Check Permissions

**Check Permissions**  
To check permissions for a user or group, enter their name or e-mail address.

User/Group:

Permission levels given to Priyanjan KS (i:0#.f|membership|priyanjan@ctsplayground.onmicrosoft.com)

Limited Access	Given directly
Full Control,	Given through the "Playground Owners" group.
Limited Access	

After running the script:

### Updated Web Site: Check Permissions

**Check Permissions**  
To check permissions for a user or group, enter their name or e-mail address.

User/Group:

Permission levels given to Priyanjan KS (i:0#.f|membership|priyanjan@ctsplayground.onmicrosoft.com)

Custom Role	Given directly
Definition, Limited Access	
Full Control,	Given through the "Playground Owners" group.
Limited Access	

## H. Add Role Definition to Group

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', addRoleToGroup);
});

function addRoleToGroup() {

    //Get the client context and web object
    var clientContext = new SP.ClientContext.get_current();
    var oWeb = clientContext.get_web();

    //Get the group and role definition object
    var oGroup = oWeb.get_siteGroups().getByName("HR");
    var roleDef = oWeb.get_roleDefinitions().getByName("Custom Role
Definition");
    var collRoleDefinitionBinding =
SP.RoleDefinitionBindingCollection.newObject(clientContext);

    //Add the role definition to the group
    collRoleDefinitionBinding.add(roleDef);
    oWeb.get_roleAssignments().add(oGroup, collRoleDefinitionBinding);

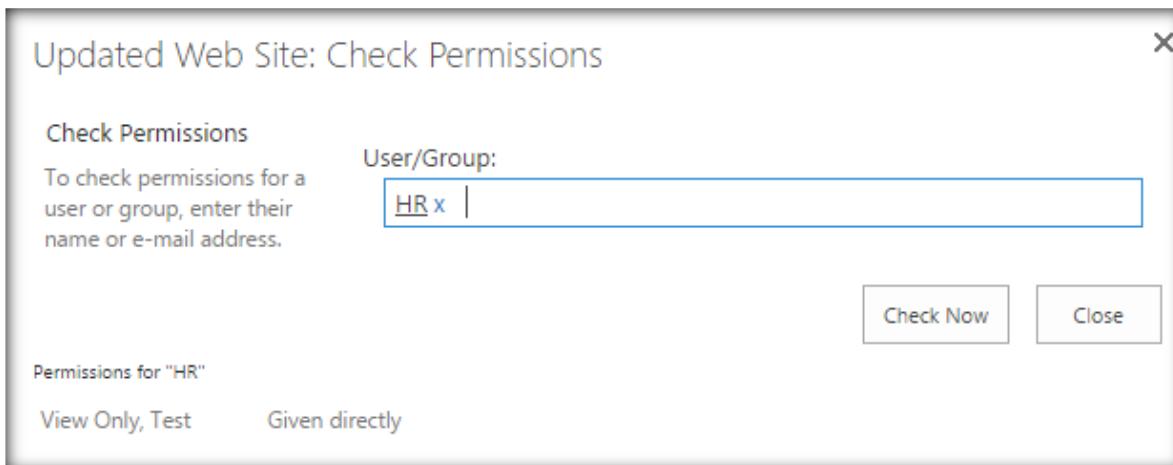
    //Execute the batch
    clientContext.executeQueryAsync(Succeeded, Failure);
}

function Succeeded() {
    console.log('The role definition has been added to the group.');
}

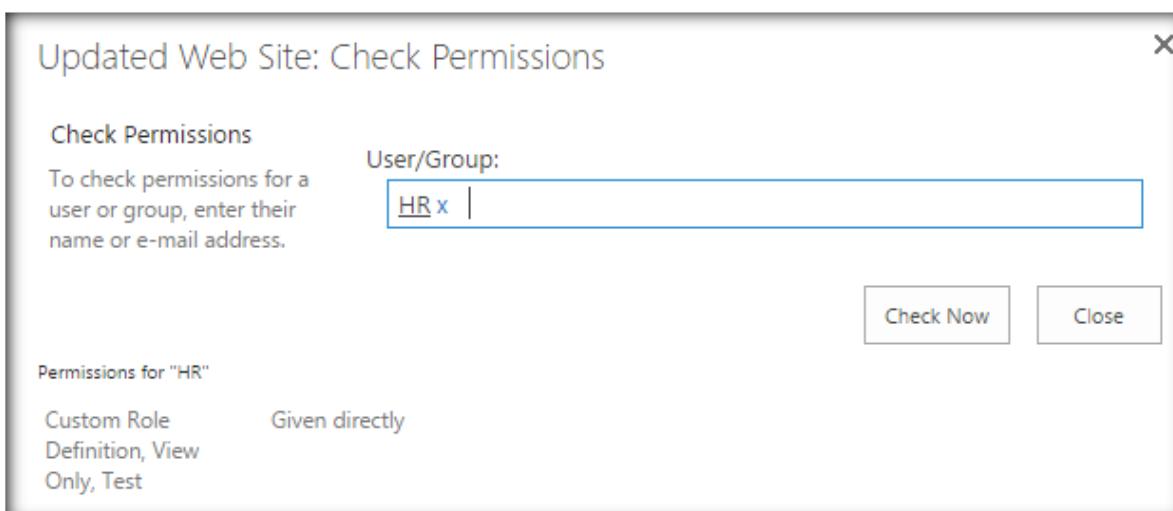
function Failure(sender, args) {
    console.log('Request failed'+ args.get_message());
}
</script>
```

## **Output:**

Before running the script:



After running the script:



## **I. Remove Role Definition**

### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', removeRoleDefinition);
});

var roleDef,clientContext;
function removeRoleDefinition() {

    //Get the client context and web object
    clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get the role definition and delete it
    roleDef = oWeb.get_roleDefinitions().getByName("Custom Role Definition");
    roleDef.deleteObject();

    //Execute the batch
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

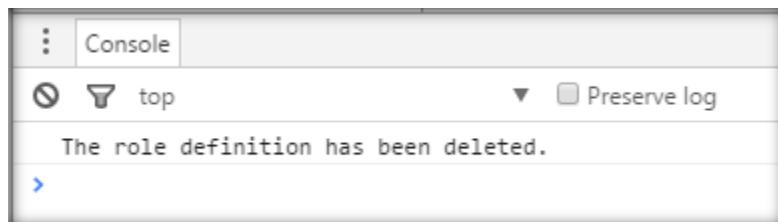
function QuerySuccess() {
    console.log('The role definition has been deleted.');
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

### Output:



## XV. Working with Views

### A. Create View

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', createListView);
});

var oListViews;
function createListView() {

    //Get client context,web and list object
    var clientContext = new SP.ClientContext();
    var oWebsite = clientContext.get_web();
    var oList = oWebsite.get_lists().getByTitle('Products');

    //Set the view fields
    var viewFields= new
Array('Total_x0020_Sales','Sales_x0020_Target','Product_x0020_Name');
    var viewType = new SP.ViewType();

    //Create view using ViewCreationInformation object
    var creationInfo = new SP.ViewCreationInformation();
    creationInfo.set_title("CustomProductView");
    creationInfo.set_setAsDefaultView("true");
    creationInfo.set_rowLimit("10");
    creationInfo.set_personalView("false");
    creationInfo.set_viewFields(viewFields);

    //set caml query so that the view shows only a subset of items
    var camlQuery = new SP.CamlQuery();
    var query = "<Where><IsNotNull><FieldRef Name='Product_x0020_Name'>
```

```

/></IsNotNull></Where>";
    camlQuery.set_viewXml(query);
    creationInfo.set_query(camlQuery);
    oListViews=oList.get_views().add(creationInfo);

    //Load the client context and execute the batch
    clientContext.load(oListViews);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    console.log("Views created successfully!");
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## Output:

Products			
<a href="#">+ new item or edit this list</a>			
	CustomProductView	All Items	ColorCodeTarget
✓	Product Name	Total Sales	Sales Target
	Ford Ecosport	3,500	3,000
	Hyundai Creta	6,500	7,500
	Hyundai i10	1,400	3,000
	Hyundai i20	12,000	10,000
	Nissan Terrano	500	1,500
	Renault Duster	1,300	1,500
	Suzuki Alto	10,000	11,000
	Suzuki Baleno	7,000	11,000
	Suzuki Wagon R	6,000	5,000

## B. Get All List Views

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getListViews);
});

var oListViews;
function getListViews() {

    //Get the client context,web and list object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();
    var oList = oWeb.get_lists().getByTitle('Products');

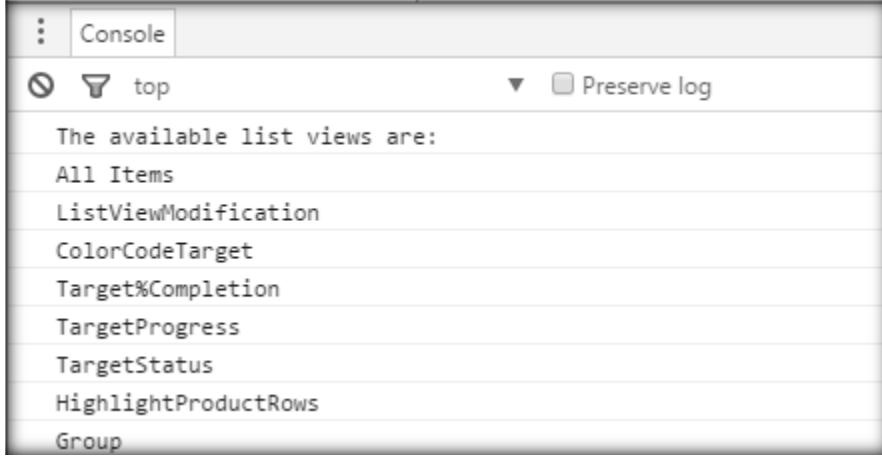
    //Get the list view and load it to client context and execute the batch
    oListViews=oList.get_views();
    clientContext.load(oListViews);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    //Get the enumerator collection of list view and loop through it
    var enumerator =oListViews.getEnumerator();
    console.log("The available list views are:");
    while (enumerator.moveNext()) {
        console.log(enumerator.get_current().get_title() + '\n');
    }
}

function QueryFailure() {
    console.log('Request failed'+ args.get_message());
}

</script>
```

## Output:



The available list views are:

- All Items
- ListViewModification
- ColorCodeTarget
- Target%Completion
- TargetProgress
- TargetStatus
- HighlightProductRows
- Group

## C. Update List Views

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', updateListView);
});

var oListViews;
function updateListView() {

    //Get the client context,web and list object
    var clientContext = new SP.ClientContext();
    var oWebsite = clientContext.get_web();
    var oList = oWebsite.get_lists().getByTitle('Products');

    //Get the view object
}
```

```

var oView = oList.get_views().getByTitle('All Items');
oView.set_defaultView("true");

//Set a new caml query to update the items returned in the view
var camlQuery = new SP.CamlQuery();
var query = "<Where><IsNotNull><FieldRef Name='Product_x0020_Name'></IsNotNull></Where>";
camlQuery.set_viewXml(query);
oView.set_viewQuery(camlQuery);
oView.update();

//Load the client context and execute the batch
clientContext.load(oView);
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    console.log('The view has been updated.');
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## Output:

Products					
<span style="color: blue;">⊕</span> new item or edit this list					
	All Items	ColorCodeTarget	CustomProductView	...	Find an i
✓	ID	Product Brand	Product Name	Total Sales	Sales Target
1	Ford		Ford Ecosport	3,500	3,000
2	Hyundai		Hyundai Creta	6,500	7,500
3	Hyundai		Hyundai i10	1,400	3,000
4	Hyundai		Hyundai i20	12,000	10,000
5	Nissan		Nissan Terrano	500	1,500
6	Renault		Renault Duster	1,300	1,500
7	Suzuki		Suzuki Alto	10,000	11,000
8	Suzuki		Suzuki Baleno	7,000	11,000
9	Suzuki		Suzuki Wagon R	6,000	5,000

## D. Get Fields from List View

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getListViewFields);
});


var oFieldColl;
function getListViewFields() {

    //Get the client context,web and list object
    var clientContext = new SP.ClientContext();
    var oWebsite = clientContext.get_web();
    var oList = oWebsite.get_lists().getByTitle('Products');

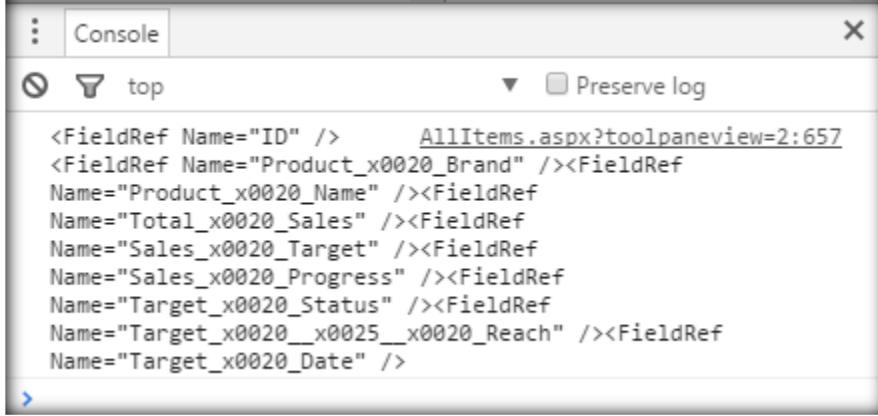
    //Get all the fields in the view
    var oView = oList.get_views().getByTitle('All Items');
    oFieldColl = oView.get_viewFields();

    //Load the client context and execute the batch
    clientContext.load(oFieldColl);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    //Get the field collection schema
    console.log(oFieldColl.get_schemaXml());
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}
</script>
```

## Output:



The screenshot shows the SharePoint browser console window titled "Console". It displays the output of a script, specifically the list of field references being processed. The output includes field names like "ID", "Product\_x0020\_Brand", "Product\_x0020\_Name", "Total\_x0020\_Sales", "Sales\_x0020\_Target", "Sales\_x0020\_Progress", "Target\_x0020\_Status", "Target\_x0020\_x0025\_x0020\_Reach", and "Target\_x0020\_Date". The log also indicates the file path "AllItems.aspx?toolpaneview=2:657".

```
<FieldRef Name="ID" />      AllItems.aspx?toolpaneview=2:657
<FieldRef Name="Product_x0020_Brand" /><FieldRef
Name="Product_x0020_Name" /><FieldRef
Name="Total_x0020_Sales" /><FieldRef
Name="Sales_x0020_Target" /><FieldRef
Name="Sales_x0020_Progress" /><FieldRef
Name="Target_x0020_Status" /><FieldRef
Name="Target_x0020_x0025_x0020_Reach" /><FieldRef
Name="Target_x0020_Date" />
```

## E. Set JSLink in List View

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', setJSLink);
});

var oView;
function setJSLink() {

    //Get the client context,web and list object
    var clientContext = new SP.ClientContext();
    var oWebsite = clientContext.get_web();
    var oList = oWebsite.get_lists().getByTitle('Products');

    //Get the view object and set the jsLink property
    oView = oList.get_views().getByTitle('CustomProductView');
    oView.set_jsLink("~site/siteassets/ListViewModification.js");
}
```

```

oView.update();

//Load the client context and execute the batch
clientContext.load(oView);
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    console.log("JSLink has been set to the view.");
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## Output:

Before setting JSLink:

Products			
<span style="color: blue;">⊕</span> new item or edit this list			
	All Items	ColorCodeTarget	CustomProductView
✓	Product Name	Total Sales	Sales Target
	Ford Ecosport	3,500	3,000
	Hyundai Creta	6,500	7,500
	Hyundai i10	1,400	3,000
	Hyundai i20	12,000	10,000
	Nissan Terrano	500	1,500
	Renault Duster	1,300	1,500
	Suzuki Alto	10,000	11,000
	Suzuki Baleno	7,000	11,000
	Suzuki Wagon R	6,000	5,000

After setting JSLink:

## Products

[+ new item](#) or edit this list

All Items   ColorCodeTarget   **CustomProductView**   ...   Find an item 

Product Name : Ford Ecosport . The sales target for Ford Ecosport is 3,000 . The Current Sales Count is 3,500 units .

Product Name : Hyundai Creta . The sales target for Hyundai Creta is 7,500 . The Current Sales Count is 6,500 units .

Product Name : Hyundai i10 . The sales target for Hyundai i10 is 3,000 . The Current Sales Count is 1,400 units .

Product Name : Hyundai i20 . The sales target for Hyundai i20 is 10,000 . The Current Sales Count is 12,000 units .

Product Name : Nissan Terrano . The sales target for Nissan Terrano is 1,500 . The Current Sales Count is 500 units .

Product Name : Renault Duster . The sales target for Renault Duster is 1,500 . The Current Sales Count is 1,300 units .

Product Name : Suzuki Alto . The sales target for Suzuki Alto is 11,000 . The Current Sales Count is 10,000 units .

Product Name : Suzuki Baleno . The sales target for Suzuki Baleno is 11,000 . The Current Sales Count is 7,000 units .

Product Name : Suzuki Wagon R . The sales target for Suzuki Wagon R is 5,000 . The Current Sales Count is 6,000 units .

## F. Delete View

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', deleteListView);
});

function deleteListView() {
    //Get the client context, web and list object
}
```

```
var clientContext = new SP.ClientContext();
var oWeb = clientContext.get_web();
var oList = oWeb.get_lists().getByTitle('Products');

//Get the view object to delete
var oView = oList.get_views().getByTitle('CustomProductView');
oView.deleteObject();

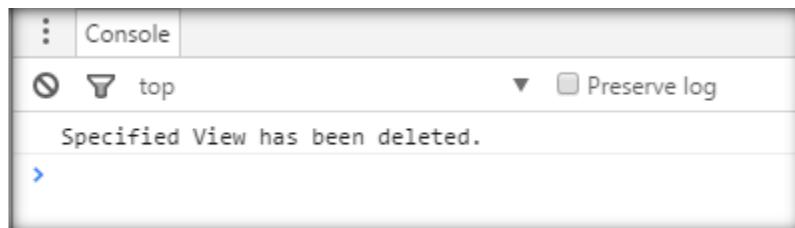
//Execute the batch
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    console.log('Specified View has been deleted.');
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>
```

## Output:



## XVI. Working with Web

### A. Add Language

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', addLanguage);
});

var oWeb;
function addLanguage() {

    //get the client context and web object
    var clientContext = new SP.ClientContext();
    oWeb = clientContext.get_web();

    //Set the multilingual property to true and add a language say dutch to the web
    oWeb.set_isMultilingual(true);
    oWeb.addSupportedUILanguage(1043); //Dutch
    oWeb.update();

    //Load the client context and execute the batch
    clientContext.load(oWeb, 'SupportedUILanguageIds');
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

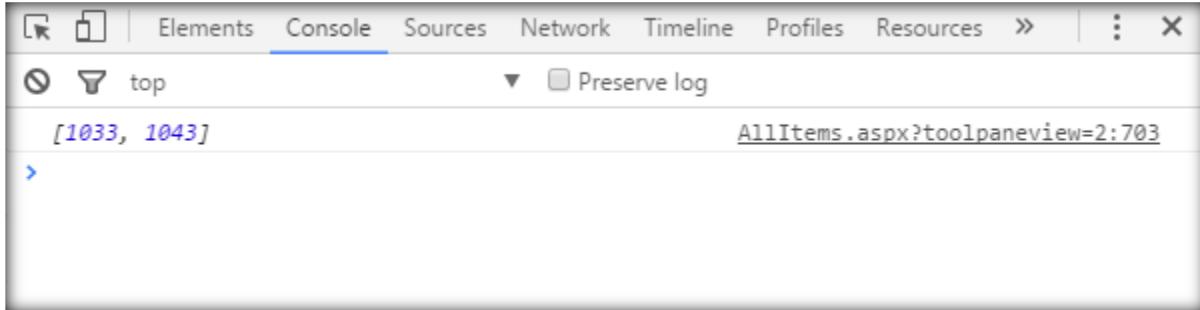
    //Get the supported languages and display it in the console
    var supportedLanguages = oWeb.get_supportedUILanguageIds();
    console.log(supportedLanguages);
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## **Output:**



## **B. Get Current Locale**

### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### **Script:**

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getCurrentLocale);
});

var oLanguageColl ;
function getCurrentLocale() {

    //Get client context and installed language
    var clientContext = new SP.ClientContext.get_current();
    oLanguageColl = SP.ServerSettings.getGlobalInstalledLanguages(clientContext, 15);

    //Execute the batch
    clientContext.executeQueryAsync(Success, Failure);
}

function Success()
{

    //Get the current locale
    var currentLocale = parseInt(SP.Res.lcid);
    var currentLanguage = oLanguageColl.filter(function(lang) {
```

```

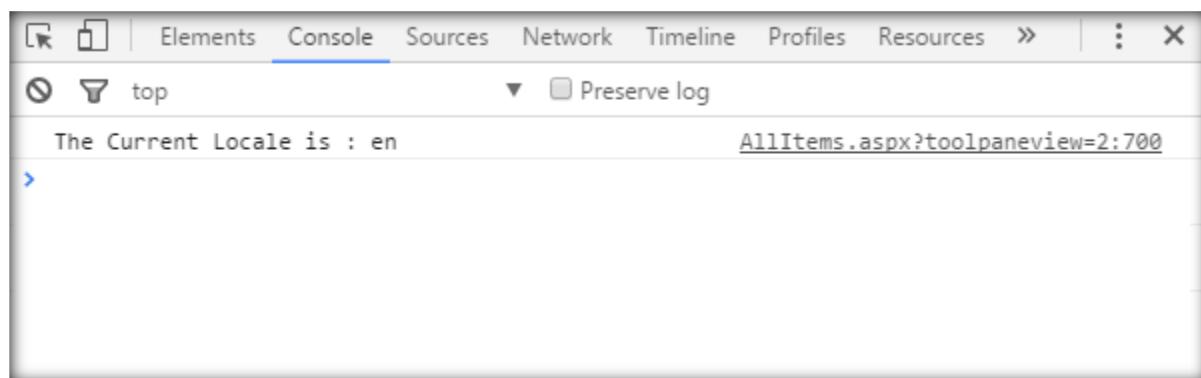
        return lang.get_lcid() === currentLocale;
    })[0];
var currentLanguageTag = currentLanguage.get_languageTag();
var currentLanguageCode = currentLanguageTag.split('-')[0];
console.log("The Current Locale is : "+currentLanguageCode);

}

function Failure(sender, args) {
    console.log('Request failed'+ args.get_message());
}
</script>

```

### **Output:**



### **C. Get Time Zone**

#### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### **Script:**

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getTimeZone);
});

var timeZone;
function getTimeZone() {

    //Get the client context and web object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

```

```

//Get time zone
timeZone = oWeb.getRegionalSettings().get_timeZone();

//Load the client context and execute the batch
clientContext.load(timeZone);
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

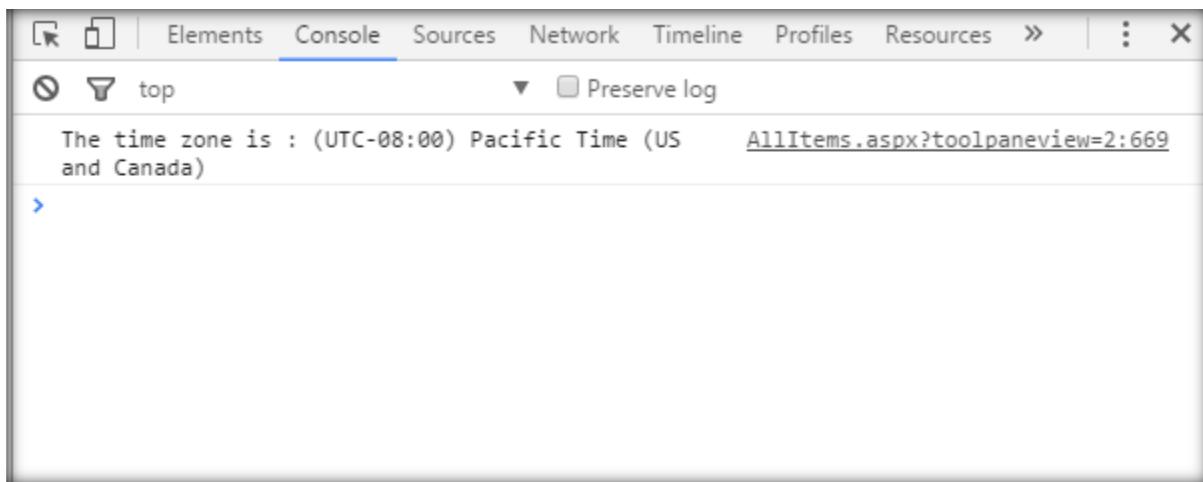
function QuerySuccess() {
    console.log("The time zone is : "+timeZone.get_description());
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## Output:



## D. Get Web Language

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getWebLanguage);
});

var oWeb;
function getWebLanguage() {

    //Get the client context and web object
    var clientContext = new SP.ClientContext();
    oWeb = clientContext.get_web();

    //Load the client context and execute the batch
    clientContext.load(oWeb);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

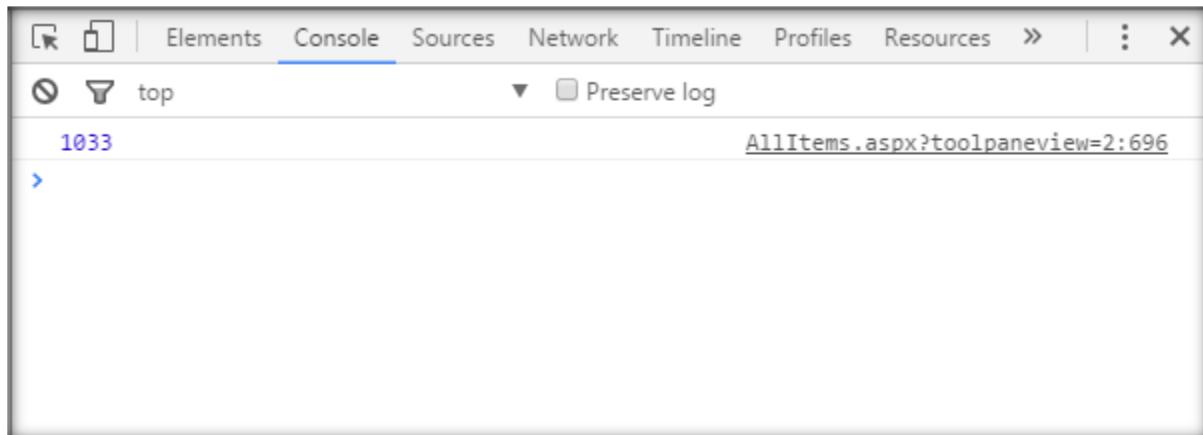
    //Display the web language code
    console.log( oWeb.get_language());
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## Output:



## E. Get Specific Web Properties

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getWebProperties);
});

var oWeb;
function getWebProperties() {

    //Get the client context and web object
    var clientContext = new SP.ClientContext.get_current();
    oWeb = clientContext.get_web();

    //Load the client context and execute the batch
    clientContext.load(oWeb);
    clientContext.executeQueryAsync(Success,Failure);
}

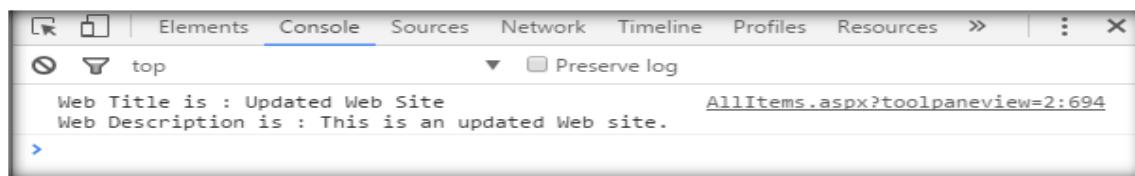
function Success() {
    console.log("Web Title is : "+oWeb.get_title()+'\n'+ "Web Description is : "+
oWeb.get_description());
}

function Failure(sender, args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

### Output:



## F. Get All Web Properties

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', readProperties);
});

var oWebProperties;
function readProperties() {

    //Get client context and web object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get the web properties
    oWebProperties = oWeb.get_allProperties()

    //Get the client context and execute the batch
    clientContext.load(oWebProperties);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

    //Get web property field value collection and loop through it
    var properties = oWebProperties.get_fieldValues();
    console.log("The web properties are : ");
    for (property in properties)
    {
        var propertyName = property;
        var propertyValue = properties[property];

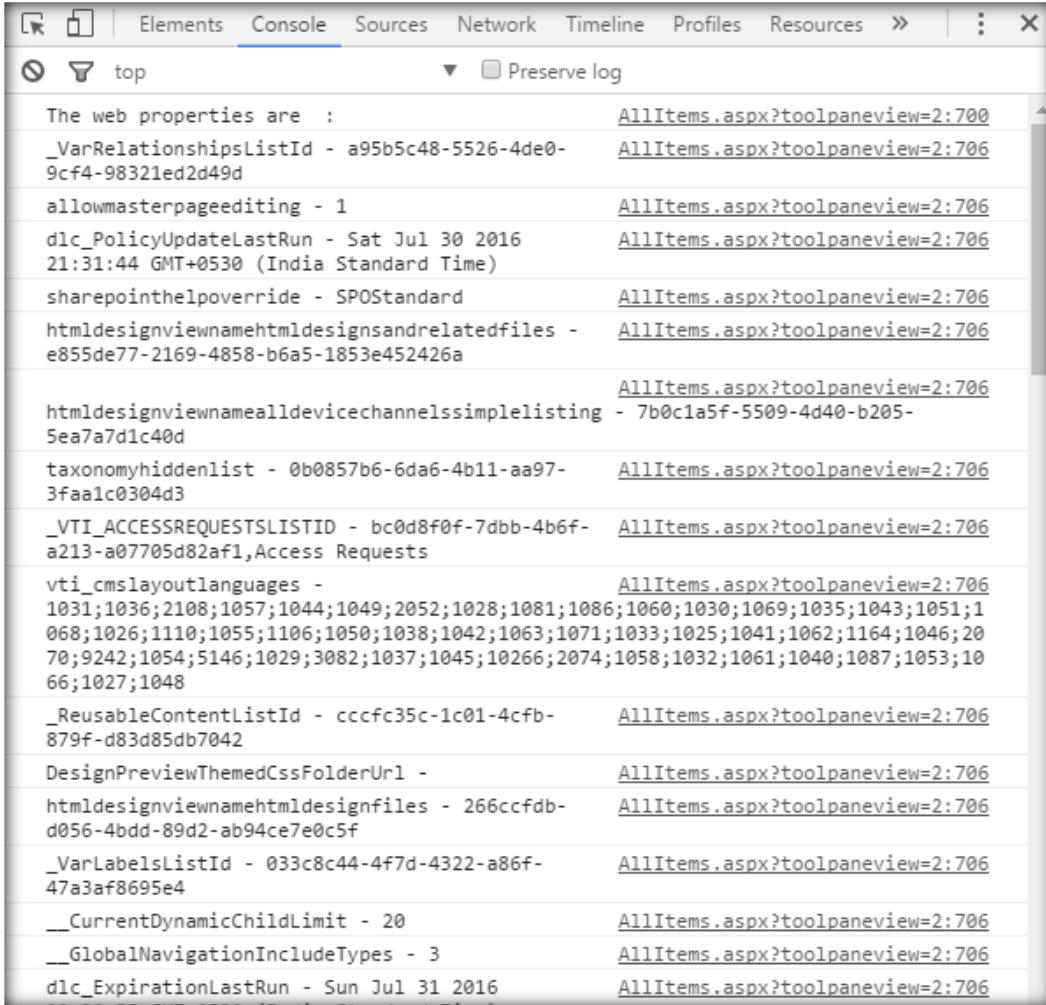
        console.log(propertyName + " - " + propertyValue);
    }
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## **Output:**



The web properties are : [AllItems.aspx?toolpaneview=2:700](#)

\_VarRelationshipsListId - a95b5c48-5526-4de0-9cf4-98321ed2d49d [AllItems.aspx?toolpaneview=2:706](#)

allowmasterpageediting - 1 [AllItems.aspx?toolpaneview=2:706](#)

dlc\_PolicyUpdateLastRun - Sat Jul 30 2016 21:31:44 GMT+0530 (India Standard Time) [AllItems.aspx?toolpaneview=2:706](#)

sharepointhelpoverride - SPOStandard [AllItems.aspx?toolpaneview=2:706](#)

htmldesignviewnamehtmldesignsandrelatedfiles - e855de77-2169-4858-b6a5-1853e452426a [AllItems.aspx?toolpaneview=2:706](#)

htmldesignviewnamealldevicechannelssimplelisting - 7b0c1a5f-5509-4d40-b205-5ea7a7d1c40d [AllItems.aspx?toolpaneview=2:706](#)

taxonomyhiddenlist - 0b0857b6-6da6-4b11-aa97-3faa1c0304d3 [AllItems.aspx?toolpaneview=2:706](#)

\_VTI\_ACCESSREQUESTSLISTID - bc0d8f0f-7dbb-4b6f-a213-a07705d82af1,Access Requests [AllItems.aspx?toolpaneview=2:706](#)

vti\_cmstablelanguages - 1031;1036;2108;1057;1044;1049;2052;1028;1081;1086;1060;1030;1069;1035;1043;1051;1068;1026;1110;1055;1106;1050;1038;1042;1063;1071;1033;1025;1041;1062;1164;1046;2070;9242;1054;5146;1029;3082;1037;1045;10266;2074;1058;1032;1061;1040;1087;1053;1066;1027;1048 [AllItems.aspx?toolpaneview=2:706](#)

\_ReusableContentListId - cccfc35c-1c01-4cfb-879f-d83d85db7042 [AllItems.aspx?toolpaneview=2:706](#)

DesignPreviewThemedCssFolderUrl - [AllItems.aspx?toolpaneview=2:706](#)

htmldesignviewnamehtmldesignfiles - 266ccfdb-d056-4bdd-89d2-ab94ce7e0c5f [AllItems.aspx?toolpaneview=2:706](#)

\_VarLabelsListId - 033c8c44-4f7d-4322-a86f-47a3af8695e4 [AllItems.aspx?toolpaneview=2:706](#)

\_CurrentDynamicChildLimit - 20 [AllItems.aspx?toolpaneview=2:706](#)

\_GlobalNavigationIncludeTypes - 3 [AllItems.aspx?toolpaneview=2:706](#)

dlc\_ExpirationLastRun - Sun Jul 31 2016 [AllItems.aspx?toolpaneview=2:706](#)

## **G. Remove Language**

### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', removeLanguage);
});

var oWeb;
function removeLanguage() {

    //Get client context and web object
    var clientContext = new SP.ClientContext();
    oWeb = clientContext.get_web();

    //Remove the language by ID
    oWeb.set_isMultilingual(true);
    oWeb.removeSupportedUILanguage(1043); //Dutch
    oWeb.update();

    //Load the client context and execute the batch
    clientContext.load(oWeb, 'SupportedUILanguageIds');
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

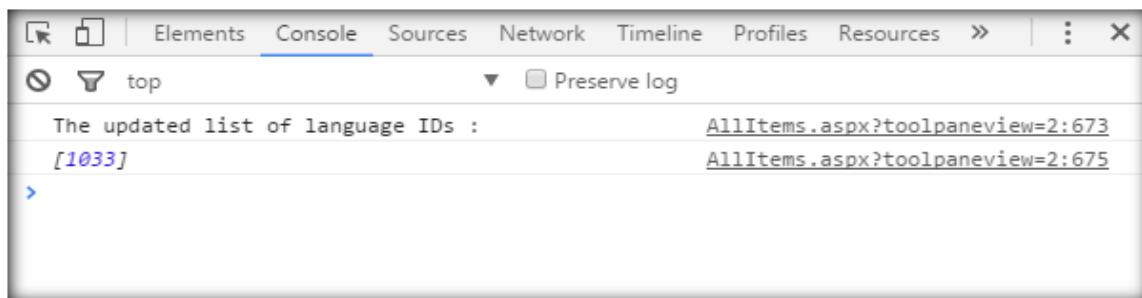
    //Get the updated list of language IDs
    console.log("The updated list of language IDs :");
    var supportedLanguages = oWeb.get_supportedUILanguageIds();
    console.log(supportedLanguages);
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

### Output:



## H. Apply Theme

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', applyTheme);
});

var oWeb;
function applyTheme() {

    //Get the client context and web object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

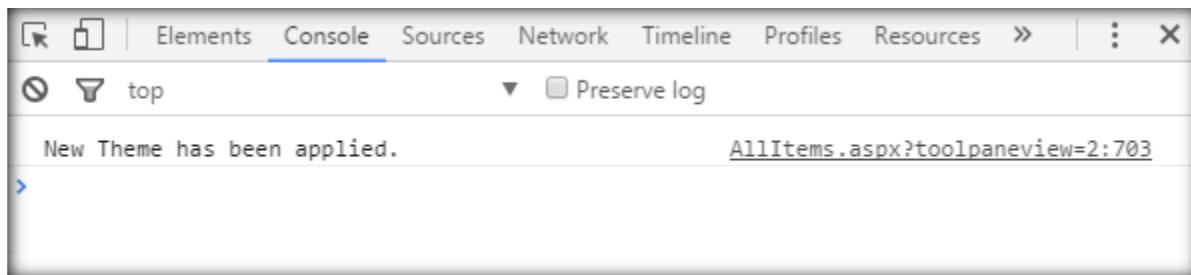
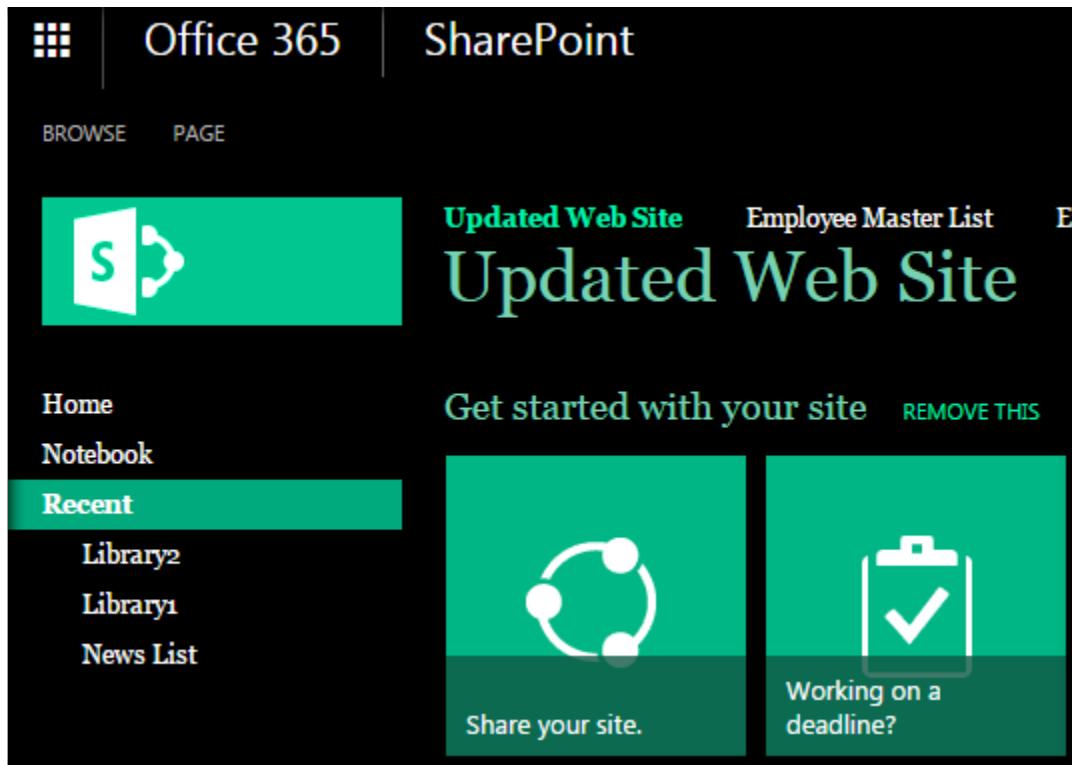
    //Get the server url for the color palette and the font
    var colorUrl =
_spPageContextInfo.siteServerRelativeUrl+"/_catalogs/theme/15/Palette009.spcolor";
    var fontUrl =
_spPageContextInfo.siteServerRelativeUrl+"/_catalogs/theme/15/fontscheme002.spfont";

    //Apply the theme
    oWeb.applyTheme(colorUrl , fontUrl , null, true);

    //Load the client context and execute the batch
    clientContext.load(oWeb);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    console.log("New Theme has been applied.");
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}
</script>
```

**Output:****I. Get Subwebs****Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getSubWeb);
});

var oSubWebs;
function getSubWeb() {

    //Get the client context,web and the sub web collection
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();
    oSubWebs= oWeb.get_webs();

    //Load the client context and execute the batch
    clientContext.load(oSubWebs);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

    //Get the subweb enumerator collection and loop through it
    var webEnumerator = oSubWebs.getEnumerator();
    console.log("The subwebs are :");
    while (webEnumerator.moveNext()) {
        var oWeb = webEnumerator.get_current();
        console.log(oWeb.get_title());
    }
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

### Output:



## J. Get Features List

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getFeatures);
});

var featureColl;
function getFeatures() {

    //Get the client context and site object
    var clientContext = new SP.ClientContext.get_current();
    var site = clientContext.get_site();

    //Get the list of the site features
    featureColl = site.get_features();

    //Load the client context and execute the batch
    clientContext.load(featureColl, 'Include(DisplayName,DefinitionId)');
    clientContext.executeQueryAsync(Success,Failure);
}

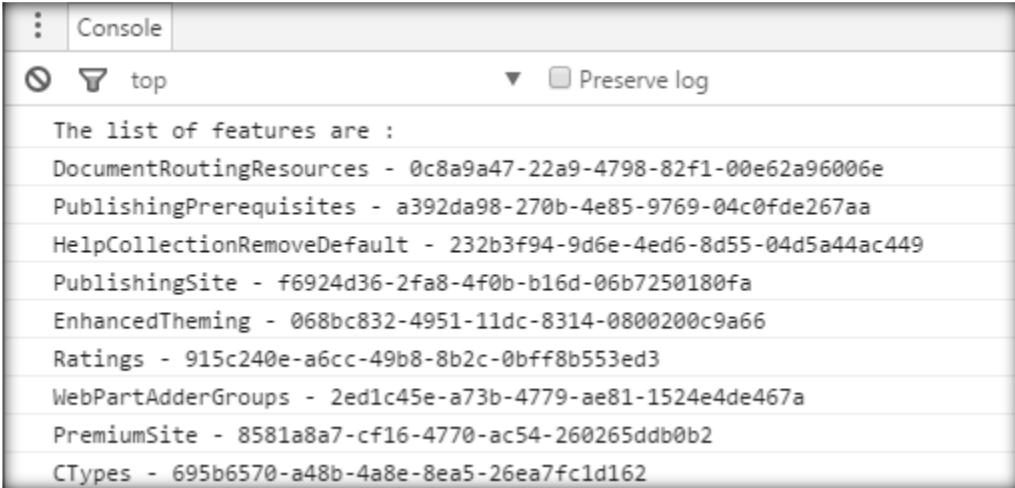
function Success() {

    //Get the features enumerator collection and loop through it
    var featureEnumerator = featureColl.getEnumerator();
    console.log("The list of features are : ");
    while (featureEnumerator.moveNext()) {
        console.log(featureEnumerator.get_current().get_displayName() + " - " +
featureEnumerator.get_current().get_definitionId());
    }
}

function Failure(sender, args) {
    console.log('Request failed'+ args.get_message());
}

</script>
```

## Output:



The list of features are :

- DocumentRoutingResources - 0c8a9a47-22a9-4798-82f1-00e62a96006e
- PublishingPrerequisites - a392da98-270b-4e85-9769-04c0fde267aa
- HelpCollectionRemoveDefault - 232b3f94-9d6e-4ed6-8d55-04d5a44ac449
- PublishingSite - f6924d36-2fa8-4f0b-b16d-06b7250180fa
- EnhancedTheming - 068bc832-4951-11dc-8314-0800200c9a66
- Ratings - 915c240e-a6cc-49b8-8b2c-0bff8b553ed3
- WebPartAdderGroups - 2ed1c45e-a73b-4779-ae81-1524e4de467a
- PremiumSite - 8581a8a7-cf16-4770-ac54-260265ddb0b2
- CTypes - 695b6570-a48b-4a8e-8ea5-26ea7fc1d162

## K. Get Server Date and Time

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getServerTime);
});

var timeZone;
function getServerTime() {

    //Get the client context,web and timezone object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();
    timeZone = oWeb.getRegionalSettings().get_timeZone();

    //Load the client context and execute the batch
    clientContext.load(timeZone);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    alert("Success");
}

function QueryFailure(sender, args) {
    alert("Failure");
}

```

```

}

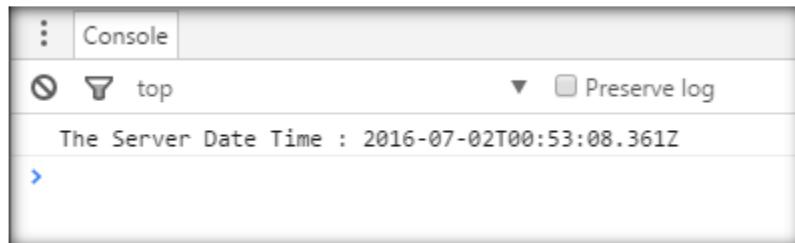
function QuerySuccess() {
    //Get the server date and time
    var timeZoneInformation = timeZone.get_information();
    var offsetTime = (timeZoneInformation.get_bias() +
timeZoneInformation.get_daylightBias()) / 60.0;
    var serverDateTime = new Date(new Date().getTime() - offsetTime * 3600 * 1000).toISOString();
    console.log("The Server Date Time : " + serverDateTime);
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

### Output:



## L. Get Supported Languages

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getSupportedLanguages);
});

var oWeb;
function getSupportedLanguages() {

```

```
//Get the client context and web object
var clientContext = new SP.ClientContext();
oWeb = clientContext.get_web();

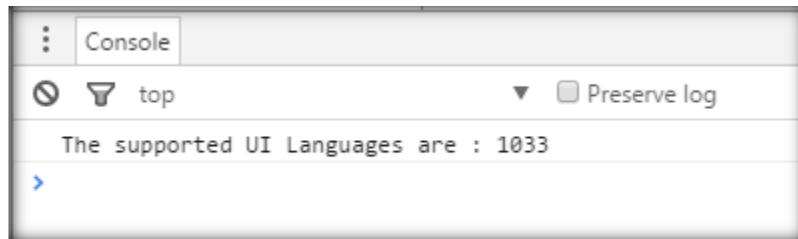
//Load the client context and execute the batch
clientContext.load(oWeb, 'SupportedUILanguageIds');
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    //Get the supported UI language IDs from the web object
    var supportedLanguages = oWeb.get_supportedUILanguageIds();
    console.log("The supported UI Languages are : "+supportedLanguages );
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>
```

### Output:



### **M. Get Web Templates**

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getWebTemplates);
});
```

```

var oWebTemplates,oListTemplates;
function getWebTemplates() {

    //Get the client context,web and web templates collection object
    var clientContext = new SP.ClientContext.get_current();
    oWeb = clientContext.get_web();
    oWebTemplates=oWeb.getAvailableWebTemplates(1033);

    //Load the client context and execute the batch
    clientContext.load(oWebTemplates);
    clientContext.executeQueryAsync(Success,Failure);
}

function Success() {

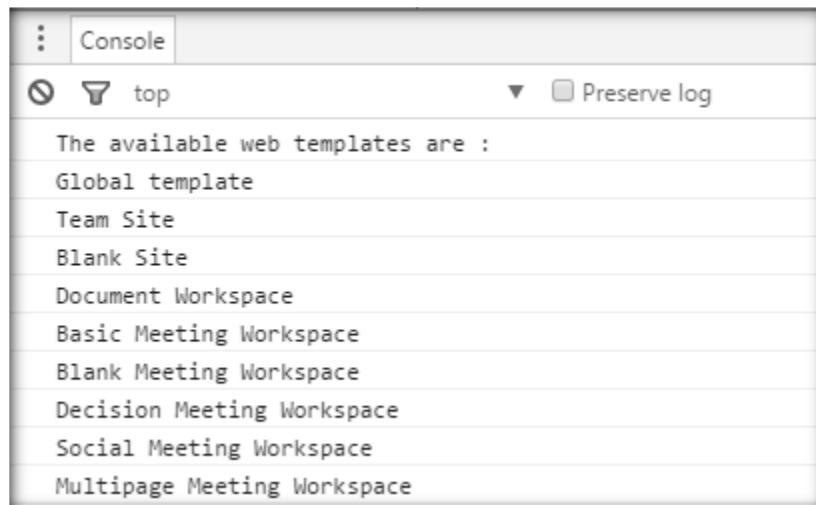
    //Get the web templates enumerator collection and loop through it
    var webTemplateEnumerator = oWebTemplates.getEnumerator();
    console.log("The available web templates are : ");
    while (webTemplateEnumerator.moveNext()) {
        console.log(webTemplateEnumerator.get_current().get_title());
    }
}

function Failure(sender, args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## Output:



```

:
Console
:
The available web templates are :
Global template
Team Site
Blank Site
Document Workspace
Basic Meeting Workspace
Blank Meeting Workspace
Decision Meeting Workspace
Social Meeting Workspace
Multipage Meeting Workspace

```

## XVII. Working with Users

### A. Check if User has Full Permissions

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', checkUserPermissions);
});

var oGroupColl,oWeb;
function checkUserPermissions() {

    //Get the client context, web and current user object
    var clientContext = new SP.ClientContext.get_current();
    oWeb = clientContext.get_web();
    oCurrentUser = oWeb.get_currentUser();

    //Load the client context with the required objects
    clientContext.load(oWeb, 'EffectiveBasePermissions')
    clientContext.load(oCurrentUser);

    //Execute the batch
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

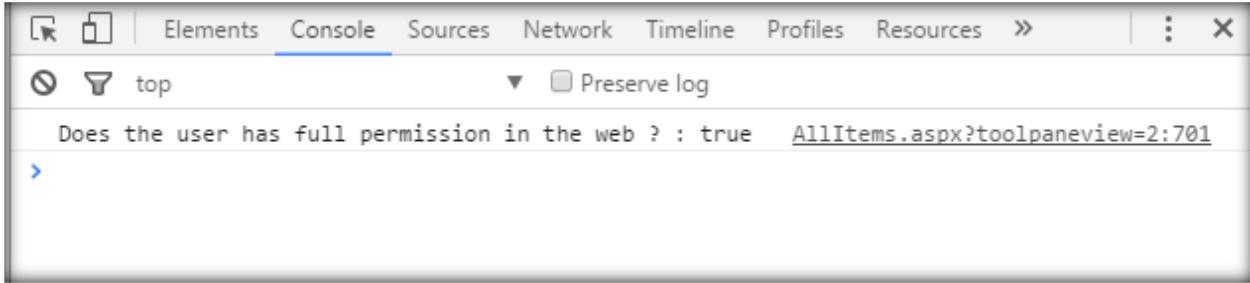
    //Check if user has Full Control Permission Level
    var userName = oCurrentUser.get_loginName();
    console.log("Does the user has full permission in the web ? : "+oWeb.get_effectiveBasePermissions().has(SP.PermissionKind.manageWeb));
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## **Output:**



The screenshot shows the Google Chrome Developer Tools with the 'Console' tab selected. The log output is as follows:

```
Does the user has full permission in the web ? : true AllItems.aspx?toolpaneview=2:701
```

## **B. Check if User is Present in Group**

### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### **Script:**

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', checkUserMembership);
});

var oGroupColl;
function checkUserMembership() {

    //Get the client context, web and current user object
    var clientContext = new SP.ClientContext.get_current();
    var oWeb = clientContext.get_web();
    oCurrentUser = oWeb.get_currentUser();

    //Get the group collection
    oGroupColl = oCurrentUser.get_groups();

    //Load the client context and execute the batch
    clientContext.load(oGroupColl);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
```

```

//Get the group collection and loop through it
var groupMembership = false;
var groupCollEnumerator = oGroupColl.getEnumerator();
while (groupCollEnumerator.moveNext()) {
    var group = groupCollEnumerator.get_current();

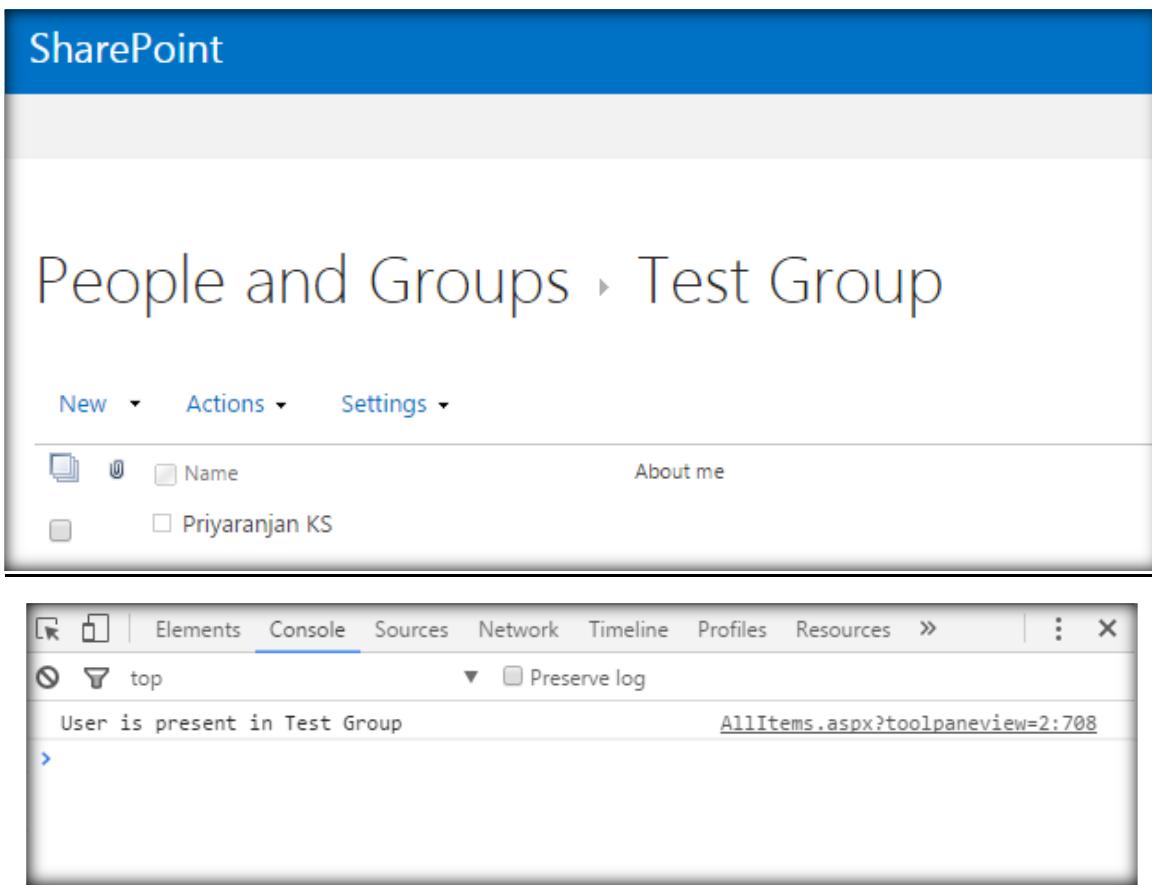
    //Check if a particular group is present in user's group collection
    if(group.get_title() == "Test Group") {
        isMember = true;
        console.log('User is present in Test Group');
        break;
    }
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

### Output:



The screenshot shows a SharePoint interface. At the top, there's a blue header bar with the text "SharePoint". Below it, the main content area has a title "People and Groups › Test Group". Underneath the title, there are buttons for "New", "Actions", and "Settings". The main content area displays a list of users in the "Test Group". One user, "Priyaranjan KS", is shown with a checkbox next to their name. To the right of the user list, there's a link "About me". At the bottom of the page, there's a "Preserve log" section of the browser's developer tools console. The log shows the following message: "User is present in Test Group" followed by the URL "AllItems.aspx?toolpaneview=2:708".

## C. Get All Users in the Web

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getAllUsers);
});

var oUsers;
function getAllUsers() {

    //Get the client context and web object
    var clientContext = new SP.ClientContext.get_current();
    var oWeb = clientContext.get_web();

    //Get the user collection information
    var userInfoList = oWeb.get_siteUserInfoList();
    var query = new SP.CamlQuery.createAllItemsQuery()

    //Use CAML Query to get all users
    oUsers = userInfoList.getItems(query);

    //Load the client context and execute the batch
    clientContext.load(oUsers, "Include(DisplayName)");
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

    //Loop through the user collection and fetch the display name
    var usersEnumerator = oUsers.getEnumerator();
    console.log("The users in the site are : ");
    while (usersEnumerator.moveNext()) {
        var oUser = usersEnumerator.get_current();
        console.log(oUser.get_displayName());
    }
}

function QueryFailure(sender,args) {
    alert('Request failed'+ args.get_message());
}
</script>

```

## Output:

Elements	Console
Sources	Network
Timeline	Profiles
Resources	»
	⋮
Preserve log	X
The users in the site are :	AllItems.aspx?toolpaneview=2:704
Excel Services Viewers	AllItems.aspx?toolpaneview=2:707
Everyone	AllItems.aspx?toolpaneview=2:707
Everyone except external users	AllItems.aspx?toolpaneview=2:707
Playground Owners	AllItems.aspx?toolpaneview=2:707
Playground Visitors	AllItems.aspx?toolpaneview=2:707
Playground Members	AllItems.aspx?toolpaneview=2:707
Priyaranjan KS	AllItems.aspx?toolpaneview=2:707
_spocrwl_337_16600	AllItems.aspx?toolpaneview=2:707
NT AUTHORITY\authenticated users	AllItems.aspx?toolpaneview=2:707
Company Administrator	AllItems.aspx?toolpaneview=2:707
Administrator Group	AllItems.aspx?toolpaneview=2:707
Style Resource Readers	AllItems.aspx?toolpaneview=2:707
Designers	AllItems.aspx?toolpaneview=2:707
Hierarchy Managers	AllItems.aspx?toolpaneview=2:707
Approvers	AllItems.aspx?toolpaneview=2:707
Restricted Readers	AllItems.aspx?toolpaneview=2:707
Quick Deploy Users	AllItems.aspx?toolpaneview=2:707
Translation Managers	AllItems.aspx?toolpaneview=2:707
Test Group	AllItems.aspx?toolpaneview=2:707
HR	AllItems.aspx?toolpaneview=2:707
Jinesh KS	AllItems.aspx?toolpaneview=2:707
HR Group	AllItems.aspx?toolpaneview=2:707
HR Owner	AllItems.aspx?toolpaneview=2:707
Priyan	AllItems.aspx?toolpaneview=2:707
SharePoint App	AllItems.aspx?toolpaneview=2:707
System Account	AllItems.aspx?toolpaneview=2:707

## D. Get Current User Properties

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getCurrentUserProperties);
});

var oCurrentUser;
function getCurrentUserProperties() {

    //Get the client context, web and current user object
    var clientContext = new SP.ClientContext.get_current();
    var oWeb = clientContext.get_web();
    oCurrentUser = oWeb.get_currentUser();

    //Load the client context and execute the batch
    clientContext.load(oCurrentUser);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

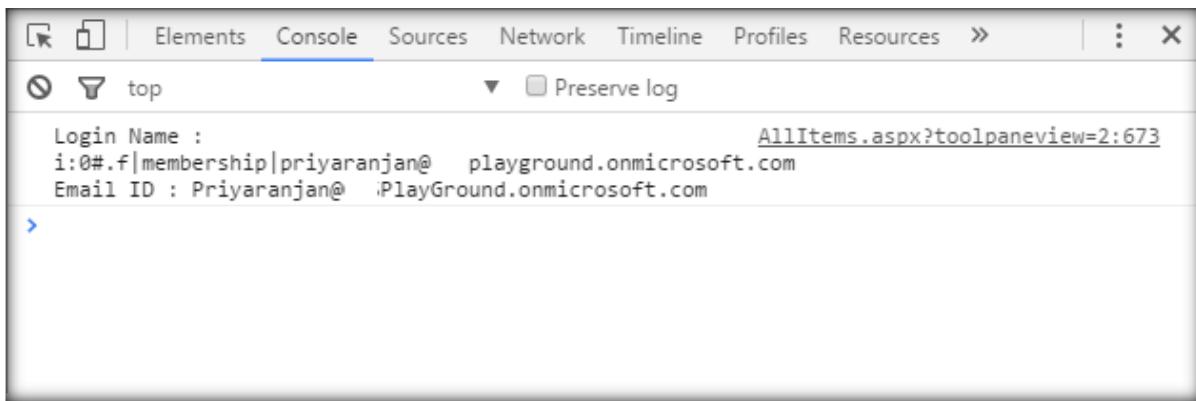
    console.log("Login Name : " + oCurrentUser.get_loginName() +"\nEmail ID : "
    "+oCurrentUser.get_email());
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

### Output:



The screenshot shows the Chrome Developer Tools Console tab. The output in the console is:

```

Login Name : i:0#.f|membership|priyaranjan@ playground.onmicrosoft.com
Email ID : Priyaranjan@ PlayGround.onmicrosoft.com

```

## E. Get User's Groups

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getGroups);
});

var oGroupColl;
function getGroups() {

    //Get the client context, web and current user object
    var clientContext = new SP.ClientContext.get_current();
    var oWeb = clientContext.get_web();
    oCurrentUser = oWeb.get_currentUser();

    //Get the group collection of the user
    oGroupColl = oCurrentUser.get_groups();

    //Load the client context and execute the batch
    clientContext.load(oGroupColl);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

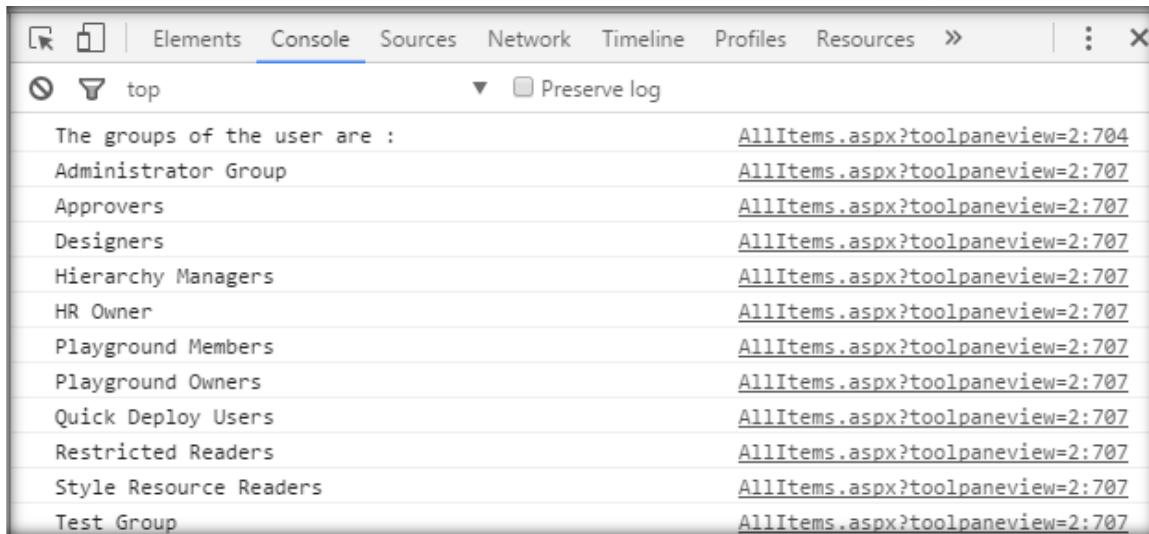
    //Loop through the user's group collection
    var groupMembership = false;
    var groupCollEnumerator = oGroupColl.getEnumerator();
    console.log("The groups of the user are : ");
    while (groupCollEnumerator.moveNext()) {
        var group = groupCollEnumerator.get_current();
        console.log(group.get_title());
    }
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## Output:



Groups	URL
The groups of the user are :	AllItems.aspx?toolpaneview=2:704
Administrator Group	AllItems.aspx?toolpaneview=2:707
Approvers	AllItems.aspx?toolpaneview=2:707
Designers	AllItems.aspx?toolpaneview=2:707
Hierarchy Managers	AllItems.aspx?toolpaneview=2:707
HR Owner	AllItems.aspx?toolpaneview=2:707
Playground Members	AllItems.aspx?toolpaneview=2:707
Playground Owners	AllItems.aspx?toolpaneview=2:707
Quick Deploy Users	AllItems.aspx?toolpaneview=2:707
Restricted Readers	AllItems.aspx?toolpaneview=2:707
Style Resource Readers	AllItems.aspx?toolpaneview=2:707
Test Group	AllItems.aspx?toolpaneview=2:707

## F. Get User Information

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', getUserInfo);
});

var items;
function getUserInfo() {

    //Get the client context and the User Info List object that contains info about
    all site users
    var clientContext = new SP.ClientContext.get_current();
    var userInfoList = clientContext.get_web().get_siteUserInfoList();
    var query = new SP.CamlQuery();
    var viewXml = "<View> \
                  <Query> \
                  <Where> \
                  <Eq><FieldRef Name='UserName' /><Value

```

```
Type='Text'>priyanjan@ctsplayground.onmicrosoft.com</Value></Eq> \
    </Where> \
    </Query> \
    <RowLimit>1</RowLimit> \
    </View>";
query.set_viewXml(viewXml);

//Get the specific user from SiteUserInfoList using CAML
items = userInfoList.getItems(query);

//Load the client context and execute the batch
clientContext.load(items, 'Include(Department,EMail,FirstName,IsSiteAdmin,JobTitle,LastName,Name,UserName)');
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

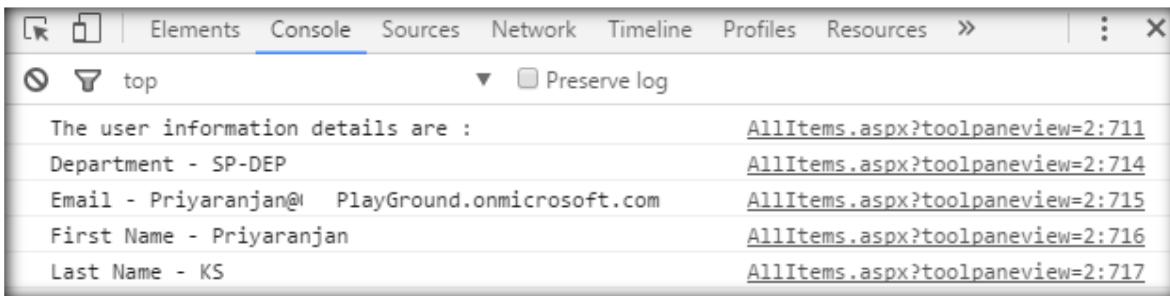
function QuerySuccess() {

    //Loop through the collection and ge the user information values
    console.log("The user information details are : ");
    if(items.get_count() > 0) {
        var item = items.itemAt(0);
        console.log("Department - "+ item.get_fieldValues().Department);
        console.log("Email - "+ item.get_fieldValues().EMail);
        console.log("First Name - "+ item.get_fieldValues().FirstName);
        console.log("Last Name - "+ item.get_fieldValues().LastName);
    }
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>
```

## Output:



Output	Timestamp
The user information details are :	AllItems.aspx?toolpaneview=2:711
Department - SP-DEP	AllItems.aspx?toolpaneview=2:714
Email - Priyaranjan@ PlayGround.onmicrosoft.com	AllItems.aspx?toolpaneview=2:715
First Name - Priyaranjan	AllItems.aspx?toolpaneview=2:716
Last Name - KS	AllItems.aspx?toolpaneview=2:717

## XVIII. Working with Search

### A. Display Managed Property in Results

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
        $.getScript(scriptbase + "SP.js", function() {
            $.getScript(scriptbase + "SP.search.js", getSearchResults);
        });
    });
});

var results;
function getSearchResults() {

    //Get the client context and the web object
    var clientContext = new SP.ClientContext();
    oWeb = clientContext.get_web();

    //Fetch records with managed property value Demo is equal to Done
    var keywordQuery = new
Microsoft.SharePoint.Client.Search.Query.KeywordQuery(clientContext);
    keywordQuery.set_queryText("Demo:Done");

    //Get the selected properties of the returned results
    var displayProperties = keywordQuery.get_selectProperties();
    displayProperties.add('Demo');
    displayProperties.add('EditorOWSUSER');

    //Initiate the search executor to Execute the search query to return results
    var searchExecutor = new

```

```

Microsoft.SharePoint.Client.Search.Query.SearchExecutor(clientContext);
results = searchExecutor.executeQuery(keywordQuery);

//Execute the batch
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

//Iterate the returned results and print the properties
$.each(results.m_value.ResultTables[0].ResultRows, function () {

    console.log("-----");
    console.log("Title -> "+this.Title +"\n");
    console.log("Path-> "+this.Path +"\n");
    console.log("Author -> "+this.Author +"\n");
    console.log("Demo -> "+this.Demo +"\n");
    console.log("Modified By (Principal) -> "+this.EditorOWSUSER +"\n");
});

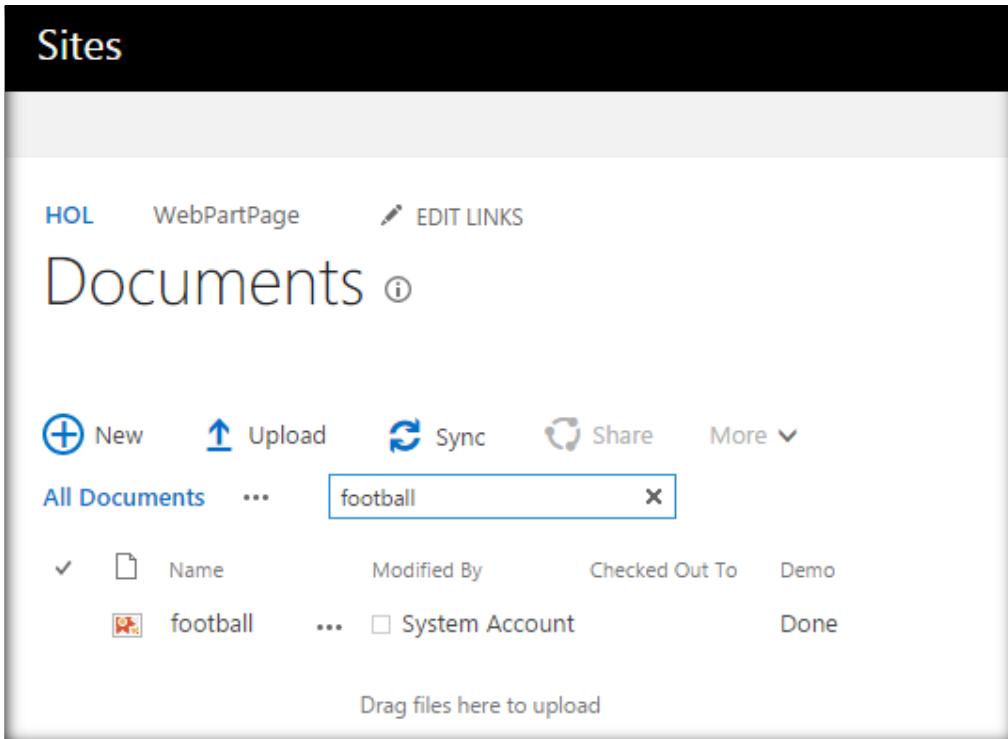
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

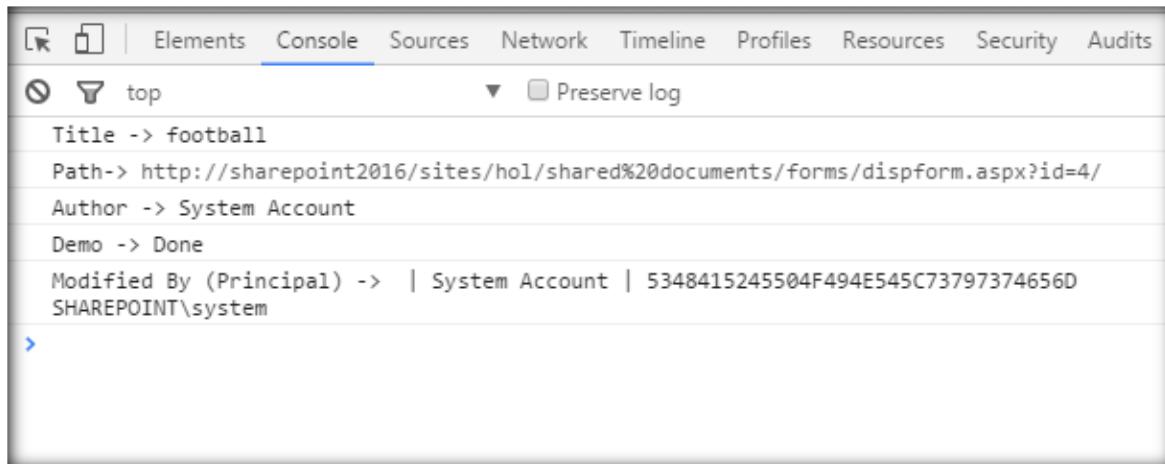
```

## Output:



The screenshot shows a SharePoint 'Documents' library page. At the top, there's a navigation bar with 'HOL' and 'WebPartPage' buttons, and an 'EDIT LINKS' button. Below the navigation is a search bar containing the text 'football'. Underneath the search bar, there's a table with columns: 'Name', 'Modified By', 'Checked Out To', and 'Demo'. A single item is listed: 'football' by 'System Account' (checked out to 'Demo'). At the bottom of the list, there's a placeholder text 'Drag files here to upload'.

Name	Modified By	Checked Out To	Demo
football	System Account	Done	



## B. Filter by Managed Property

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {

        $.getScript(scriptbase + "SP.js", function () {

            $.getScript(scriptbase + "SP.search.js", getSearchResults);

        });
    });
});

var results;
function getSearchResults() {

    //Get the client context and the web object
    var clientContext = new SP.ClientContext();
    oWeb = clientContext.get_web();
```

```

//Fetch records with managed property value Demo is equal to Done
var keywordQuery = new
Microsoft.SharePoint.Client.Search.Query.KeywordQuery(clientContext);
keywordQuery.set_queryText("Demo:Done");

//Initiate the search executor to Execute the search query to return results
var searchExecutor = new
Microsoft.SharePoint.Client.Search.Query.SearchExecutor(clientContext);
results = searchExecutor.executeQuery(keywordQuery);

//Execute the batch
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

//Iterate the returned results and print the properties
$.each(results.m_value.ResultTables[0].ResultRows, function () {
    console.log("-----");
    console.log("Title -> "+this.Title +"\n");
    console.log("Path-> "+this.Path +"\n");
    console.log("Author -> "+this.Author +"\n");
    console.log("Updated On -> "+this.Write +"\n");
});

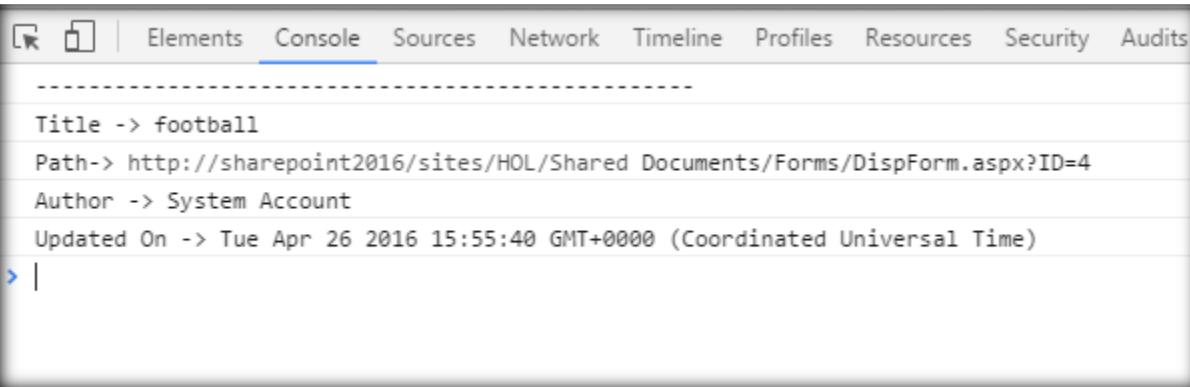
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## **Output:**



```

Title -> football
Path-> http://sharepoint2016/sites/HOL/Shared Documents/Forms/DispForm.aspx?ID=4
Author -> System Account
Updated On -> Tue Apr 26 2016 15:55:40 GMT+0000 (Coordinated Universal Time)
> |

```

## C. Filter by Result Source

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {

        $.getScript(scriptbase + "SP.js", function() {

            $.getScript(scriptbase + "SP.search.js", getSearchResults);

        });
    });
});

var results;
function getSearchResults() {

    //Get the client context and the web object
    var clientContext = new SP.ClientContext();
    oWeb = clientContext.get_web();

    //Fetch records from result source Documents matching the keyword 'SharePoint'
    var keywordQuery = new
Microsoft.SharePoint.Client.Search.Query.KeywordQuery(clientContext);
    keywordQuery.set_queryText("SharePoint");
    keywordQuery.get_properties().set_item('SourceName', 'Documents');
    keywordQuery.get_properties().set_item('SourceLevel', 'SSA');

    //Initiate the search executor to Execute the search query to return results
    var searchExecutor = new
Microsoft.SharePoint.Client.Search.Query.SearchExecutor(clientContext);
    results = searchExecutor.executeQuery(keywordQuery);

    //Execute the batch
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

```

```
//Iterate the returned results and print the properties
$.each(results.m_value.ResultTables[0].ResultRows, function () {
    console.log("-----");
    console.log("Title -> "+this.Title +"\n");
    console.log("Path-> "+this.Path +"\n");
    console.log("Author -> "+this.Author +"\n");
    console.log("Updated On -> "+this.Write +"\n");
});

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>
```

## **Output:**

```
-----
Title -> Add user to Group Using Nintex
Path-> http://sharepoint2016/sites/HOL/Shared Documents/Add user to Group Using Nintex.docx
Author -> 456India;Priyaranjan
Updated On -> Sat Jul 23 2016 06:40:00 GMT+0000 (Coordinated Universal Time)
-----
Title -> File_Plan_Report_Record_Library_2016-05-13T030655
Path-> http://sharepoint2016/sites/Records Management/Records/File_Plan_Report_Record_Library_2016-05-13T030655.xlsx
Author -> System Account
Updated On -> Fri May 13 2016 03:06:59 GMT+0000 (Coordinated Universal Time)
-----
>
```

## **D. Add Multiple Filters**

### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

## Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {

        $.getScript(scriptbase + "SP.js", function() {

            $.getScript(scriptbase + "SP.search.js", getSearchResults);

        });
    });
});

var results;
function getSearchResults() {

    //Get the client context and the web object
    var clientContext = new SP.ClientContext();
    oWeb = clientContext.get_web();

    //Fetch records that matches the filter criteria for file type and file name
    var keywordQuery = new
Microsoft.SharePoint.Client.Search.Query.KeywordQuery(clientContext);
    keywordQuery.set_queryText("football filetype:txt AND filename:WorldCup.txt");

    //Initiate the search executor to Execute the search query to return results
    var searchExecutor = new
Microsoft.SharePoint.Client.Search.Query.SearchExecutor(clientContext);
    results = searchExecutor.executeQuery(keywordQuery);

    //Execute the batch
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

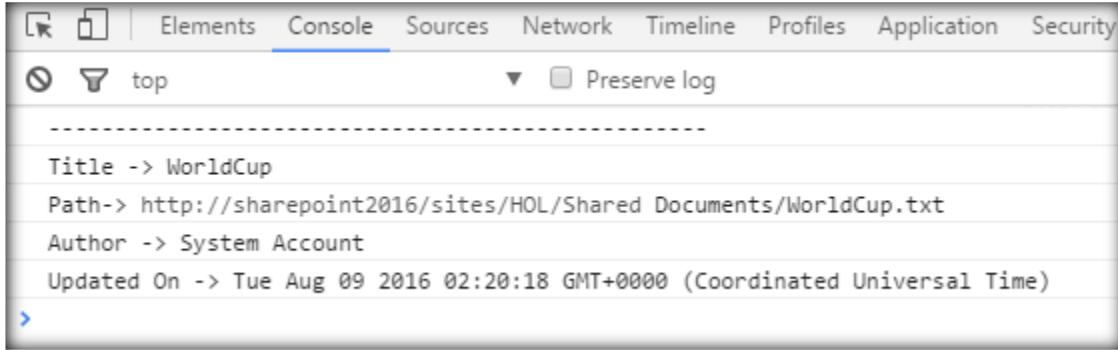
function QuerySuccess() {

    //Iterate the returned results and print the properties
    $.each(results.m_value.ResultTables[0].ResultRows, function () {
        console.log("-----")
        console.log("Title -> "+this.Title +"\n");
        console.log("Path-> "+this.Path +"\n");
        console.log("Author -> "+this.Author +"\n");
        console.log("Updated On -> "+this.Write +"\n");
    });
}

function QueryFailure(sender,args) {
    alert('Request failed'+ args.get_message());
}
</script>

```

## **Output:**



```
-----  
Title -> WorldCup  
Path-> http://sharepoint2016/sites/HOL/Shared Documents/WorldCup.txt  
Author -> System Account  
Updated On -> Tue Aug 09 2016 02:20:18 GMT+0000 (Coordinated Universal Time)
```

## **E. Refinement Filters by Filetype**

### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### **Script:**

```
<script language="javascript" type="text/javascript"  
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>  
  
<script language="javascript" type="text/javascript">  
  
$(document).ready(function() {  
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";  
    $.getScript(scriptbase + "SP.Runtime.js", function () {  
  
        $.getScript(scriptbase + "SP.js", function () {  
  
            $.getScript(scriptbase + "SP.search.js", getSearchResults);  
  
        });  
    });  
  
});  
  
var results;  
function getSearchResults() {  
  
    //Get the client context and the web object  
    var clientContext = new SP.ClientContext();  
    oWeb = clientContext.get_web();
```

```

//Fetch records that matches the filter criteria for file type
var keywordQuery = new
Microsoft.SharePoint.Client.Search.Query.KeywordQuery(clientContext);
keywordQuery.set_queryText("football filetype:png");

//Initiate the search executor to Execute the search query to return results
var searchExecutor = new
Microsoft.SharePoint.Client.Search.Query.SearchExecutor(clientContext);
results = searchExecutor.executeQuery(keywordQuery);

//Execute the batch
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

    //Iterate the returned results and print the properties
$.each(results.m_value.ResultTables[0].ResultRows, function () {
    console.log("-----");
    console.log("Title -> "+this.Title +"\n");
    console.log("Path-> "+this.Path +"\n");
    console.log("Author -> "+this.Author +"\n");
    console.log("Updated On -> "+this.Write +"\n");
});

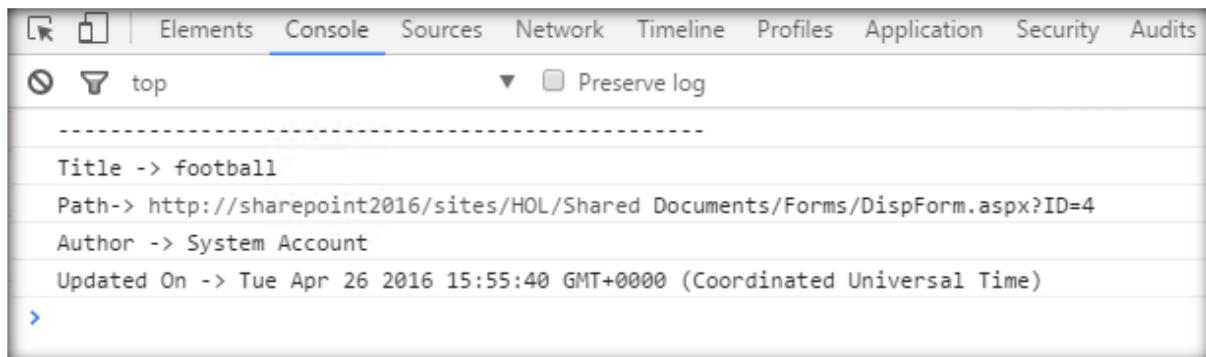
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## **Output:**



```

-----  

Title -> football  

Path-> http://sharepoint2016/sites/HOL/Shared Documents/Forms/DispForm.aspx?ID=4  

Author -> System Account  

Updated On -> Tue Apr 26 2016 15:55:40 GMT+0000 (Coordinated Universal Time)
>

```

## F. Refinement Filters by Date Time

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
        $.getScript(scriptbase + "SP.js", function() {
            $.getScript(scriptbase + "SP.search.js", getSearchResults);

        });
    });
});

var results;
function getSearchResults() {

    //Get the client context and the web object
    var clientContext = new SP.ClientContext();
    oWeb = clientContext.get_web();

    //Fetch records that matches the filter criteria for created date
    var keywordQuery = new
Microsoft.SharePoint.Client.Search.Query.KeywordQuery(clientContext);
    keywordQuery.set_queryText('football write>04/25/2016 ');

    //Initiate the search executor to Execute the search query to return results
    var searchExecutor = new
Microsoft.SharePoint.Client.Search.Query.SearchExecutor(clientContext);
    results = searchExecutor.executeQuery(keywordQuery);

    //Execute the batch
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

    //Iterate the returned results and print the properties
    $.each(results.m_value.ResultTables[0].ResultRows, function () {
        console.log("-----")
    })
}
```

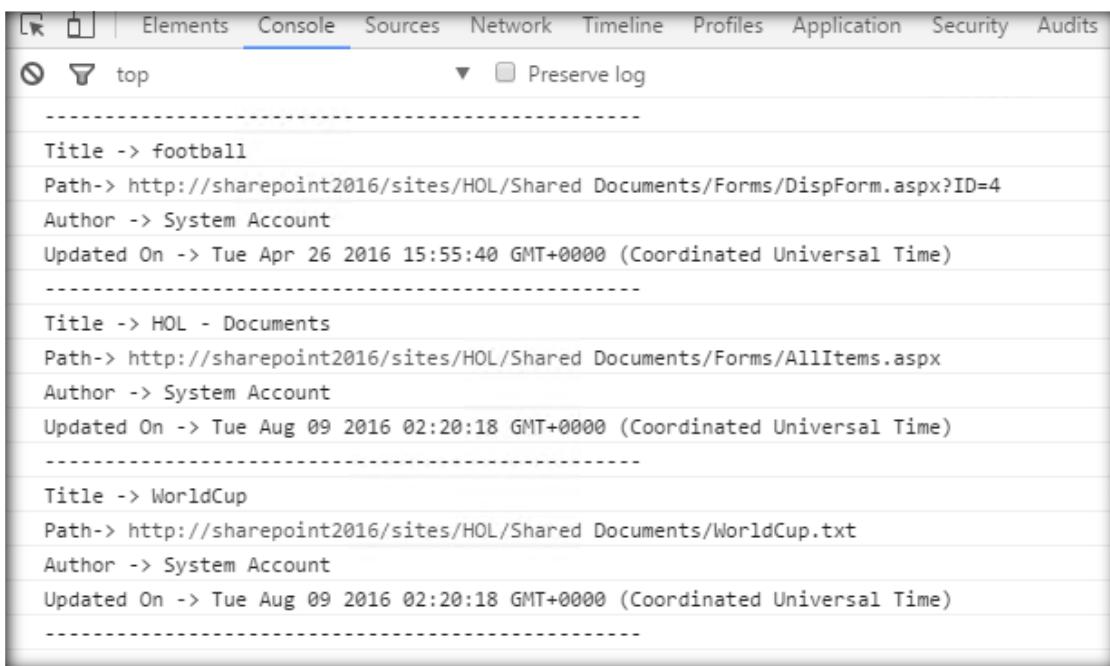
```
        console.log("Title -> "+this.Title +"\n");
        console.log("Path-> "+this.Path +"\n");
        console.log("Author -> "+this.Author +"\n");
        console.log("Updated On -> "+this.Write +"\n");
    });

}

function QueryFailure(sender,args) {
    alert('Request failed'+ args.get_message());
}

</script>
```

### Output:



The screenshot shows the Chrome DevTools Console tab with three log entries. Each entry contains the item's title, path, author, and update timestamp.

```
-----  
Title -> football  
Path-> http://sharepoint2016/sites/HOL/Shared Documents/Forms/DispForm.aspx?ID=4  
Author -> System Account  
Updated On -> Tue Apr 26 2016 15:55:40 GMT+0000 (Coordinated Universal Time)  
-----  
Title -> HOL - Documents  
Path-> http://sharepoint2016/sites/HOL/Shared Documents/Forms/AllItems.aspx  
Author -> System Account  
Updated On -> Tue Aug 09 2016 02:20:18 GMT+0000 (Coordinated Universal Time)  
-----  
Title -> WorldCup  
Path-> http://sharepoint2016/sites/HOL/Shared Documents/WorldCup.txt  
Author -> System Account  
Updated On -> Tue Aug 09 2016 02:20:18 GMT+0000 (Coordinated Universal Time)
```

## XIX. Working with Social Features

### A. Follow Document

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {

        $.getScript(scriptbase + "SP.js", function() {

            $.getScript(scriptbase + "SP.UserProfiles.js", followDocument);

        });
    });

});
var followingManager,documentActorInfo;
function followDocument() {

    //Get the client context and social following manager object
    var clientContext = new SP.ClientContext();
    followingManager = new SP.Social.SocialFollowingManager(clientContext);

    // Create a SocialActorInfo object that would associate the document to be
followed.
    newDocumentActorInfo = new SP.Social.SocialActorInfo();

newDocumentActorInfo.set_contentUri("http://sharepoint2016/sites/HOL/SiteAssets/Search
/DisplayManagedPropertyInResults.txt");
    newDocumentActorInfo.set_actorType(SP.Social.SocialActorType.document);
    followingManager.follow(newDocumentActorInfo);

    //Execute the batch
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    console.log("The document has been followed");
}

```

```

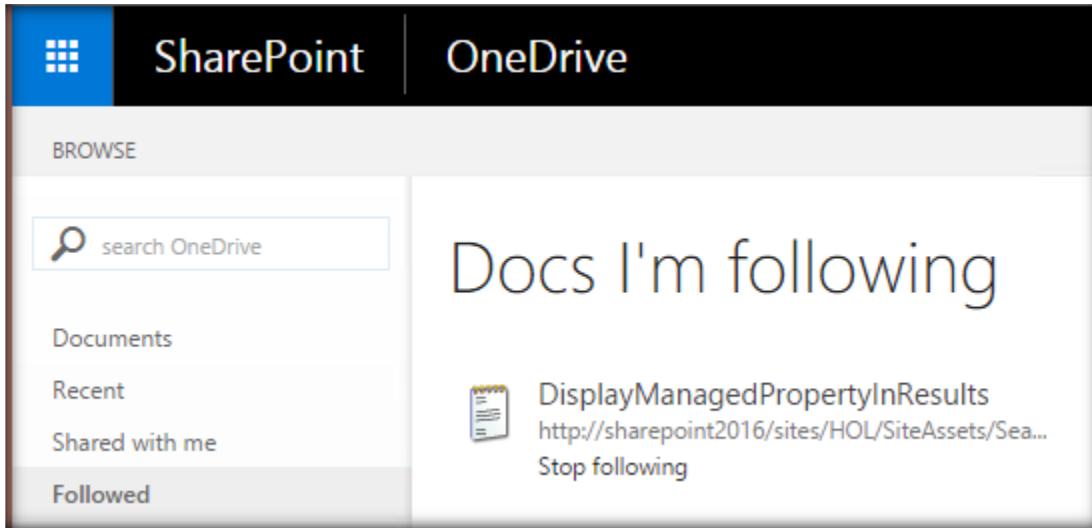
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

### Output:



### B. Followed Users

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {

        $.getScript(scriptbase + "SP.js", function() {

            $.getScript(scriptbase + "SP.UserProfiles.js", followedUsers);

```

©2016 C# CORNER.

SHARE THIS DOCUMENT AS IT IS. PLEASE DO NOT REPRODUCE, REPUBLISH, CHANGE OR COPY.

```

        });
    });

});

var followingManager,actorInfo;
function followedUsers() {

    //Get client context and the social following manager object
    var clientContext = new SP.ClientContext();
    followingManager = new SP.Social.SocialFollowingManager(clientContext);

    //Get the followed users
    followed = followingManager.getFollowed(1);

    //Execute the batch
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

    //Loop through the users and display each user details.
    for (var i = 0; i < followed.length; i++) {
        var user = followed[i];
        var name = user.get_name();
        console.log("Followed User "+i+" - "+name);
    }

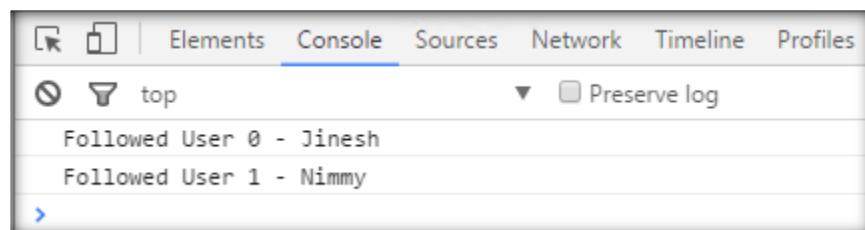
}

function QueryFailure(sender,args) {
    alert('Request failed'+ args.get_message());
}

</script>

```

## Output:



## C. Follow Site

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {

        $.getScript(scriptbase + "SP.js", function() {

            $.getScript(scriptbase + "SP.UserProfiles.js", followSite);

        });
    });

});

var followingManager,actorInfo;
function followSite() {

    //Get client context and social following manager object
    var clientContext = new SP.ClientContext();
    followingManager = new SP.Social.SocialFollowingManager(clientContext);

    // Create a SocialActorInfo object to associate the following site.
    siteActorInfo = new SP.Social.SocialActorInfo();
    siteActorInfo.set_contentUri("http://sharepoint2016/sites/HOL");
    siteActorInfo.set_actorType(SP.Social.SocialActorType.site);
    followingManager.follow(siteActorInfo);

    //Execute the batch
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

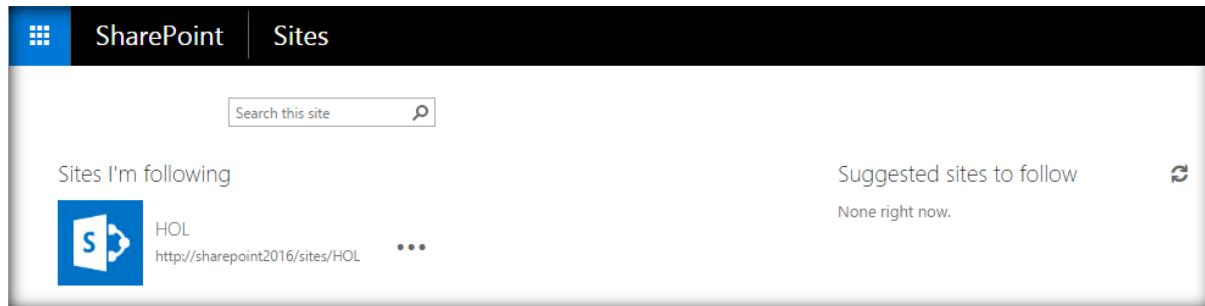
function QuerySuccess() {
    console.log("The site has been followed.");
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## Output:



## D. Follow User

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {

        $.getScript(scriptbase + "SP.js", function() {

            $.getScript(scriptbase + "SP.UserProfiles.js", followUser);

        });
    });
});

var userToFollow = 'SharePointHOL\\Jyothi';
var followingManager,actorInfo;
function followUser() {

    var clientContext = new SP.ClientContext();
    followingManager = new SP.Social.SocialFollowingManager(clientContext);

    // Create a SocialActorInfo object to associate the user to be followed.
    actorInfo = new SP.Social.SocialActorInfo();
    actorInfo.set_accountName(userToFollow);
}
```

```

//Follow the user
followingManager.follow(actorInfo);

//Execute the batch
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

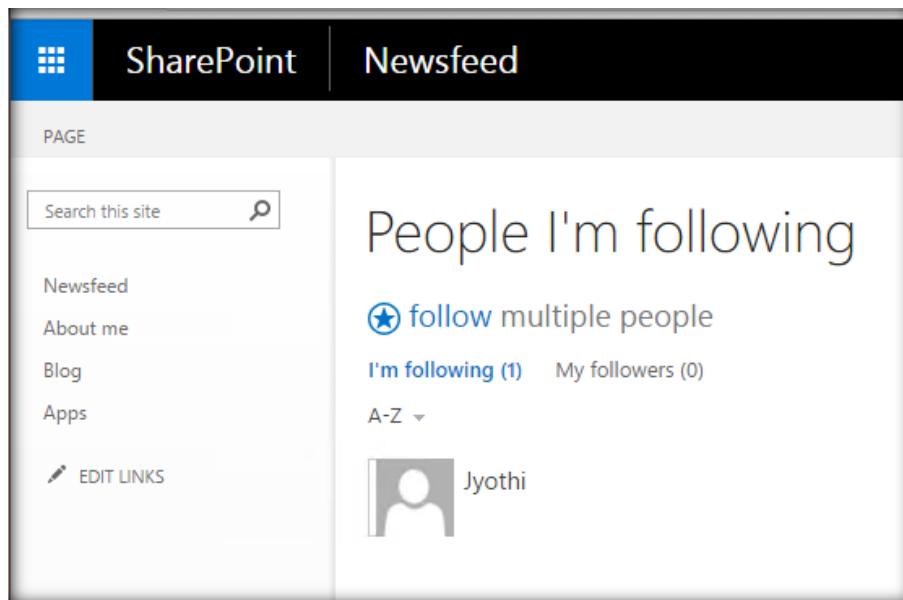
function QuerySuccess() {
    console.log("The user has been followed.");
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## Output:



## E. Get Followed Documents

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {

        $.getScript(scriptbase + "SP.js", function() {

            $.getScript(scriptbase + "SP.UserProfiles.js", followedDocuments);

        });
    });
});

var documentsFollowed ;
function followedDocuments() {

    //Get the client context and the social following manager object
    var clientContext = new SP.ClientContext();
    followingManager = new SP.Social.SocialFollowingManager(clientContext);

    //Get the followed Documents
    documentsFollowed = followingManager.getFollowed(2);

    //Execute the batch
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

    //Display the followed document details
    console.log("The Count of documents followed : " + documentsFollowed.length);
    console.log("The Followed Documents are : ");
    for (var i = 0; i < documentsFollowed.length; i++) {

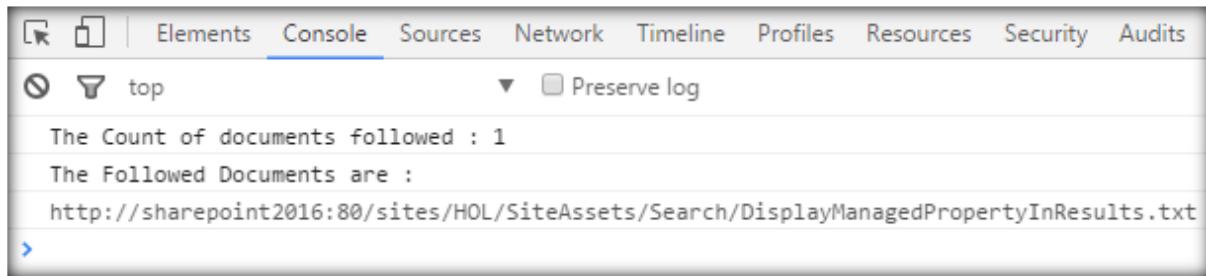
        console.log(documentsFollowed[i].get_contentUri());
    }
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

### Output:



The Count of documents followed : 1  
The Followed Documents are :  
<http://sharepoint2016:80/sites/HOL/SiteAssets/Search/DisplayManagedPropertyInResults.txt>

## F. Get Followed Sites

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {

        $.getScript(scriptbase + "SP.js", function() {

            $.getScript(scriptbase + "SP.UserProfiles.js", followedSites);

        });
    });

});

var sitesFollowed;
function followedSites() {

    //Get the client context and social following manager object
    var clientContext = new SP.ClientContext();
    followingManager = new SP.Social.SocialFollowingManager(clientContext);

    //Get followed Sites
    sitesFollowed = followingManager.getFollowed(4);

    //Execute the batch
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}
```

```

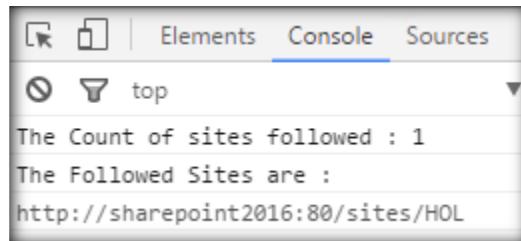
function QuerySuccess() {
    //Display the details of the followed sites
    console.log("The Count of sites followed : " + sitesFollowed.length);
    console.log("The Followed Sites are : ");
    for (var i = 0; i < sitesFollowed.length; i++) {
        console.log(sitesFollowed[i].get_contentUri());
    }
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

### Output:



### G. Get Followed Users

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
        $.getScript(scriptbase + "SP.js", function () {
            $.getScript(scriptbase + "SP.UserProfiles.js", followedUsers);
        });
    });
});

```

```

        });
    });

});

var followedCount;
function followedUsers() {

    //Get client context and social following manager object
    var clientContext = new SP.ClientContext();
    followingManager = new SP.Social.SocialFollowingManager(clientContext);

    //Get the followed users count
    followedCount = followingManager.getFollowedCount(1);

    //Execute the batch
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

    //Display the count of users
    console.log('Total count of followed people: ' + followedCount.getValue());

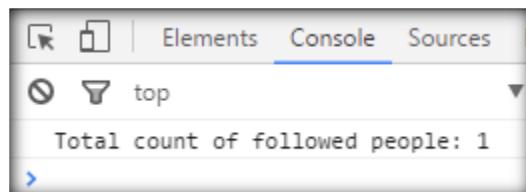
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.getMessage());
}

</script>

```

## Output:



## H. Get Following Status

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

## Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {

        $.getScript(scriptbase + "SP.js", function() {

            $.getScript(scriptbase + "SP.UserProfiles.js", followingStatus);

        });
    });
});

var targetUser = 'SharePointHOL\\Jyothi';
var isFollowed ;
function followingStatus() {

    //Get client context and social following manager object
    var clientContext = new SP.ClientContext();
    followingManager = new SP.Social.SocialFollowingManager(clientContext);

    //Get the social actor info for the target user
    actorInfo = new SP.Social.SocialActorInfo();
    actorInfo.set_accountName(targetUser);

    //Get the following status
    isFollowed = followingManager.isFollowed(actorInfo);

    //Execute the batch
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

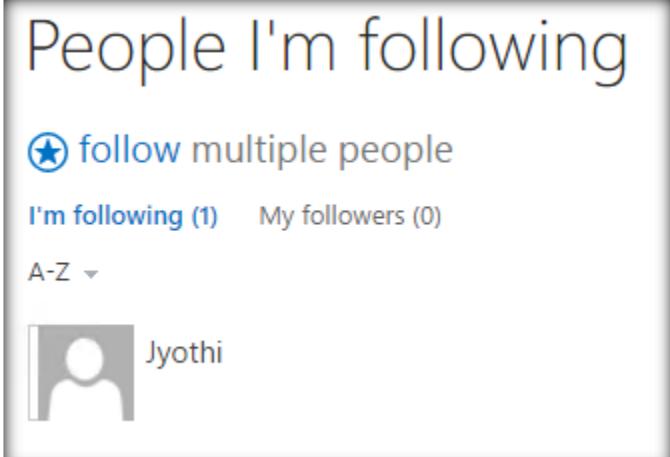
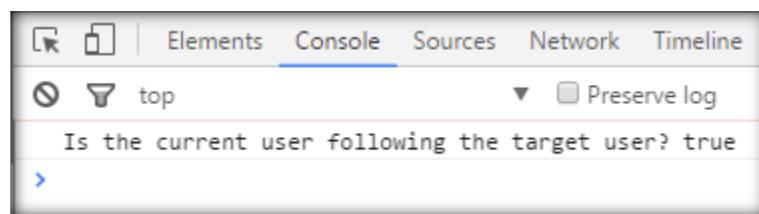
function QuerySuccess() {
    console.log('Is the current user following the target user? ' +
isFollowed.get_value());
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## Output:

## I. Stop Following Document

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
```

```
$.getScript(scriptbase + "SP.js", function() {
    $.getScript(scriptbase + "SP.UserProfiles.js", stopFollowingDocument);
});

});

});

var followingManager,documentActorInfo;
function stopFollowingDocument() {
    //Get the client context and the social following manager object
    var clientContext = new SP.ClientContext();
    followingManager = new SP.Social.SocialFollowingManager(clientContext);

    // Create a SocialActorInfo object to associate the document.
    documentActorInfo = new SP.Social.SocialActorInfo();

    documentActorInfo.set_contentUri("http://sharepoint2016/sites/HOL/SiteAssets/Social/DisplayManagedPropertyInResults.txt");
    documentActorInfo.set_actorType(SP.Social.SocialActorType.document);

    //Stop following the document
    followingManager.stopFollowing(documentActorInfo);

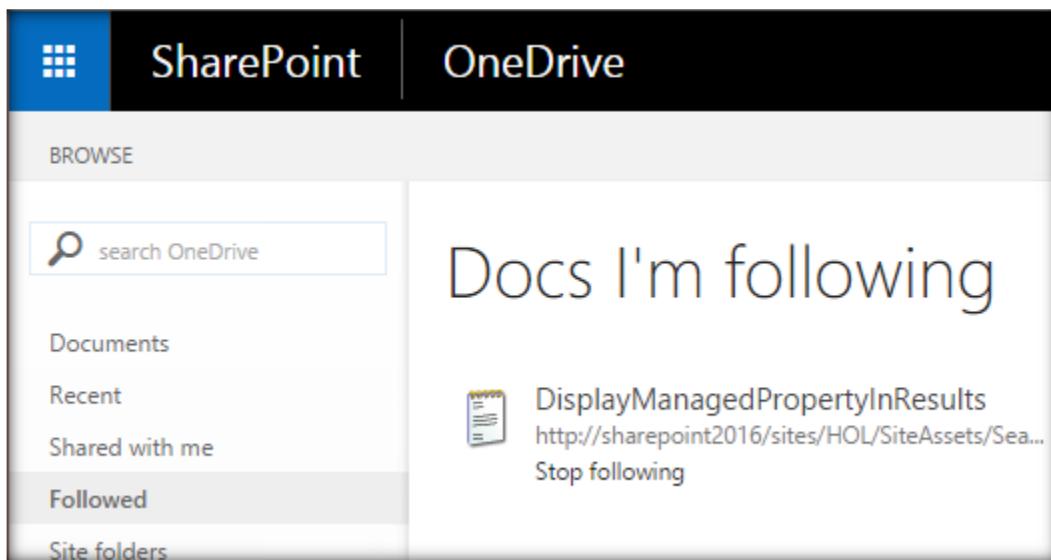
    //Execute the batch
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    console.log("The document is no longer followed.");
}

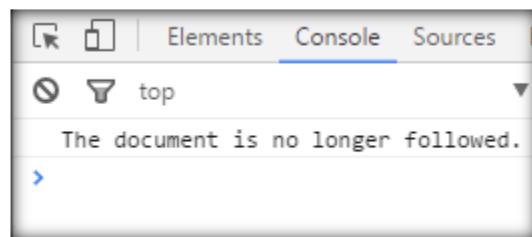
function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>
```

## Output:



The screenshot shows the SharePoint OneDrive interface. On the left, there's a sidebar with a search bar labeled 'search OneDrive'. Below it are links: 'Documents', 'Recent', 'Shared with me', **'Followed'** (which is highlighted), and 'Site folders'. On the right, under the heading 'Docs I'm following', there is a card for a document titled 'DisplayManagedPropertyInResults' with the URL 'http://sharepoint2016/sites/HOL/SiteAssets/Sea...'. Below the title, there's a link 'Stop following'.



## J. Stop Following Site

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
```

```

$.getScript(scriptbase + "SP.js", function() {
    $.getScript(scriptbase + "SP.UserProfiles.js", stopFollowingSite);

});

});

});

var followingManager,actorInfo;
function stopFollowingSite() {
    //Get the client context and the social following manager
    var clientContext = new SP.ClientContext();
    followingManager = new SP.Social.SocialFollowingManager(clientContext);

    // Create a SocialActorInfo object to associate the site
    siteActorInfo = new SP.Social.SocialActorInfo();
    siteActorInfo.set_contentUri("http://sharepoint2016/sites/HOL");
    siteActorInfo.set_actorType(SP.Social.SocialActorType.site);

    //Stop following the site
    followingManager.stopFollowing(siteActorInfo);

    //Execute the batch
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

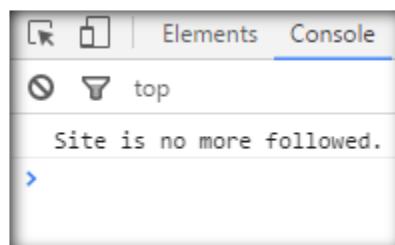
function QuerySuccess() {
    console.log("Site is no more followed.");
}

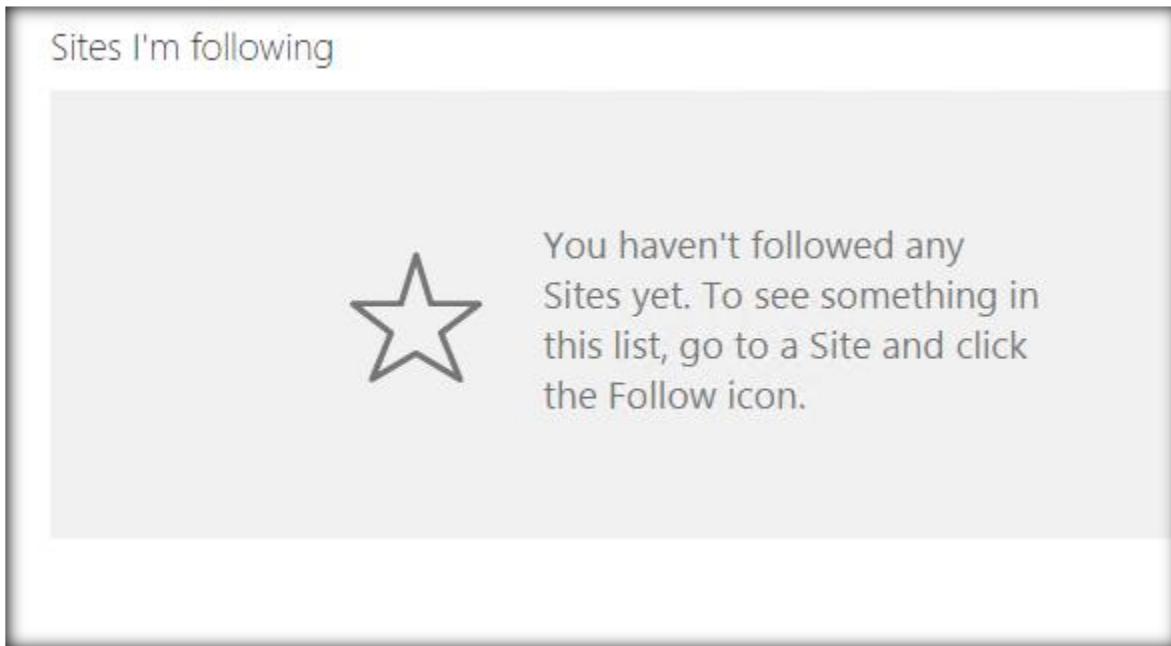
function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## Output:





## K. Stop Following User

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {

        $.getScript(scriptbase + "SP.js", function() {

            $.getScript(scriptbase + "SP.UserProfiles.js", stopfollowingUser);

        });
    });
});
```

```

});;

var targetUser = 'SharePointHOL\\Jyothi';
var followingManager,actorInfo;
function stopfollowingUser() {

    //Get the client context and social following manager object
    var clientContext = new SP.ClientContext();
    followingManager = new SP.Social.SocialFollowingManager(clientContext);

    // Create a SocialActorInfo object to represent the target user.
    actorInfo = new SP.Social.SocialActorInfo();
    actorInfo.set_accountName(targetUser);

    //Stop following the user
    followingManager.stopFollowing(actorInfo);

    //Execute the batch
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

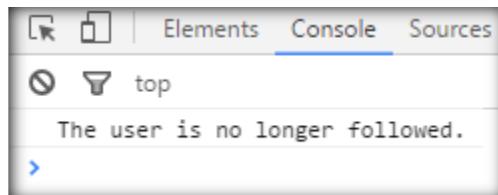
function QuerySuccess() {
    console.log("The user is no longer followed.");
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## Output:



A screenshot of a SharePoint page titled "People I'm following". The page features a "follow multiple people" button with a star icon. Below it are links for "I'm following (0)" and "My followers (0)". A dropdown menu shows "A-Z". At the bottom, a message states "You're not following anyone yet."

## XX. Working with User Profile

### A. Get User Profile

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
        $.getScript(scriptbase + "SP.js", function () {
            $.getScript(scriptbase + "SP.UserProfiles.js", getUserProfile);
        });
    });
});

var userProperties ;
function getUserProfile() {
    // Get the current client context and PeopleManager instance.
    var clientContext = new SP.ClientContext.get_current();
    var peopleManager = new SP.UserProfiles.PeopleManager(clientContext);

    //Get the user profile properties of the current user
    userProperties = peopleManager.getMyProperties();

    //Load the client context and execute the batch
    clientContext.load(userProperties);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    //Display the user's user profile properties
    var messageText = "DisplayName - " + userProperties.get_displayName();
    messageText += "\nDepartment - " +
    userProperties.get_userProfileProperties()['Department'];
}

```

```

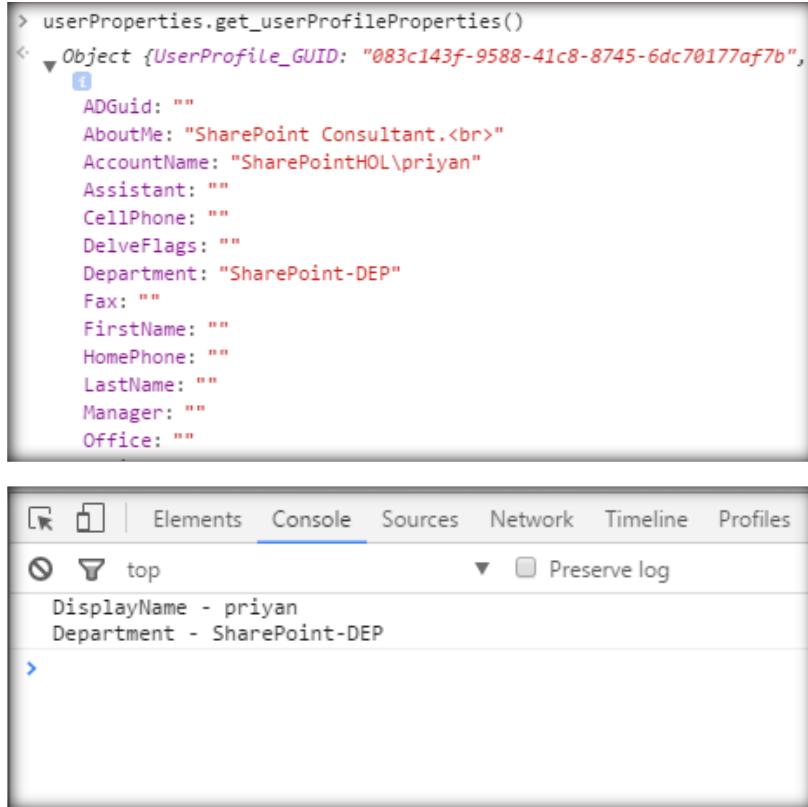
        console.log(messageText);
    }

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

### Output:



The screenshot shows the browser's developer tools with the 'Console' tab selected. At the top, there is a code snippet: > userProperties.get\_userProfileProperties(). Below it, the output of the script is shown as an object with various properties. The properties listed are: ADGuid, AboutMe, AccountName, Assistant, CellPhone, DelveFlags, Department, Fax, FirstName, HomePhone, LastName, Manager, and Office. Each property is followed by a value enclosed in double quotes.

```

> userProperties.get_userProfileProperties()
< Object {UserProfile_GUID: "083c143f-9588-41c8-8745-6dc70177af7b",
  ADGuid: ""
  AboutMe: "SharePoint Consultant.<br>"
  AccountName: "SharePointHOL\priyan"
  Assistant: ""
  CellPhone: ""
  DelveFlags: ""
  Department: "SharePoint-DEP"
  Fax: ""
  FirstName: ""
  HomePhone: ""
  LastName: ""
  Manager: ""
  Office: ""
}

```

Below the object output, the browser's console log shows two entries: 'DisplayName - priyan' and 'Department - SharePoint-DEP'. There is also a blue arrow icon pointing to the right at the bottom of the log area.

## B. Get Single User Profile Properties

### Steps:

- Add Content Editor web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
        $.getScript(scriptbase + "SP.js", function () {
            $.getScript(scriptbase + "SP.UserProfiles.js", getUserProfile);
        });
    });
});

var userProperties ;
function getUserProfile() {
    // Get the current client context and PeopleManager instance.
    var clientContext = new SP.ClientContext.get_current();
    var peopleManager = new SP.UserProfiles.PeopleManager(clientContext);

    //Set the property to retrieve
    var propertyName = "FirstName";

    //If you are on SharePoint Online use the below format :
    //var targetUser = "i:0#.f|membership|Jinesh@CTSPlayGround.onmicrosoft.com";

    //If you are working on SharePoint 2016:
    var targetUser = "SharePointHOL\\Priyaranjan" ;

    //Create new instance of UserProfileProperty
    userProperty = peopleManager.getUserProfilePropertyFor(targetUser, propertyName)

    //Execute the batch
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

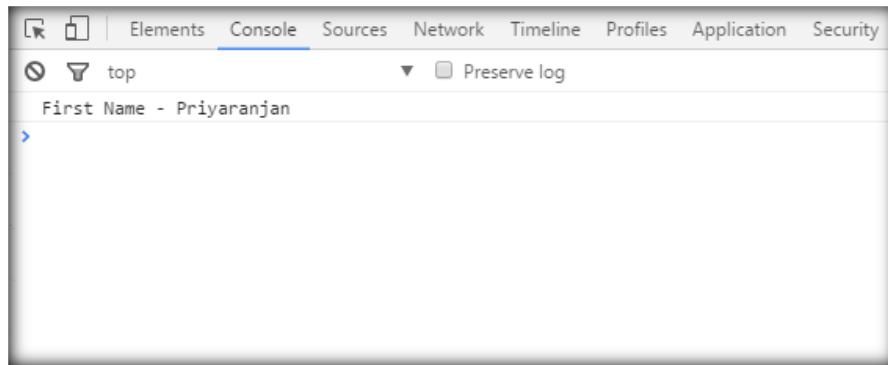
function QuerySuccess() {
    //Display the user profile property for the selected user
    var messageText = "First Name - " + userProperty.get_value();
    console.log(messageText);
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## **Output:**



## **C. Get Multiple User Profile Properties of a User**

### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### **Script:**

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
        $.getScript(scriptbase + "SP.js", function () {
            $.getScript(scriptbase + "SP.UserProfiles.js", getUserProfile);
        });
    });
});

var userProperties ;
function getUserProfile() {
    //Specify the target user
}
```

```

var specificUser = "SharePointHOL\\Priyaranjan";

// Get the current client context and PeopleManager instance.
var clientContext = new SP.ClientContext.get_current();
var peopleManager = new SP.UserProfiles.PeopleManager(clientContext);

//Fetch the user profile properties
userProperties = peopleManager.getPropertiesFor(specificUser);

//Load the client context and execute the batch
clientContext.load(userProperties);
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

    //Display the user profile properties
    var messageText = "DisplayName - " + userProperties.get_displayName();

    messageText += "\nDepartment - " + userProperties.get_userProfileProperties()['Department'];

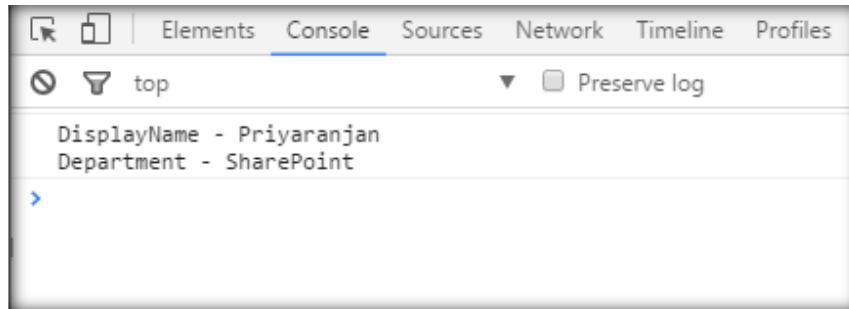
    console.log(messageText);
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## Output:



## D. Get User Profile Properties for Multiple Users

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

## Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
        $.getScript(scriptbase + "SP.js", function () {
            $.getScript(scriptbase + "SP.UserProfiles.js", getUserProfile);
        });
    });

var clientContext,peopleManager ;
var userProperties = [];
var specificUsers =[];
function getUserProfile() {
    specificUsers =[ "SharePointHOL\\Priyan", "SharePointHOL\\Priyaranjan"];
    // Get the current client context and PeopleManager instance.
    clientContext = new SP.ClientContext.get_current();
    peopleManager = new SP.UserProfiles.PeopleManager(clientContext);

    var profilePropertyNames = [ "AccountName", "Department"];

    for(var i=0;i<specificUsers.length;i++){
        var userProfilePropertiesForUser = new
SP.UserProfiles.UserProfilePropertiesForUser(clientContext,specificUsers[i],profilePro
pertyNames);

        //Create new instance of UserProfileProperty
        userProfileProperties[i] =
peopleManager.getUserProfilePropertiesForUser(userProfilePropertiesForUser);
    }
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

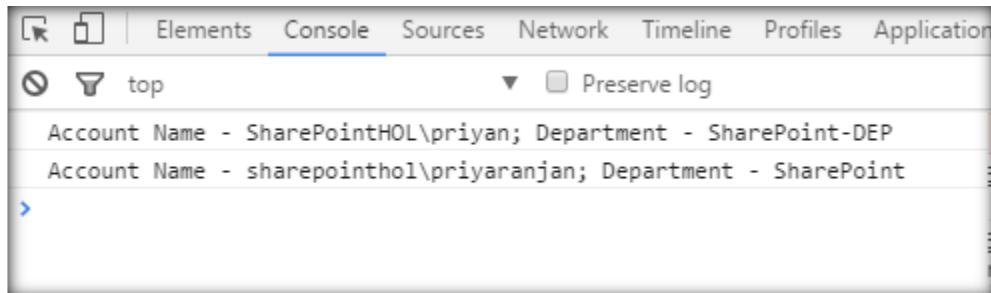
function QuerySuccess() {

    for(var i=0;i<userProperties.length;i++){
        var currentUserProperties = userProperties[i];
        console.log("Account Name - " + currentUserProperties[0]+"; Department - "+
currentUserProperties[1])
    }
}

function QueryFailure(sender,args) {
    alert('Request failed'+ args.get_message());
}
</script>

```

## Output:



```
Account Name - SharePointHOL\priyan; Department - SharePoint-DEP
Account Name - sharepointhol\priyaranjan; Department - SharePoint
```

## E. Set Multi Value User Profile Property

Setting of user profile property via JSOM is supported only in Office 365 and not in SharePoint On-premise. Hopefully, the API will be updated soon for On-premise as well.

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
        $.getScript(scriptbase + "SP.js", function () {
            $.getScript(scriptbase + "SP.UserProfiles.js",
SetCurrentUserProperties);
        });
    });
});

var userProfileProperties,peopleManager,clientContext;
function SetCurrentUserProperties() {
```

```

//Get Current Context
clientContext = SP.ClientContext.get_current();

//Get Instance of People Manager Class
peopleManager = new SP.UserProfiles.PeopleManager(clientContext);

//Get properties of the current user
userProfileProperties = peopleManager.getMyProperties();

//Get only the accountname instead of all the properties.
clientContext.load(userProfileProperties, "AccountName");

//Execute the Query.
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);

}

function QuerySuccess() {

    //Get the account name of the current user. It will be in the following format:
    "i:0#.f|membership|Priyaranjan@CTSPPlayground.onmicrosoft.com"
    var currentUserAccountName = userProfileProperties.get_accountName();

    var multipleValues = ["Design Patterns", "SharePoint 2016", "Office 365"];
    peopleManager.setMultiValuedProfileProperty(currentUserAccountName, "SPS-Skills",
multipleValues);

    //Execute the Query.
    clientContext.executeQueryAsync(SecondQuerySuccess, SecondQueryFailure);
}

function QueryFailure(sender,args) {
    alert('Request failed'+ args.get_message());
}

function SecondQuerySuccess() {
    console.log("Success");
}

function SecondQueryFailure(sender,args) {
    alert('Request failed'+ args.get_message());
}

</script>

```

## **Output:**

Office Location:

Enter your current location.  
(e.g. China, Tokyo, West Campus)

Assistant:

Past projects:

Provide information on previous projects, teams or groups.

Skills:

Design Patterns; SharePoint 2016; Office 365;

Include skills used to perform your job or previous projects.  
(e.g. C++, Public Speaking, Design)

Schools:

List the schools you have attended.

Birthday:

Enter the date in the following format: August 9

## F. Set Current User Properties

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
        $.getScript(scriptbase + "SP.js", function () {
            $.getScript(scriptbase + "SP.UserProfiles.js",
SetCurrentUserProperties);
        });
    });
});
```

```

        );
    });

var clientContext,userProfileProperties,peopleManager;

function SetCurrentUserProperties() {

    //Get Current Context
    clientContext = SP.ClientContext.get_current();

    //Get Instance of People Manager Class
    peopleManager = new SP.UserProfiles.PeopleManager(clientContext);

    //Get properties of the current user
    userProfileProperties = peopleManager.getMyProperties();

    //Get only the accountname instead of all the properties.
    clientContext.load(userProfileProperties, "AccountName");

    //Execute the Query.
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);

}

function QuerySuccess(){

    var currentUserAccountName = userProfileProperties.get_accountName();

    //Set a single value property
    peopleManager.setSingleValueProfileProperty(currentUserAccountName, "AboutMe",
"SharePoint Consultant");

    clientContext.executeQueryAsync(SecondQuerySuccess, SecondQueryFailure);
}

function QueryFailure(sender,args) {
    console.log ('Request failed' + args.get_message());
}

function SecondQuerySuccess() {
    console.log("Success");
}

function SecondQueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

**Output:**

Admin

enter

Use this page to edit this user profile by changing values for the following properties. Properties that are mapped to imported.

 Account name:	i:0#.f membership priyaranjan@ctsplayground.onmicrosoft.com
 First name:	Priyaranjan
 Last name:	KS
 Name: *	Priyaranjan KS
 Work phone: *	
 Department: *	SP-DEP
 Title: *	
Department:	
 Manager:	 
About me:	<input type="text" value="SharePoint Consultant"/> 

## XXI. Working with Modal Dialog

### A. Call Out menu

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```

<head>
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {

    SP.SOD.executeFunc("callout.js", "Callout", createSharePointCallOutPopup);
});

function createSharePointCallOutPopup()
{

    //Set the element that has to be shown as call out pop up
    var destinationDiv = document.getElementById('LaunchCallout');

    //Set the call out menu options
    var calloutPopUpOptions = new CalloutOptions();
    calloutPopUpOptions.ID = 'SharePointCallout';           //Set the id
    calloutPopUpOptions.launchPoint = destinationDiv;      //Set the destination element
    calloutPopUpOptions.beakOrientation = 'topBottom';
    calloutPopUpOptions.content = '<p>&ampnbsp</p><p>C# Corner: A social community for developers and
programmers.</p><p><span style="text-decoration: underline;"><strong>C# Corner
Contacts</strong></span></p><p><strong>Praveen Kumar(<a href="mailto:praveen@c-
sharpcorner.com">praveen@c-sharpcorner.com</a>)</strong></p><p><strong>Dinesh Kumar
(<a href="mailto:dinesh@c-sharpcorner.com">dinesh@c-
sharpcorner.com</a>)</strong></p>';

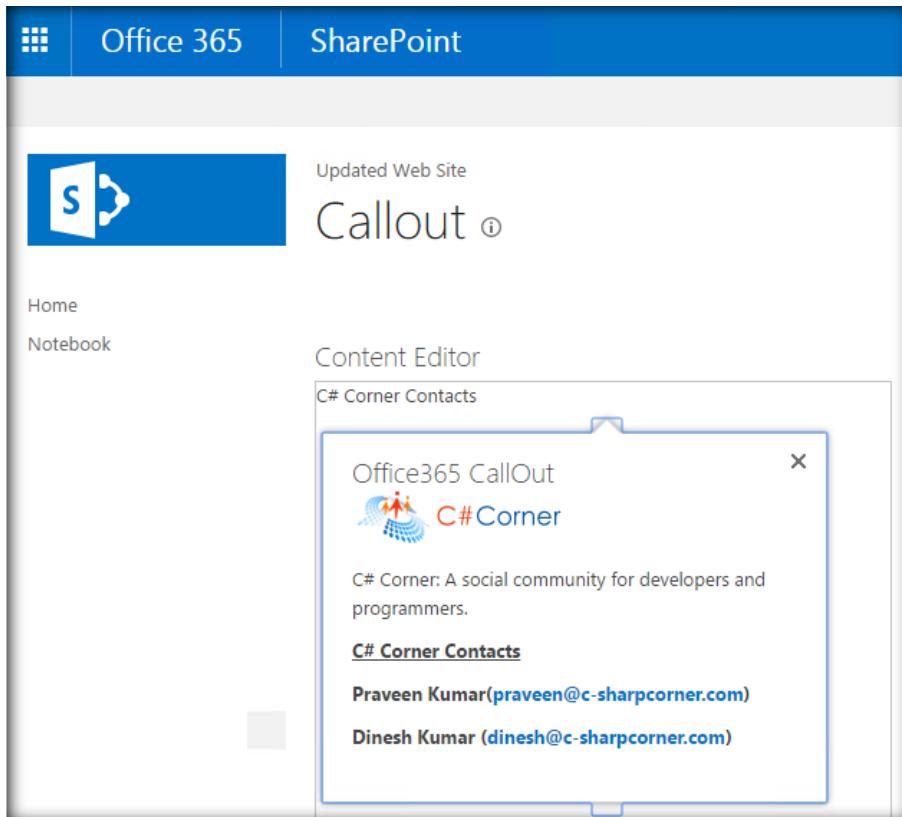
    calloutPopUpOptions.title = 'SharePoint 2016 CallOut';

    //Instantiate the call out pop up
    var displayedCallOut= CalloutManager.createNew(calloutPopUpOptions);
}

</script></head>
<body>
<div id="LaunchCallout">C# Corner Contacts</div>
</body>

```

## Output:



## B. Modal Dialog

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

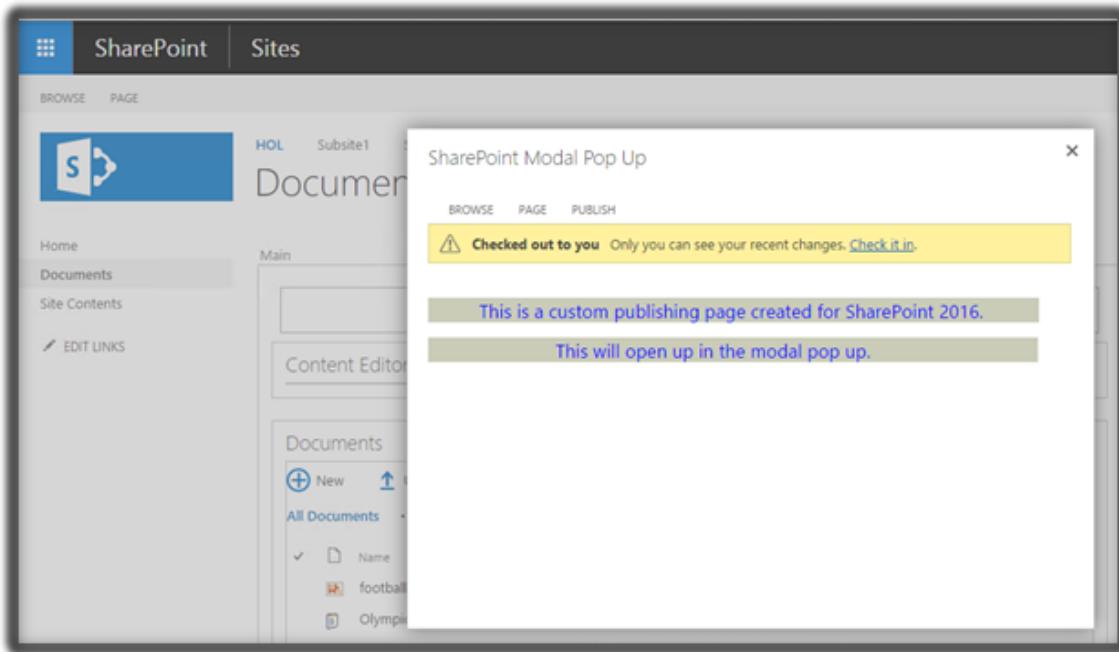
```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
<script language="javascript" type="text/javascript">
$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', showModalPopUp);
});
```

```

function showModalPopUp() {
    //Set options for Modal PopUp
    var options = {
        url: '/sites/HOL/Pages/Custom-Publishing-Page.aspx?IsDlg=1', //Set the url of
        the page
        title: 'SharePoint Modal Pop Up', //Set the title for the pop up
        allowMaximize: false,
        showClose: true,
        width: 600,
        height: 400
    };
    //Invoke the modal dialog by passing in the options array variable
    SP.SOD.execute('sp.ui.dialog.js', 'SP.UI.ModalDialog.showModalDialog', options);
    return false;
}
</script>

```

## Output:



## XXII. Working with Taxonomy

### A. Create TermGroup

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
        $.getScript(scriptbase + "SP.js", function () {
            $.getScript(scriptbase + "SP.Taxonomy.js", getTerms);
        });
    });
});

var terms;
function getTerms() {
    //Create the client context and the taxonomy session object
    var clientContext = new SP.ClientContext();
    taxonomiesession = SP.Taxonomy.TaxonomySession.getTaxonomySession(clientContext);

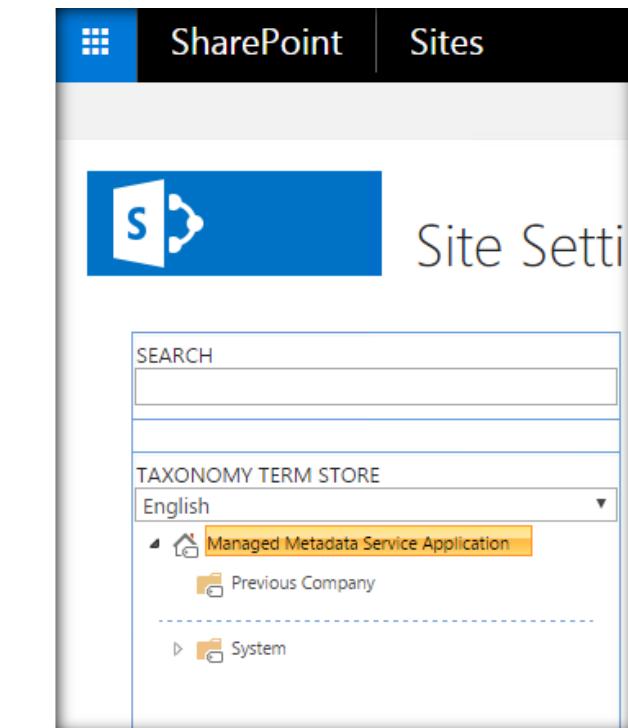
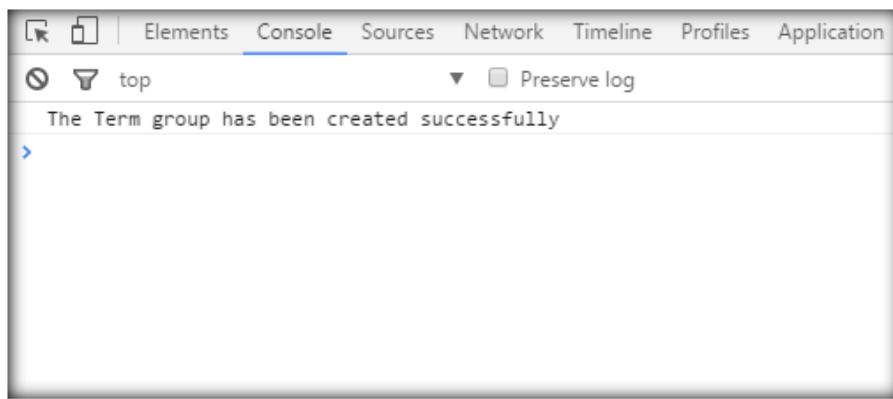
    //Get the term store collection and term store object
    termStoreCollection = taxonomiesession.get_termStores();
    var termStore = termStoreCollection.getByName("Managed Metadata Service
Application");

    //Create new term store group
    var newGroup = termStore.createGroup("Previous Company", "b300304a-1693-4629-a1c0-
dff7bda644ff");

    //Load the client context and execute the batch
    clientContext.load(newGroup);
}
```

```
        clientContext.executeQueryAsync(QuerySuccess, QueryFailure);  
    }  
  
    function QuerySuccess() {  
        console.log("The Term group has been created successfully");  
    }  
  
    function QueryFailure(sender,args) {  
        console.log('Request failed - '+ args.get_message());  
    }  
  
</script>
```

### Output:



## B. Create Termset

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
        $.getScript(scriptbase + "SP.js", function () {
            $.getScript(scriptbase + "SP.Taxonomy.js", getTerms);
        });
    });
});

var terms;
function getTerms() {
    //Create the client context and the taxonomy session object
    var clientContext = new SP.ClientContext();
    taxonomysession = SP.Taxonomy.TaxonomySession.getTaxonomySession(clientContext);

    //Get the term store collection, term store and group object
    termStoreCollection = taxonomysession.get_termStores();
    var termStore = termStoreCollection.getByName("Managed Metadata Service
Application");
    var group= termStore.get_groups().getByName("Previous Company");

    //Create new Term set
    var newTermSet = group.createTermSet("Contoso Ltd", "5dda5eb3-4f4f-464a-be40-
40aca0ed4ca9",1033);

    clientContext.load(newTermSet);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
}

```

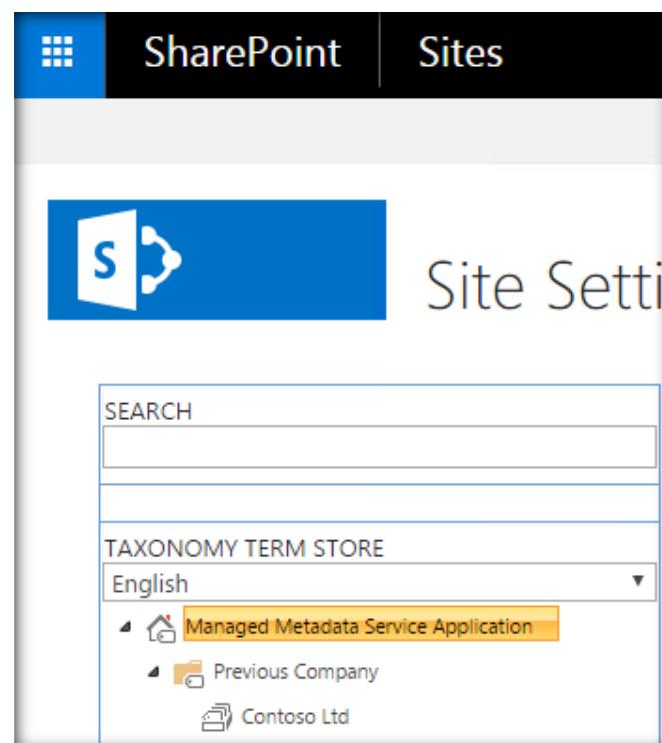
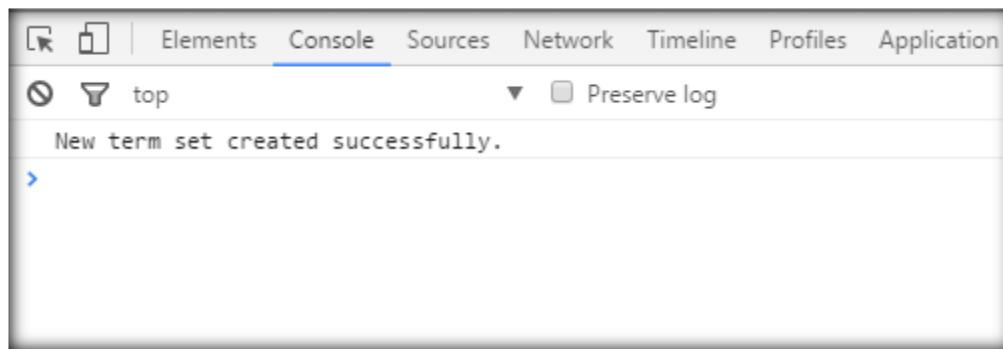
```
        console.log("New term set created successfully.");

    }

    function QueryFailure(sender,args) {
        console.log('Request failed - '+ args.get_message());
    }

</script>
```

## Output:



## C. Create Term

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
        $.getScript(scriptbase + "SP.js", function () {
            $.getScript(scriptbase + "SP.Taxonomy.js", getTerms);
        });
    });
});

var terms;
function getTerms() {
    //Create the client context and the taxonomy session object
    var clientContext = new SP.ClientContext();
    taxonomysession = SP.Taxonomy.TaxonomySession.getTaxonomySession(clientContext);

    //Get the term store collection and the term store object
    termStoreCollection = taxonomysession.get_termStores();
    var termStore = termStoreCollection.getByName("Managed Metadata Service
Application");

    //Get the termstore group and term set objects
    var group= termStore.get_groups().getByName("Previous Company");
    var termSet = group.get_termSets().getByName("Contoso Ltd")

    //Create new term
    var newTerm = termSet.createTerm("HR", 1033, "b49f64b3-4722-4336-9a5c-
56c326b344a9");

    //Load the client context and execute the batch
    clientContext.load(newTerm);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

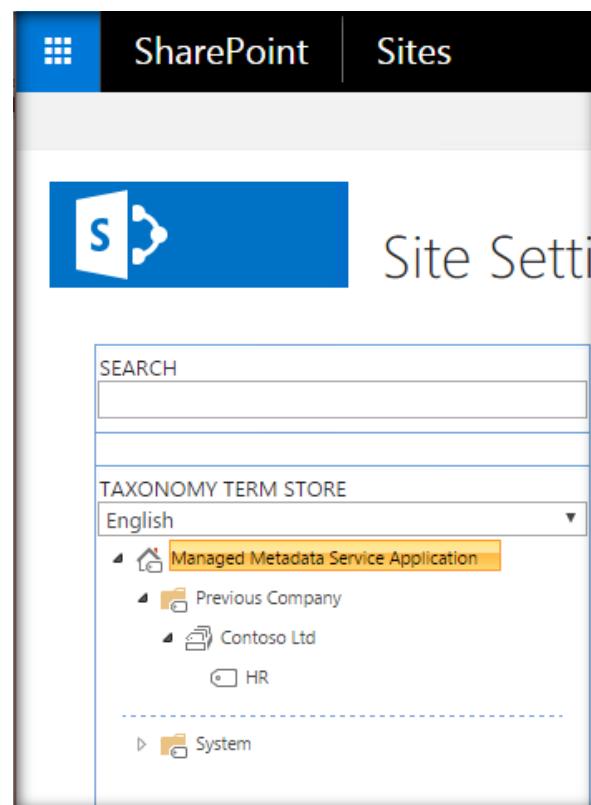
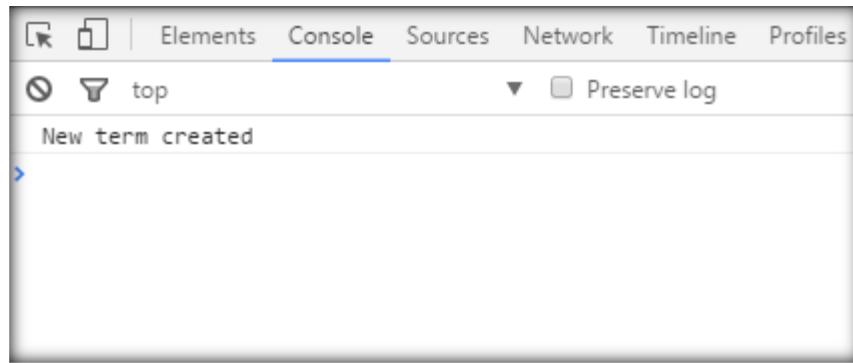
```

```
function QuerySuccess() {
    console.log("New term created ");
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>
```

## Output:



## D. Create Child Term

### Steps:

- Add Content Editor web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
        $.getScript(scriptbase + "SP.js", function () {
            $.getScript(scriptbase + "SP.Taxonomy.js", getTerms);
        });
    });
});

var terms;
function getTerms() {
    //Create the client context and the taxonomy session object
    var clientContext = new SP.ClientContext();
    taxonomiesession = SP.Taxonomy.TaxonomySession.getTaxonomySession(clientContext);

    //Get the term store collection, term store and group object
    termStoreCollection = taxonomiesession.get_termStores();
    var termStore = termStoreCollection.getByName("Managed Metadata Service
Application");
    var group= termStore.get_groups().getByName("Previous Company");

    //Get the term set and the term object
    var termSet = group.get_termSets().getByName("Contoso Ltd")
    var term = termSet.get_terms().getByName("HR");

    //Create Child Term
    var newChildTerm = term.createTerm("HR Manager", 1033, "c49f64b3-4722-4336-9a5c-
56c326b344a9");

    //Load the client context and execute the batch
    clientContext.load(newChildTerm);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

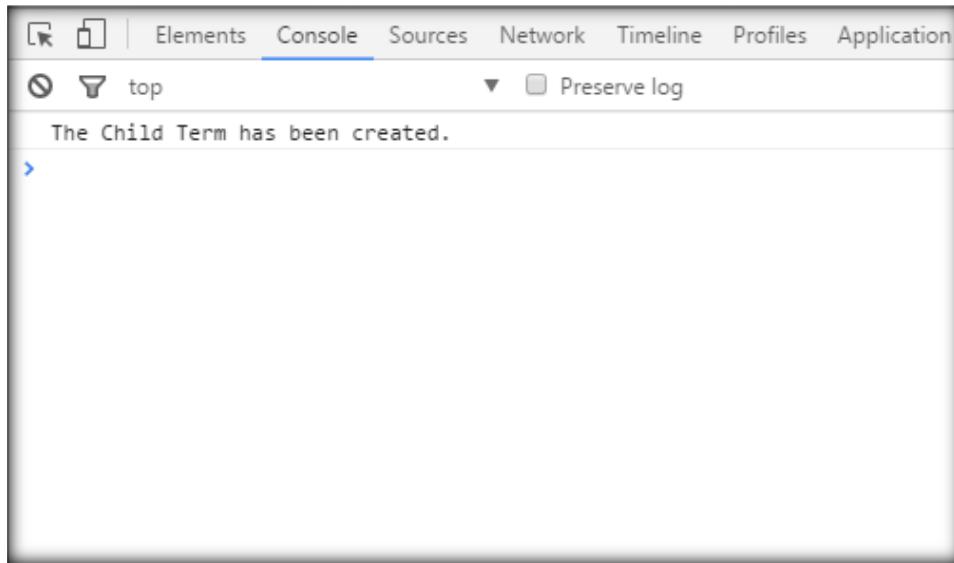
```

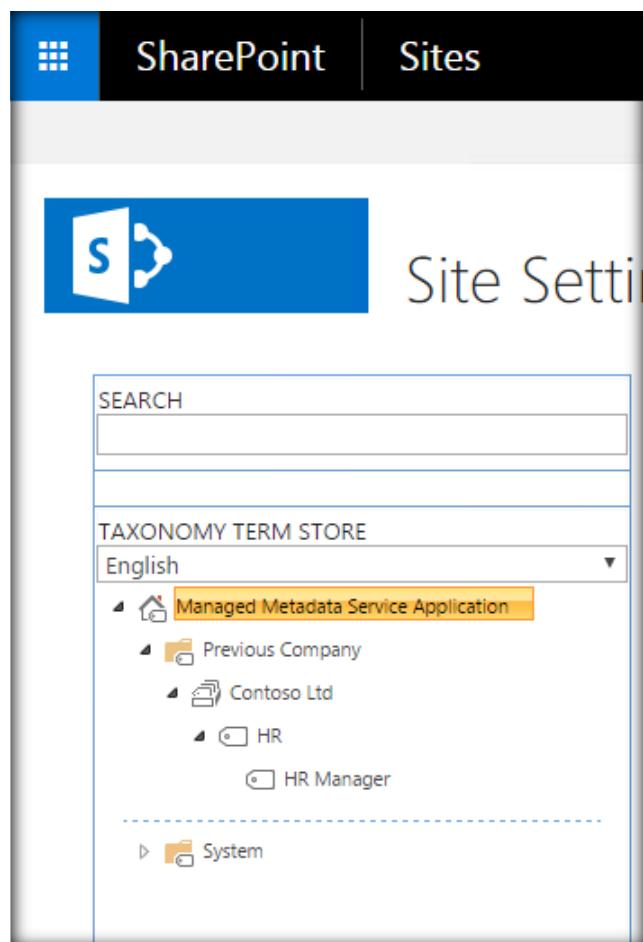
```
function QuerySuccess() {
    console.log("The Child Term has been created.");
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>
```

### Output:





## E. Create Label

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
```

```

$.getScript(scriptbase + "SP.Runtime.js", function () {
    $.getScript(scriptbase + "SP.js", function () {
        $.getScript(scriptbase + "SP.Taxonomy.js", getTerms);
    });
});

var labels;
function getTerms() {

    //Create the client context and the taxonomy session object
    clientContext = new SP.ClientContext();
    taxonomySession = SP.Taxonomy.TaxonomySession.getTaxonomySession(clientContext);

    //Get the term store collection, term store and group object
    termStoreCollection = taxonomySession.getTermStores();
    var termStore = termStoreCollection.getByName("Managed Metadata Service
Application");
    var group= termStore.getGroups().getByName("Previous Company");

    //get the term set and create a new label
    var termset = group.getTermSets().getByName("Department");
    term= termset.getTerms().getByName("SP")
    label = term.createLabel("SP Champs", 1033, false);

    //Load the client context and execute the batch
    clientContext.load(label);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

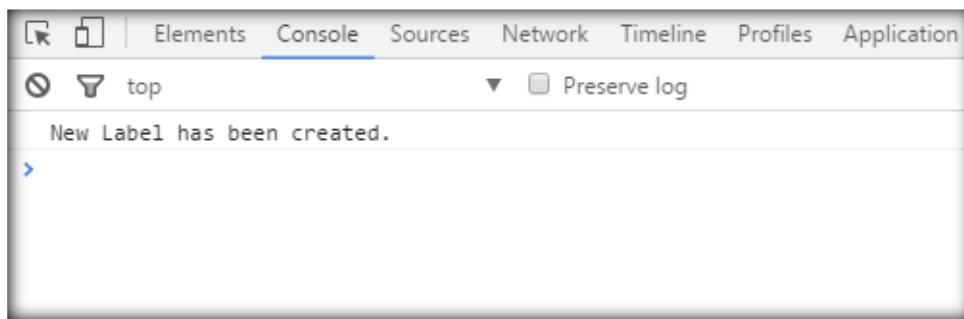
function QuerySuccess() {
    console.log("New Label has been created.");
}

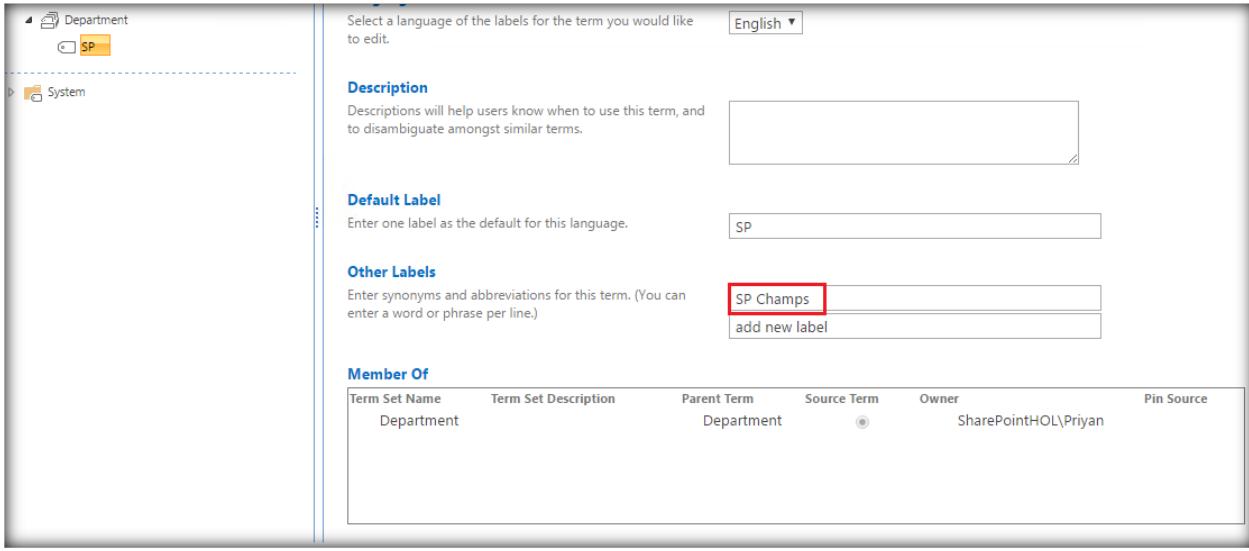
function QueryFailure(sender,args) {
    console.log('Request failed'+ args.getMessage());
}

</script>

```

## Output:





Select a language of the labels for the term you would like to edit. English ▾

**Description**  
Descriptions will help users know when to use this term, and to disambiguate amongst similar terms.

**Default Label**  
Enter one label as the default for this language. SP

**Other Labels**  
Enter synonyms and abbreviations for this term. (You can enter a word or phrase per line.) SP Champs add new label

Term Set Name	Term Set Description	Parent Term	Source Term	Owner	Pin Source
Department		Department	SP	SharePointHOL\Priyan	

## F. Copy Term

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
        $.getScript(scriptbase + "SP.js", function () {
            $.getScript(scriptbase + "SP.Taxonomy.js", getTerms);
        });
    });
});
});
```

```

var labels;
function getTerms() {

    //Create the client context and the taxonomy session object
    clientContext = new SP.ClientContext();
    taxonomySession = SP.Taxonomy.TaxonomySession.getTaxonomySession(clientContext);

    //Get the term store collection, term store and group object
    termStoreCollection = taxonomySession.getTermStores();
    var termStore = termStoreCollection.getByName("Managed Metadata Service
Application");
    var group= termStore.getGroups().getByName("Previous Company");

    //Get the term set and the term object
    var termset = group.getTermSets().getByName("Department");
    term= termset.getTerms().getByName("SP");

    //Copy the term
    var copiedTerm = term.copy(true);

    //Load the client context and execute the batch
    clientContext.load(copiedTerm);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

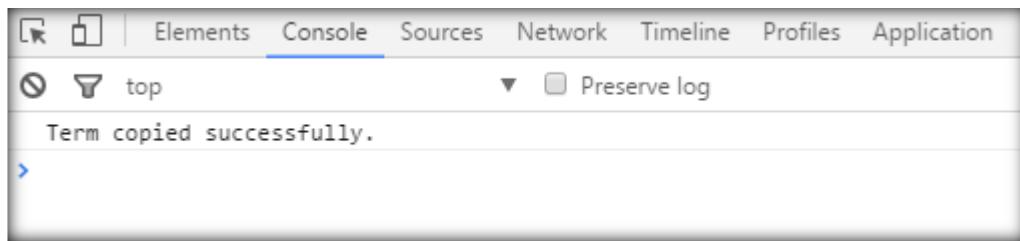
var labelsCollection;
function QuerySuccess() {
    console.log("Term copied successfully.");
}

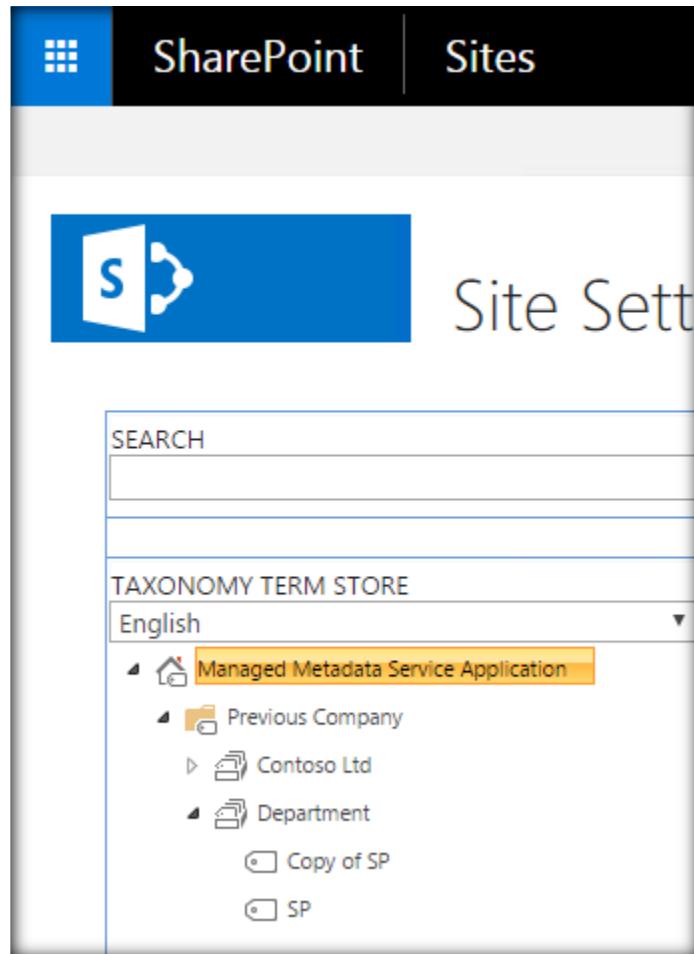
function QueryFailure(sender,args) {
    console.log('Request failed'+ args.getMessage());
}

</script>

```

## Output:





## G. Get All Labels

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";

    $.getScript(scriptbase + "SP.Runtime.js", function () {

        $.getScript(scriptbase + "SP.js", function () {

            $.getScript(scriptbase + "SP.Taxonomy.js", getTerms);

        });
    });
});

var labels;
function getTerms() {

    //Create the client context and the taxonomy session object
    clientContext = new SP.ClientContext();
    taxonomysession = SP.Taxonomy.TaxonomySession(clientContext);

    //Get term store collection,term store and group object
    termStoreCollection =taxonomysession.get_termStores();
    var termStore = termStoreCollection.getByName("Managed Metadata Service
Application");
    var group= termStore.get_groups().getByName("Previous Company");

    //Get the termset,term and labels object
    var termset = group.get_termSets().getByName("Department");
    term= termset.get_terms().getByName("SP")
    labels = term.get_labels();

    //Load the client context and execute the batch
    clientContext.load(labels);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

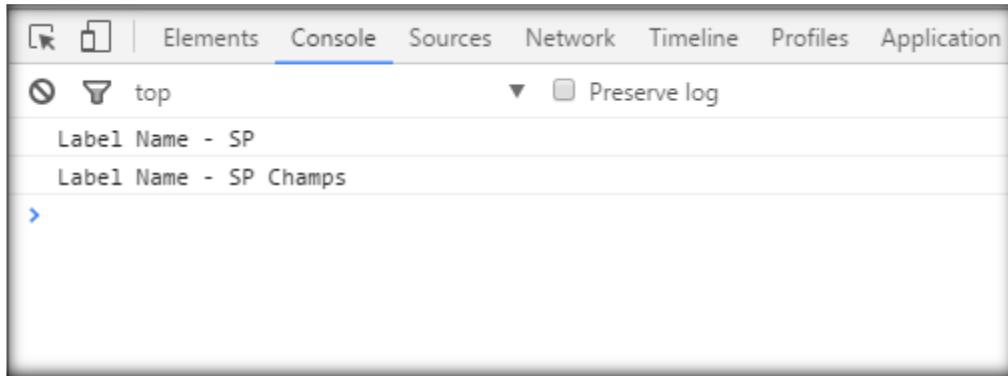
var labelsCollection;
function QuerySuccess() {

    //Enumerate the labels
    var labelsEnumerator = labels.getEnumerator();
    while (labelsEnumerator.moveNext()) {
        var label= labelsEnumerator.get_current();
        console.log("Label Name - "+label.get_value());
    }
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}
</script>

```

## Output:



## H. Get All Term Store

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
        $.getScript(scriptbase + "SP.js", function () {
            $.getScript(scriptbase + "SP.Taxonomy.js", getTerms);
        });
    });
});

var termStores;
function getTerms() {
    //Create the client context and the taxonomy session object
}
```

```

var clientContext = new SP.ClientContext();
taxonomiesession = SP.Taxonomy.TaxonomySession.getTaxonomySession(clientContext);

//Get the termstore object
termStores = taxonomiesession.get_termStores();

//Load the taxonomy session and term store object
clientContext.load(taxonomiesession);
clientContext.load(termStores);

//Execute the batch
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

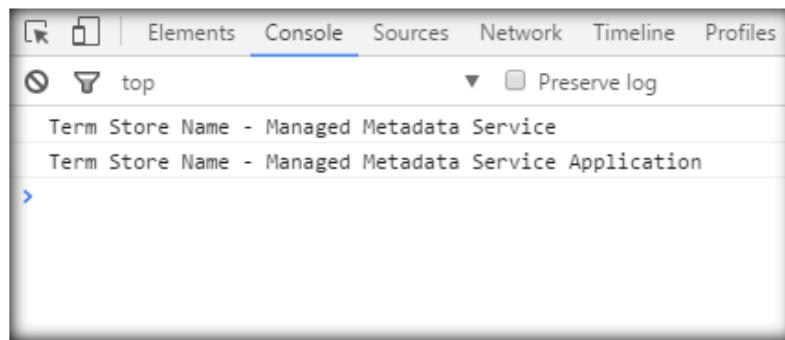
    //Enumerate the term store
    var termStoreEnumerator = termStores.getEnumerator();
    while (termStoreEnumerator.moveNext()) {
        var termStore = termStoreEnumerator.get_current();
        console.log("Term Store Name - "+termStore.get_name());
    }
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## **Output:**



## **I. Get Default Term Store**

### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

## Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
        $.getScript(scriptbase + "SP.js", function () {
            $.getScript(scriptbase + "SP.Taxonomy.js", getTerms);
        });
    });
});

var termStores;
function getTerms() {
    //Create the client context and the taxonomy session object
    var clientContext = new SP.ClientContext();
    taxonomySession = SP.Taxonomy.TaxonomySession.getTaxonomySession(clientContext);

    //Get the default site colelction term store
    termStore = taxonomySession.getDefaultSiteCollectionTermStore();

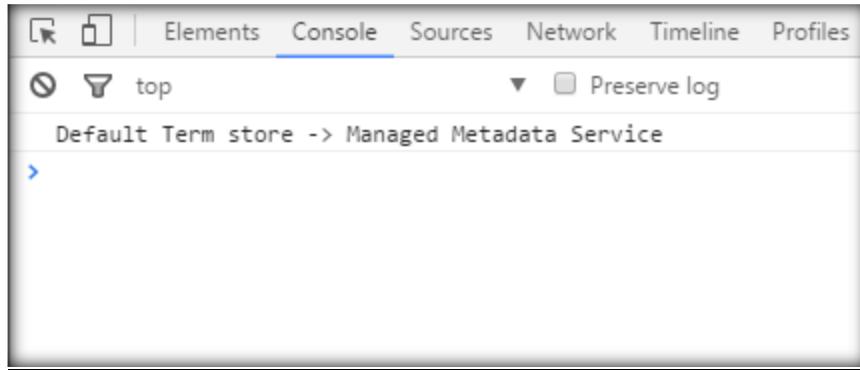
    //Load the client context and execute the batch
    clientContext.load(taxonomySession);
    clientContext.load(termStore);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    console.log("Default Term store -> " + termStore.getName());
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.getMessage());
}
</script>

```

## Output:



## J. Get All Groups

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
        $.getScript(scriptbase + "SP.js", function () {
            $.getScript(scriptbase + "SP.Taxonomy.js", getTerms);
        });
    });
});

var groups;
function getTerms() {
```

```

//Create the client context and the taxonomy session object
var clientContext = new SP.ClientContext();
taxonomiesession = SP.Taxonomy.TaxonomySession.getTaxonomySession(clientContext);

//Get the term store and group collection object
termStoreCollection = taxonomiesession.get_termStores();
var termStore = termStoreCollection.getByName("Managed Metadata Service");
groups= termStore.get_groups();

//Load the client context and execute the batch
clientContext.load(groups);
clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

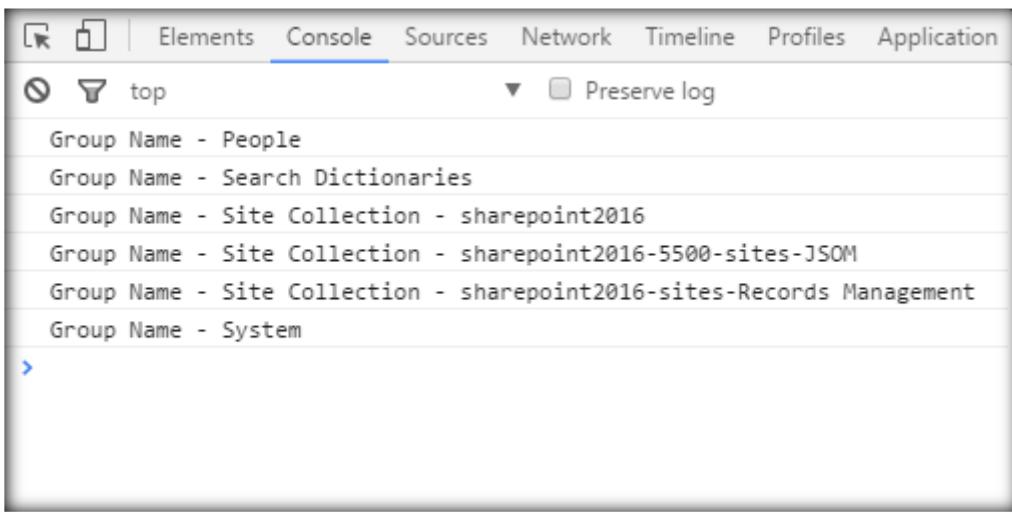
function QuerySuccess() {
    //Enumerate the group collection
    var groupsEnumerator = groups.getEnumerator();
    while (groupsEnumerator.moveNext()) {
        var group= groupsEnumerator.get_current();
        console.log("Group Name - "+group.get_name());
    }
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## Output:



The screenshot shows the Google Chrome Developer Tools with the 'Console' tab selected. The output window displays the following log entries:

```

Group Name - People
Group Name - Search Dictionaries
Group Name - Site Collection - sharepoint2016
Group Name - Site Collection - sharepoint2016-5500-sites-JSON
Group Name - Site Collection - sharepoint2016-sites-Records Management
Group Name - System

```

## K. Get Terms from Single Level

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
        $.getScript(scriptbase + "SP.js", function () {
            $.getScript(scriptbase + "SP.Taxonomy.js", getTerms);
        });
    });
});

var terms;
function getTerms() {
    //Create the client context and the taxonomy session object
    var clientContext = new SP.ClientContext();
    taxonomySession = SP.Taxonomy.TaxonomySession.getTaxonomySession(clientContext);

    //Get the term store and group object
    termStoreCollection = taxonomySession.getTermStores();
    var termStore = termStoreCollection.getByName("Managed Metadata Service
Application");
    var group= termStore.getGroups().getByName("Previous Company");

    //Get the termset and term collection
    var termset = group.getTermSets().getByName("Department");
    terms = termset.getTerms();

    //Load the client context and execute the batch
    clientContext.load(terms);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
}

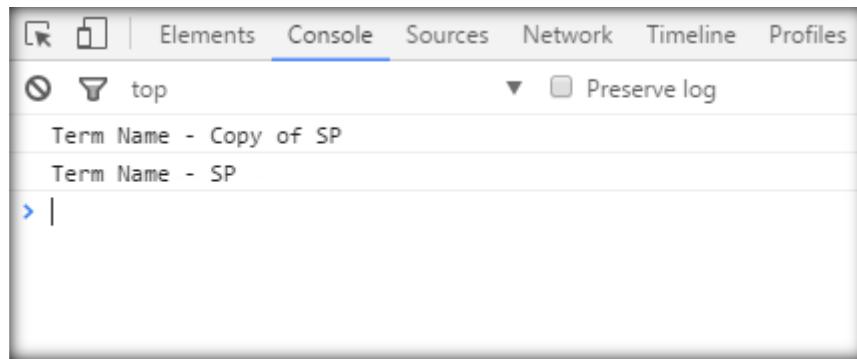
```

```
//Enumerate the Terms
var termsEnumerator = terms.getEnumerator();
while (termsEnumerator.moveNext()) {
    var term = termsEnumerator.get_current();
    console.log("Term Name - "+term.get_name());
}
}

function QueryFailure(sender,args) {
    alert('Request failed'+ args.get_message());
}

</script>
```

### Output:



### L. Get Termsets from Group

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";

    $.getScript(scriptbase + "SP.Runtime.js", function () {

        $.getScript(scriptbase + "SP.js", function () {
```

```

        $.getScript(scriptbase + "SP.Taxonomy.js", getTerms);

    });
}
};

var terms;
function getTerms() {

    //Create the client context and the taxonomy session object
    var clientContext = new SP.ClientContext();
    taxonomySession = SP.Taxonomy.TaxonomySession.getTaxonomySession(clientContext);

    //Get the term store and group object
    termStoreCollection = taxonomySession.get_termStores();
    var termStore = termStoreCollection.getByName("Managed Metadata Service
Application");
    var group= termStore.get_groups().getByName("Previous Company");

    //Get the terms collection
    terms = group.get_termSets();

    //Load the client context and execute the batch
    clientContext.load(terms);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

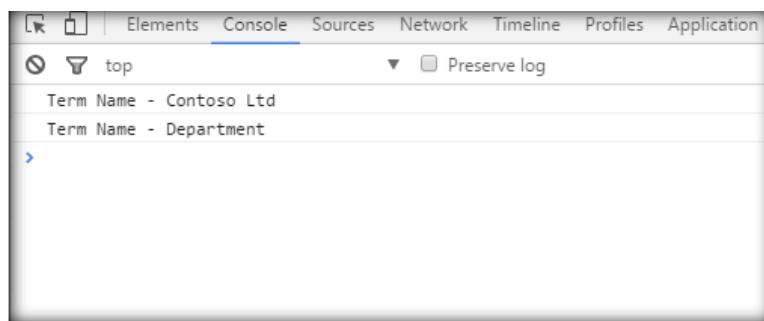
    //Enumerate the term collection
    var termsEnumerator = terms.getEnumerator();
    while (termsEnumerator.moveNext()) {
        var term= termsEnumerator.get_current();
        console.log("Term Name - "+term.get_name());
    }
}

function QueryFailure(sender,args) {
    alert('Request failed'+ args.get_message());
}

</script>

```

## Output:



## M. Get Terms from Termsets

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
        $.getScript(scriptbase + "SP.js", function () {
            $.getScript(scriptbase + "SP.Taxonomy.js", getTerms);

        });
    });
});

var terms;
function getTerms() {

    //Create the client context and the taxonomy session object
    var clientContext = new SP.ClientContext();
    taxonomysession = SP.Taxonomy.TaxonomySession(clientContext);

    //Get the term store and group object
    termStoreCollection =taxonomysession.get_termStores();
    var termStore = termStoreCollection.getByName("Managed Metadata Service
Application");
    var group= termStore.get_groups().getByName("Previous Company");

    //get the term set and the term collection
    var termset = group.get_termSets().getByName("Department");
    terms = termset.getAllTerms();

    //Load the client context and execute the batch
    clientContext.load(terms);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

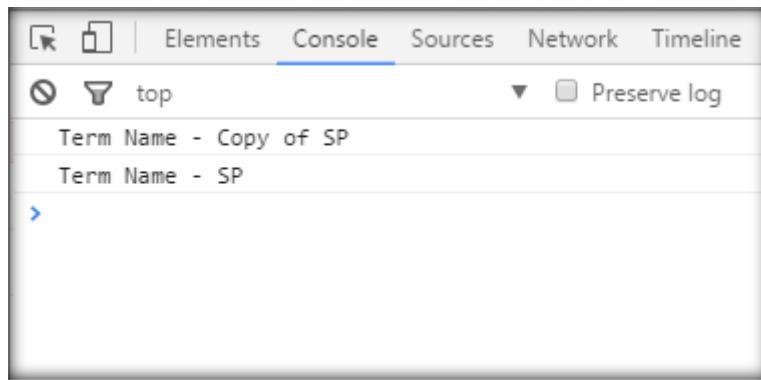
```

```
//Enumerate the terms
var termsEnumerator = terms.getEnumerator();
while (termsEnumerator.moveNext()) {
    var term = termsEnumerator.get_current();
    console.log("Term Name - "+term.get_name());
}
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>
```

### Output:



### N. Move Term

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
        $.getScript(scriptbase + "SP.js", function() {
```

```

$.getScript(scriptbase + "SP.Taxonomy.js", getTerms);

        });

    });

});

var labels;
function getTerms() {

    //Create the client context and the taxonomy session object
    clientContext = new SP.ClientContext();
    taxonomysession = SP.Taxonomy.TaxonomySession.getTaxonomySession(clientContext);

    //Get the term store and group object
    termStoreCollection =taxonomysession.get_termStores();
    var termStore = termStoreCollection.getByName("Managed Metadata Service
Application");
    var group= termStore.get_groups().getByName("HR");

    //Get the source termset,term to move and the destination termset
    var sourceTermset = group.get_termSets().getByName("Recruitment");
    term= sourceTermset.get_terms().getByName("Management");
    var destinationTermset = group.get_termSets().getByName("Payroll");

    //Move the term
    term.move(destinationTermset);

    //Execute the batch
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

var labelsCollection;
function QuerySuccess() {
    console.log("Term has been moved successfully");
}

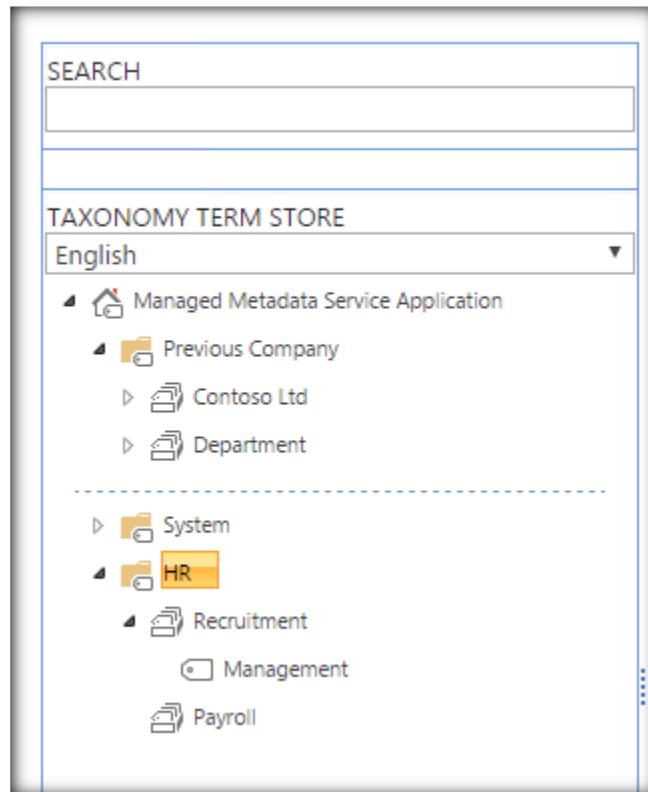
function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

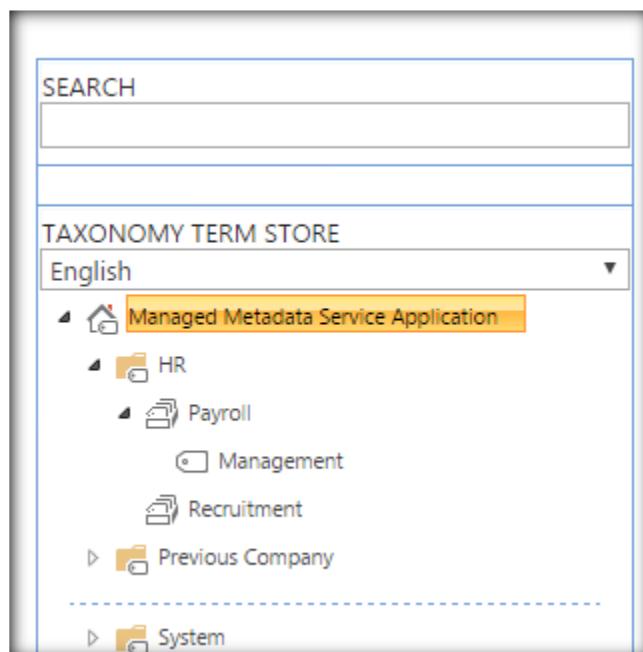
```

## Output:

Before running the script:



After running the script:



## O. Move Termset

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
        $.getScript(scriptbase + "SP.js", function () {
            $.getScript(scriptbase + "SP.Taxonomy.js", getTerms);
        });
    });
});

var labels;
function getTerms() {
    //Create the client context and the taxonomy session object
    clientContext = new SP.ClientContext();
    taxonomysession = SP.Taxonomy.TaxonomySession(clientContext);

    //Get the term store and group object
    termStoreCollection =taxonomysession.get_termStores();
    var termStore = termStoreCollection.getByName("Managed Metadata Service
Application");
    var group= termStore.get_groups().getByName("HR");

    //Get the source termset and the destination group
    var sourceTermset = group.get_termSets().getByName("Payroll");
    var destinationGroup= termStore.get_groups().getByName("Finance");

    //Move the termset and execute the batch
    sourceTermset.move(destinationGroup);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

var labelsCollection;

```

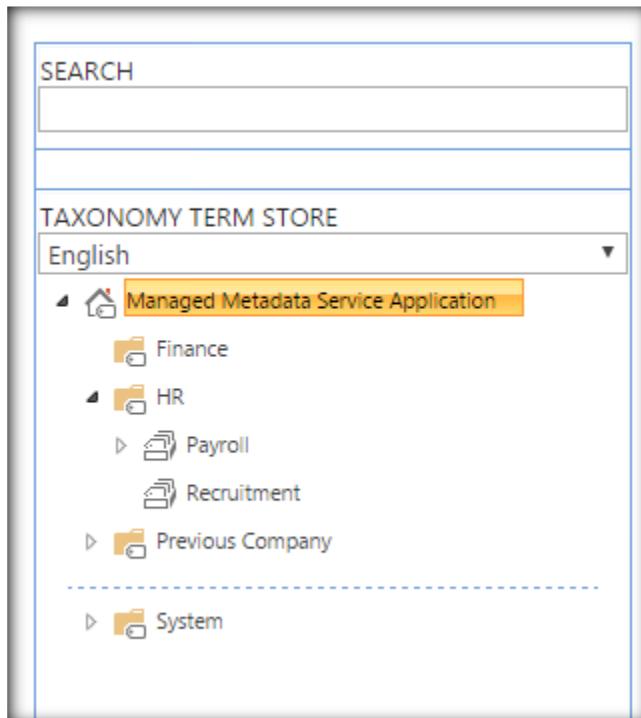
```
function QuerySuccess() {
    console.log("Termset has been moved successfully");
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

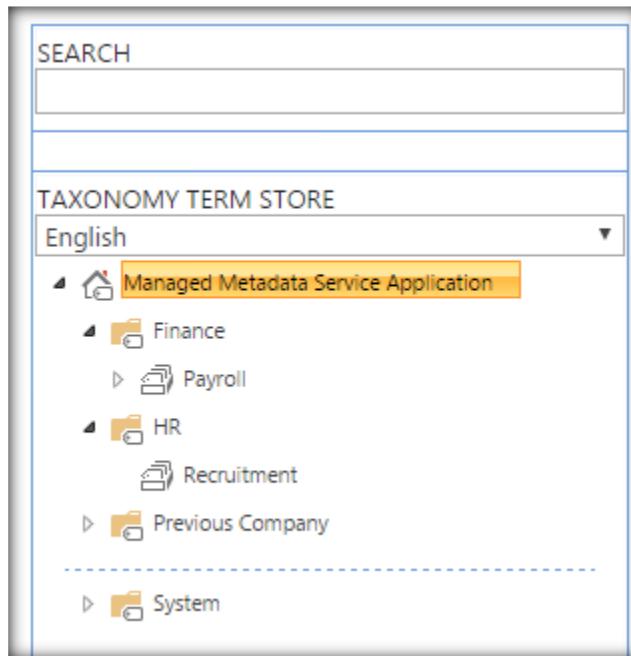
</script>
```

## Output:

Before running the script:



After running the script:



## P. Move Term as Child Term

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
        $.getScript(scriptbase + "SP.js", function () {
            $.getScript(scriptbase + "SP.Taxonomy.js", getTerms);
    });
});
});
```

```

        });
    });
});

var labels;
function getTerms() {

    //Create the client context and the taxonomy session object
    clientContext = new SP.ClientContext();
    taxonomySession = SP.Taxonomy.TaxonomySession.getTaxonomySession(clientContext);

    //Get the term store and group object
    termStoreCollection = taxonomySession.get_termStores();
    var termStore = termStoreCollection.getByName("Managed Metadata Service
Application");
    var group= termStore.get_groups().getByName("Finance");

    //Get the source termset and the destination parent term
    var sourceTermset = group.get_termSets().getByName("Payroll");
    term= sourceTermset.get_terms().getByName("Management");
    var destinationParentTerm = sourceTermset.get_terms().getByName("Insurance");

    //Move the term and execute the batch
    term.move(destinationParentTerm);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

var labelsCollection;
function QuerySuccess() {
    console.log("Term has been moved successfully");
}

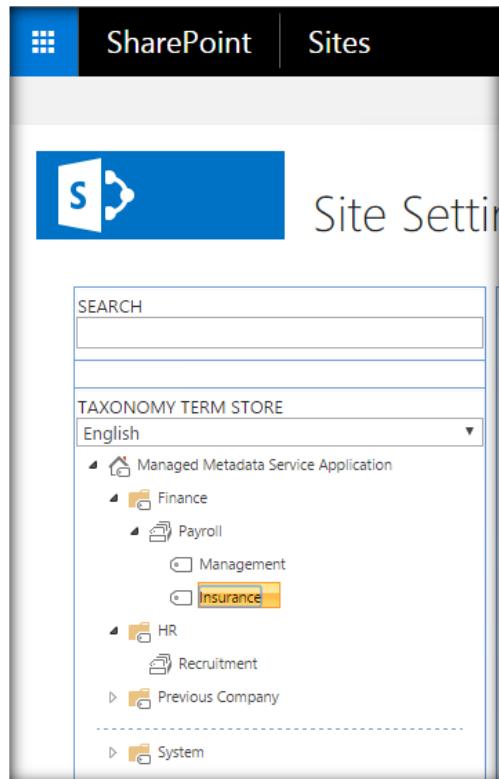
function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

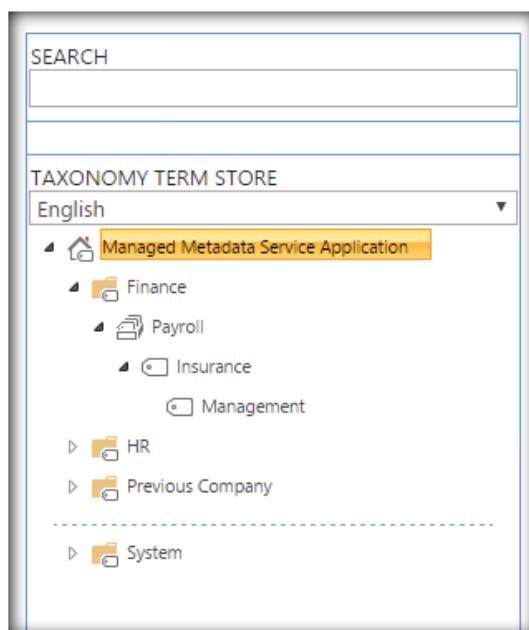
```

## Output:

Before running the script:



After running the script:



## Q. Deprecate Term

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
        $.getScript(scriptbase + "SP.js", function () {
            $.getScript(scriptbase + "SP.Taxonomy.js", getTerms);
        });
    });
});

var labels;
function getTerms() {
    //Create the client context and the taxonomy session object
    clientContext = new SP.ClientContext();
    taxonomysession = SP.Taxonomy.TaxonomySession(clientContext);

    //Get the term store and group object
    termStoreCollection =taxonomysession.get_termStores();
    var termStore = termStoreCollection.getByName("Managed Metadata Service
Application");
    var group= termStore.get_groups().getByName("Finance");

    //Get the termset and the term to deprecate
    var termset = group.get_termSets().getByName("Payroll");
    term= termset.get_terms().getByName("Insurance");

    //Deprecate the term and execute the batch
    term.deprecate(true);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

var labelsCollection;

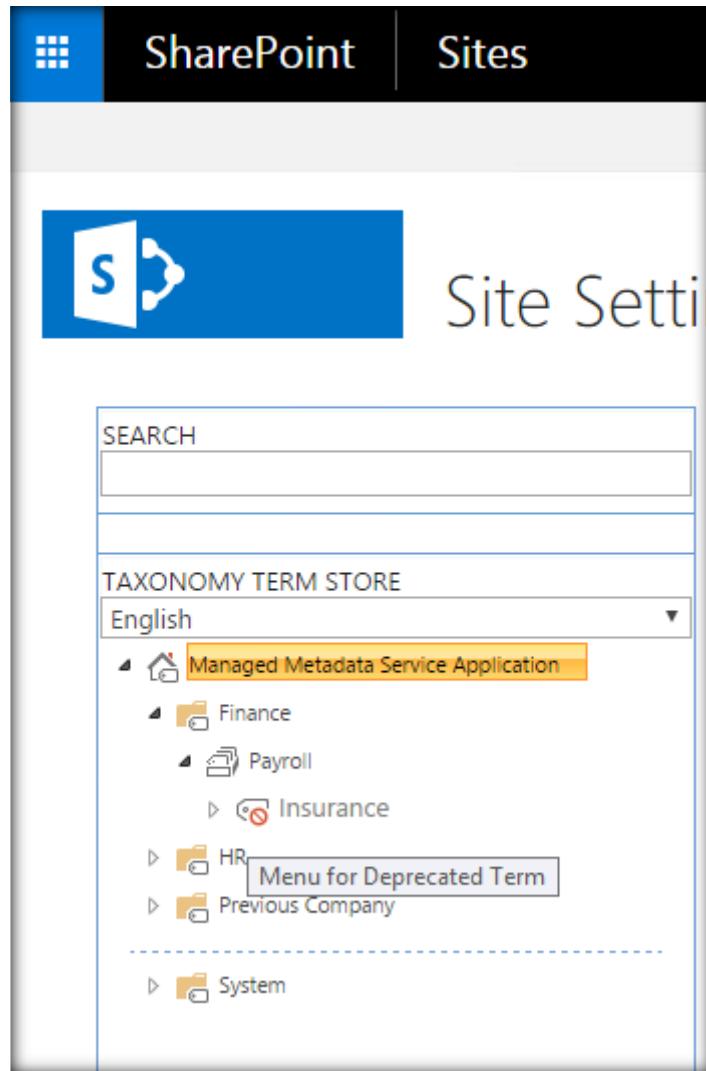
```

```
function QuerySuccess() {
    console.log("Term has been deprecated successfully");
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>
```

### Output:



## R. Re-enable Deprecated Term

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
        $.getScript(scriptbase + "SP.js", function () {
            $.getScript(scriptbase + "SP.Taxonomy.js", getTerms);
        });
    });
});

var labels;
function getTerms() {
    //Create the client context and the taxonomy session object
    clientContext = new SP.ClientContext();
    taxonomysession = SP.Taxonomy.TaxonomySession(clientContext);

    //Get the term store and group object
    termStoreCollection =taxonomysession.get_termStores();
    var termStore = termStoreCollection.getByName("Managed Metadata Service
Application");
    var group= termStore.get_groups().getByName("Finance");

    //Get the termset and the term
    var termset = group.get_termSets().getByName("Payroll");
    term= termset.get_terms().getByName("Insurance");

    //Deprecate the term and execute the batch
    term.deprecate(false);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

var labelsCollection;

```

```

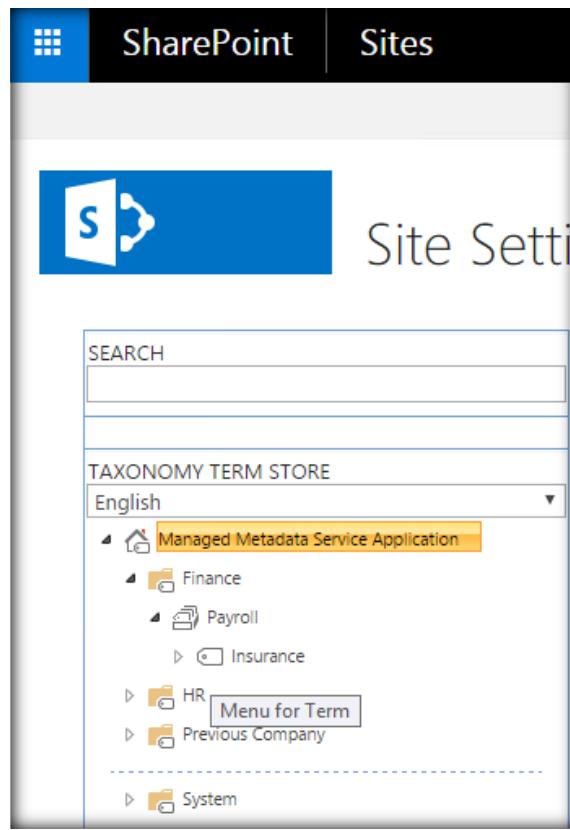
function QuerySuccess() {
    console.log('Deprecated term has been re-enabled');
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

```

## Output:



## S. Delete Label

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

## Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
        $.getScript(scriptbase + "SP.js", function () {
            $.getScript(scriptbase + "SP.Taxonomy.js", getTerms);
        });
    });
});

var labels;
function getTerms() {
    //Create the client context and the taxonomy session object
    clientContext = new SP.ClientContext();
    taxonomiesession = SP.Taxonomy.TaxonomySession.getTaxonomySession(clientContext);

    //Get the term store and group object
    termStoreCollection = taxonomiesession.get_termStores();
    var termStore = termStoreCollection.getByName("Managed Metadata Service
Application");
    var group= termStore.get_groups().getByName("Previous Company");

    //Get the termset and the term
    var termset = group.get_termSets().getByName("Department");
    var term= termset.get_terms().getByName("SP");

    //Get the label and delete the object
    var label = term.get_labels().getByValue("SP Champs");
    label.deleteObject();

    //Execute the batch
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

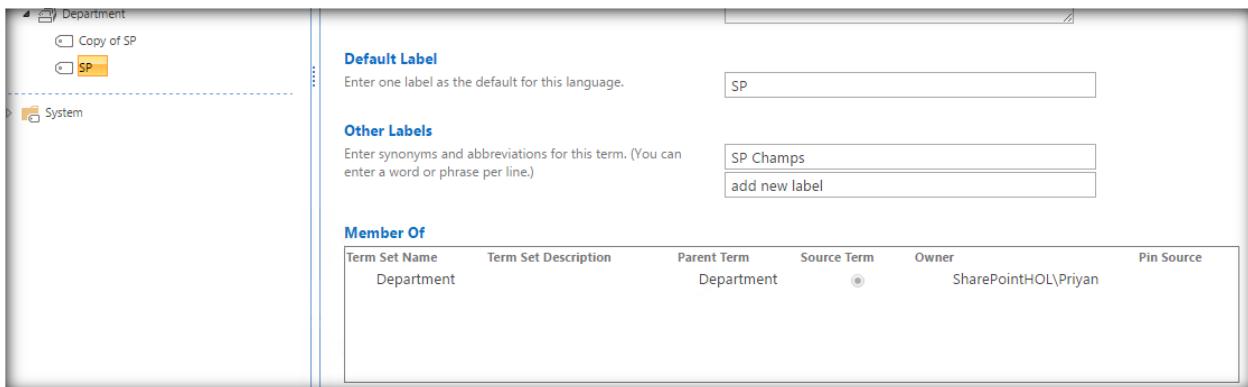
function QuerySuccess() {
    console.log('Label has been deleted');
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}
</script>

```

## Output:

Before running the script:



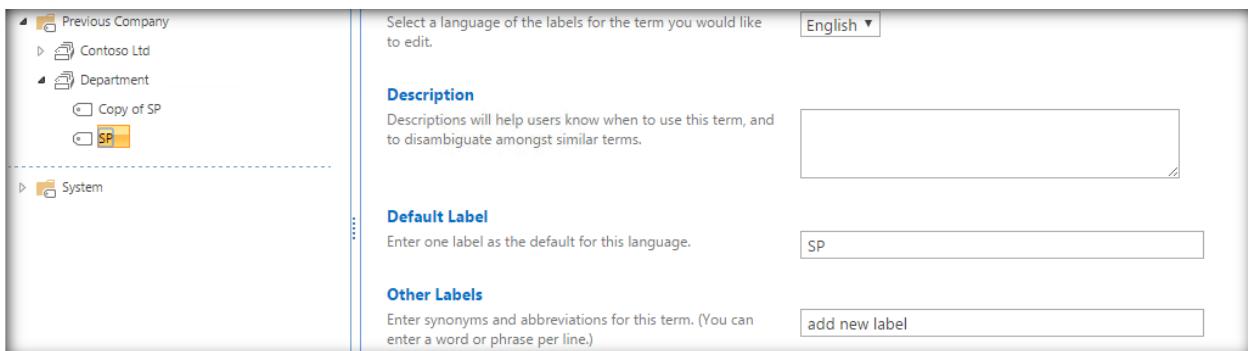
**Default Label**  
Enter one label as the default for this language.

**Other Labels**  
Enter synonyms and abbreviations for this term. (You can enter a word or phrase per line.)  
  
[add new label](#)

**Member Of**

Term Set Name	Term Set Description	Parent Term	Source Term	Owner	Pin Source
Department		Department		SharePointHOL\Priyan	

After running the script:



Select a language of the labels for the term you would like to edit.

**Description**  
Descriptions will help users know when to use this term, and to disambiguate amongst similar terms.

**Default Label**  
Enter one label as the default for this language.

**Other Labels**  
Enter synonyms and abbreviations for this term. (You can enter a word or phrase per line.)  
[add new label](#)

## T. Delete Terms

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

## Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
        $.getScript(scriptbase + "SP.js", function () {
            $.getScript(scriptbase + "SP.Taxonomy.js", getTerms);
        });
    });
});

var labels;
function getTerms() {
    //Create the client context and the taxonomy session object
    clientContext = new SP.ClientContext();
    taxonomysession = SP.Taxonomy.TaxonomySession.getTaxonomySession(clientContext);

    //Get the term store and group object
    termStoreCollection = taxonomysession.get_termStores();
    var termStore = termStoreCollection.getByName("Managed Metadata Service
Application");
    var group= termStore.get_groups().getByName("Previous Company");

    //Get the termset and the term to delete
    var termset = group.get_termSets().getByName("Department");
    term= termset.get_terms().getByName("Copy of SP")

    //Delete the term and execute the batch
    term.deleteObject();
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

var labelsCollection;
function QuerySuccess() {
    console.log('Term has been deleted');
}

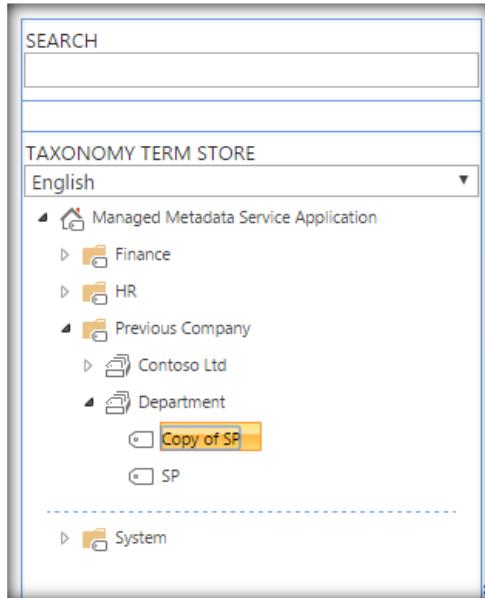
function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

</script>

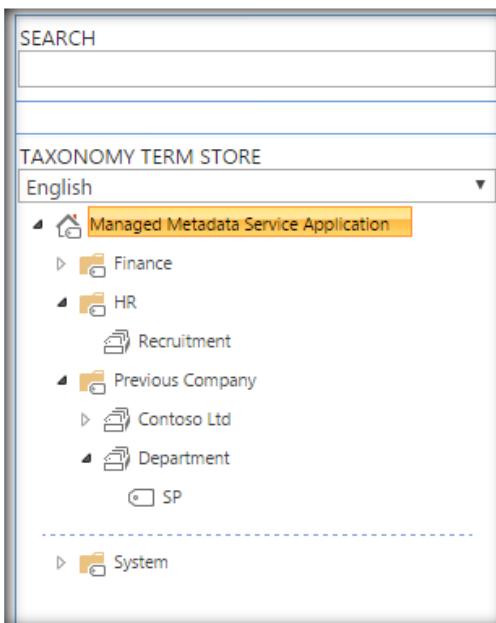
```

## **Output:**

Before running the script:



After running the script:



## U. Delete Termset

### **Steps:**

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### **Script:**

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {

    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";

    $.getScript(scriptbase + "SP.Runtime.js", function () {

        $.getScript(scriptbase + "SP.js", function () {

            $.getScript(scriptbase + "SP.Taxonomy.js", getTerms);

        });

    });
});

var labels;
function getTerms() {

    //Create the client context and the taxonomy session object
    clientContext = new SP.ClientContext();
    taxonomiesession = SP.Taxonomy.TaxonomySession.getTaxonomySession(clientContext);

    //Get the term store and group object
    termStoreCollection = taxonomiesession.get_termStores();
    var termStore = termStoreCollection.getByName("Managed Metadata Service
Application");
    var group= termStore.get_groups().getByName("Previous Company");

    //Get the termset to delete and delete the object
    var termset = group.get_termSets().getByName("Department");
    termset.deleteObject();

    //Execute the batch
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

```

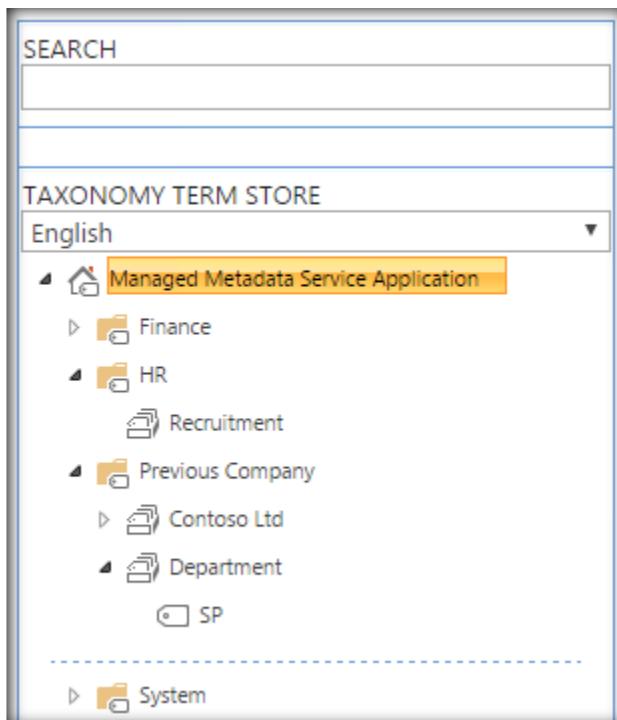
```
        console.log('Termset has been deleted');
    }

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}

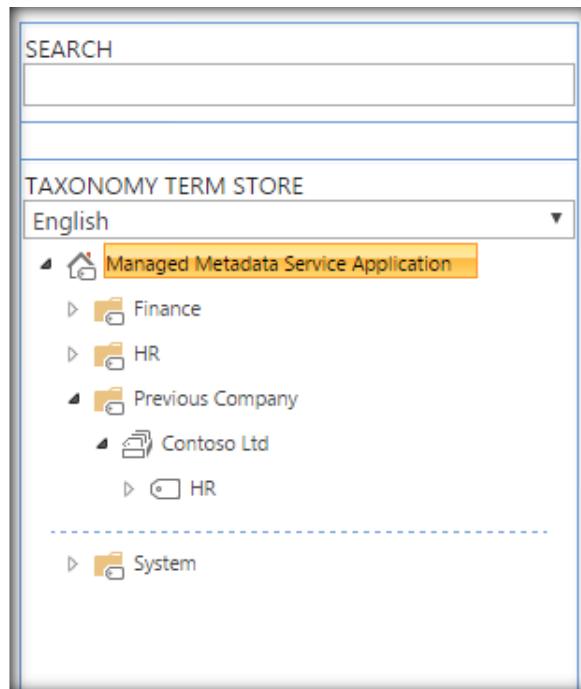
</script>
```

## Output:

Before running the script:



After running the script:



## V. Delete Group

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    var scriptbase = _spPageContextInfo.webServerRelativeUrl + "/_layouts/15/";
    $.getScript(scriptbase + "SP.Runtime.js", function () {
        $.getScript(scriptbase + "SP.js", function () {
            $.getScript(scriptbase + "SP.Taxonomy.js", getTerms);
        });
    });
});
```

```

    );
});

var labels;
function getTerms() {

    //Create the client context and the taxonomy session object
    clientContext = new SP.ClientContext();
    taxonomySession = SP.Taxonomy.TaxonomySession.getTaxonomySession(clientContext);

    //Get the term store and group object to delete
    termStoreCollection = taxonomySession.get_termStores();
    var termStore = termStoreCollection.getByName("Managed Metadata Service
Application");
    var group= termStore.get_groups().getByName("HR");

    //Delete the group object and execute the batch
    group.deleteObject();
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

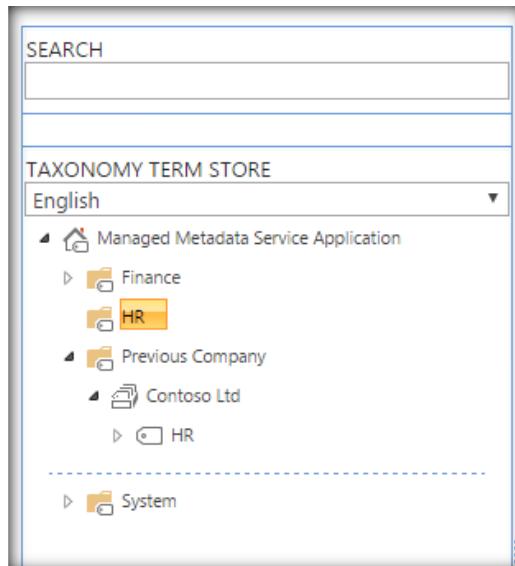
function QuerySuccess() {
    console.log('Group has been deleted');
}

function QueryFailure(sender,args) {
    console.log('Request failed'+ args.get_message());
}
</script>

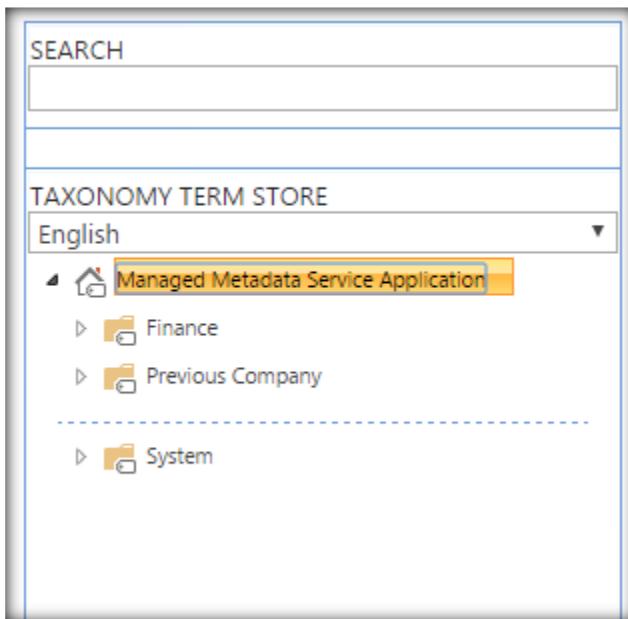
```

## Output:

Before running the script:



After running the script:



## XXIII. Working with Custom User action

### A. Add Custom Action to ECB menu

#### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

#### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', AddCustomUserActionToECB);
});

var oListItem;
function AddCustomUserActionToECB() {

    //Get the client context,web and list object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();
    var oList = oWeb.get_lists().getByTitle('Test List');

    //Create the custom user action and set the properties for adding to ECB Menu
    var userCustomActionColl = oList.get_userCustomActions();
    var oUserCustomAction = userCustomActionColl.add();
    oUserCustomAction.set_location('EditControlBlock');
    oUserCustomAction.set_sequence(100);
    oUserCustomAction.set_title("Go to Landing Page");

    oUserCustomAction.set_url("/sites/HOL/Pages/LandingPage.aspx?ListId={ListId}&ItemId={ItemId}&ItemUrl={ItemUrl}");
    oUserCustomAction.update();

    //Load the client context and execute the batch
    clientContext.load(userCustomActionColl);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    console.log("Custom Action added to ECB menu.");
}

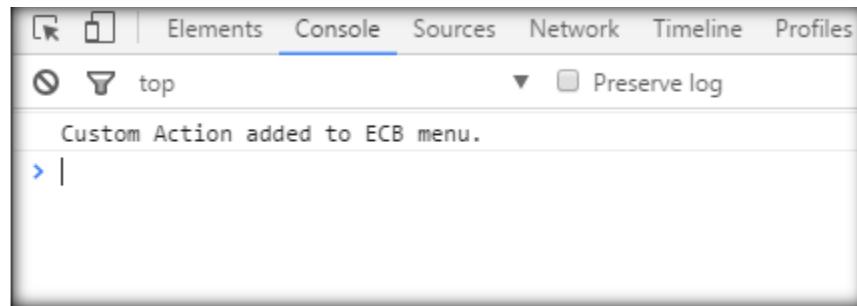
function QueryFailure() {
}

```

```
        console.log('Request failed - ' + args.get_message());
    }

</script>
```

## Output:



A screenshot of a SharePoint 'Test List' page. The page title is 'Test List'. There is a 'new item or edit this list' button. Below it, there is a search bar and a 'Find an item' button. A list of items is shown, with the first item, 'Test Item \*', selected. A context menu is open over this item, displaying options: 'Go to Landing Page', 'Edit Item', 'Delete Item', 'View Item', and 'Advanced'. The 'Edit Item' option is highlighted with a blue border.

## B. Add Custom Action to Site Settings

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {
    SP.SOD.executeFunc('sp.js', 'SP.ClientContext', AddCustomUserAction);
});

function AddCustomUserAction() {

    //Get the client context and web object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get the custom user action collection and add the new custom action to it
    var collUserCustomAction = oWeb.get_userCustomActions();
    var oUserCustomAction = collUserCustomAction.add();

    //Specify the location and properties for the new custom action
    oUserCustomAction.set_location('Microsoft.SharePoint.StandardMenu');
    oUserCustomAction.set_sequence(101);
    oUserCustomAction.set_group('SiteActions');
    oUserCustomAction.set_title("Search Settings");

    oUserCustomAction.set_url("/sites/HOL/_layouts/15/enhancedSearch.aspx?level=sitecol");
    oUserCustomAction.update();

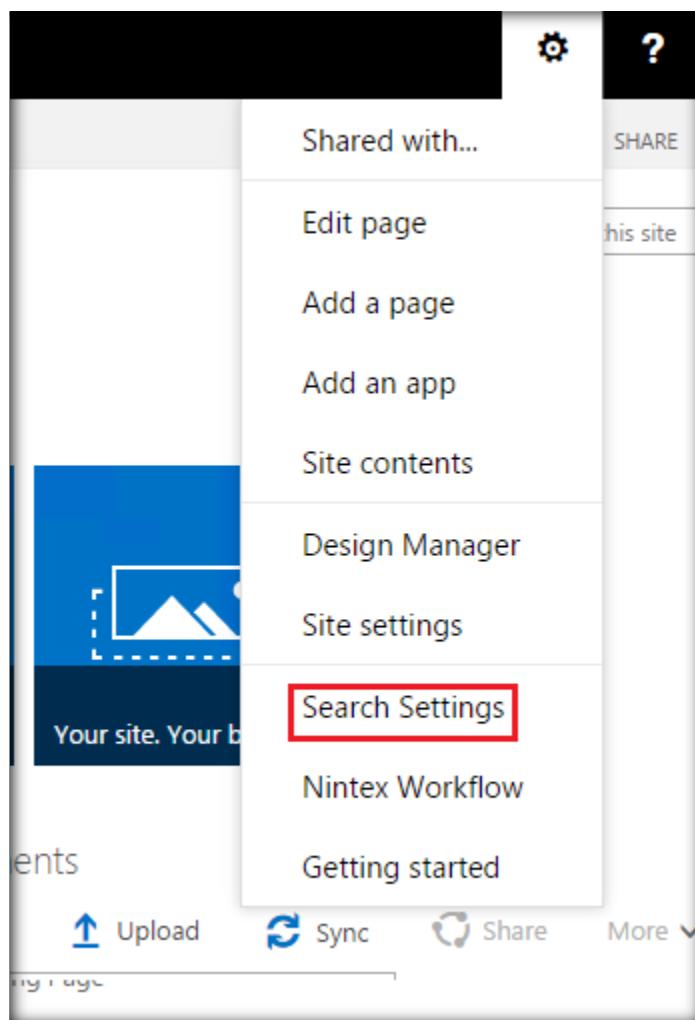
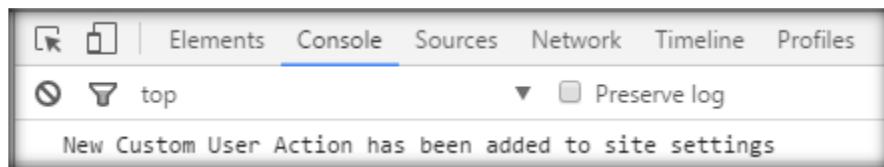
    //Load the client context and execute the batch
    clientContext.load(collUserCustomAction);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    console.log("New Custom User Action has been added to site settings");
}

function QueryFailure() {
    console.log(args.get_message());
}

</script>

```

**Output:**

## C. Add Custom Action to Ribbon

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    SP.SOD.executeOrDelayUntilScriptLoaded(AddCustomUserAction, "sp.js");
});

var oListItem;
function AddCustomUserAction() {
    //Get the client context and list object
    var context = new SP.ClientContext.get_current();
    var list = context.get_web().get_lists().getByTitle("Ribbon List");

    //Get the custom user action collection and add the user action
    var customUserAction = list.get_userCustomActions().add();

    //Set the location of the user action
    customUserAction.set_location('CommandUI.Ribbon.ListView');

    //Add the properties for the custom action
    var userActionExtension = '<CommandUIExtension
xmlns="http://schemas.microsoft.com/sharepoint/">' +
        '<CommandUIDefinitions>' +
        '<CommandUIDefinition
Location="Ribbon>List\CustomizeList\Controls._children">' +
            '<Button Id="Ribbon.Documents.New.RibbonTest" ' +
                'Command="Notify" ' +
                'Sequence="0" ' +
                'Image16by16="/_layouts/images/NoteBoard_16x16.png" ' +
                'Image32by32="/_layouts/images/NoteBoard_32x32.png" ' +
                'Description="Shows the ID of the current list." ' +
                'LabelText="Show List ID" ' +
                'TemplateAlias="o1"/>' +
        '</CommandUIDefinition>' +
        '</CommandUIDefinitions>' +
        '<CommandUIHandlers>' +
        '<CommandUIHandler Command="Notify" ' +
        'CommandAction="javascript:SP.UI.Notify.addNotification(\\"ListId={ListId}\\");"' +
}

```

```

/>'+

'</CommandUIHandlers>'+
'</CommandUIExtension>';

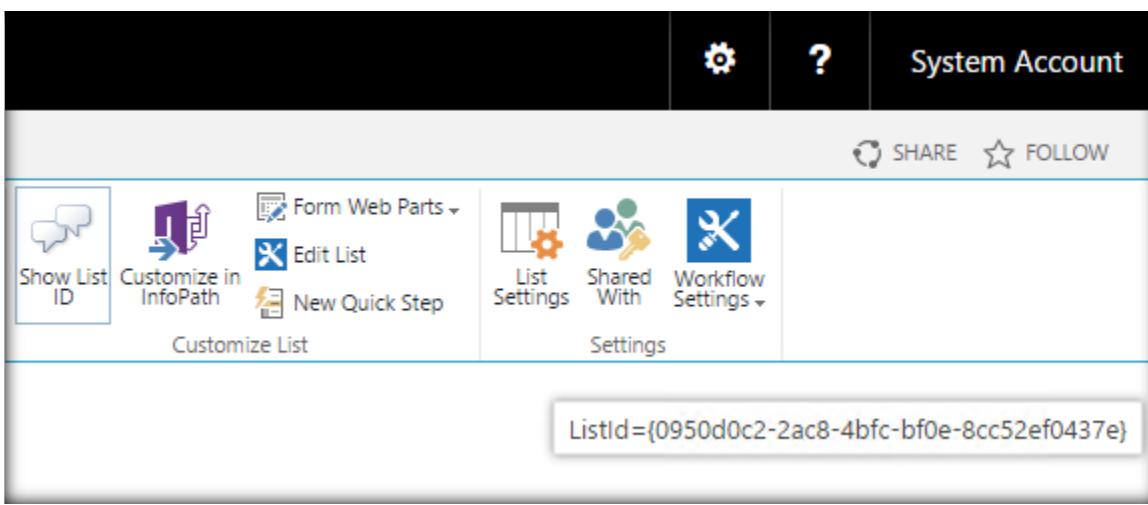
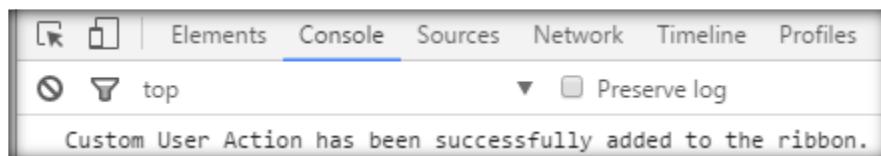
//Add the command UI extension and update the custom user action
customUserAction.set_commandUIExtension(userActionExtension)
customUserAction.update();

//Load the client context and execute the batch
context.load(list,'UserCustomActions');
context.executeQueryAsync(function() {
    console.log("Custom User Action has been successfully added to the ribbon
in the list.");
},
function(sender, args) {
    console.log("Error Occurred - "+args.get_message());
});

}
</script>

```

## Output:



## D. Add Custom Action to Site Actions

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

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

$(document).ready(function() {

    SP.SOD.executeOrDelayUntilScriptLoaded(AddCustomUserAction, "sp.js");

});

var oListItem;
function AddCustomUserAction() {

    //Get the client context and web object
    var clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();

    //Get the custom user action collection and add the user action
    var collUserCustomAction = oWeb.get_userCustomActions();
    var oUserCustomAction = collUserCustomAction.add();

    //Set the custom user action properties
    oUserCustomAction.set_location('Microsoft.SharePoint.SiteSettings');
    oUserCustomAction.set_group('SiteTasks');
    oUserCustomAction.set_sequence(1000);
    oUserCustomAction.set_title('Add an app to the site ');
    oUserCustomAction.set_description('This custom action facilitates the user to add
an app to the page.');
    oUserCustomAction.set_url("sites/HOL/_layouts/15/addanapp.aspx");
    oUserCustomAction.update();

    //Load the client context and execute the batch
    clientContext.load(collUserCustomAction);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

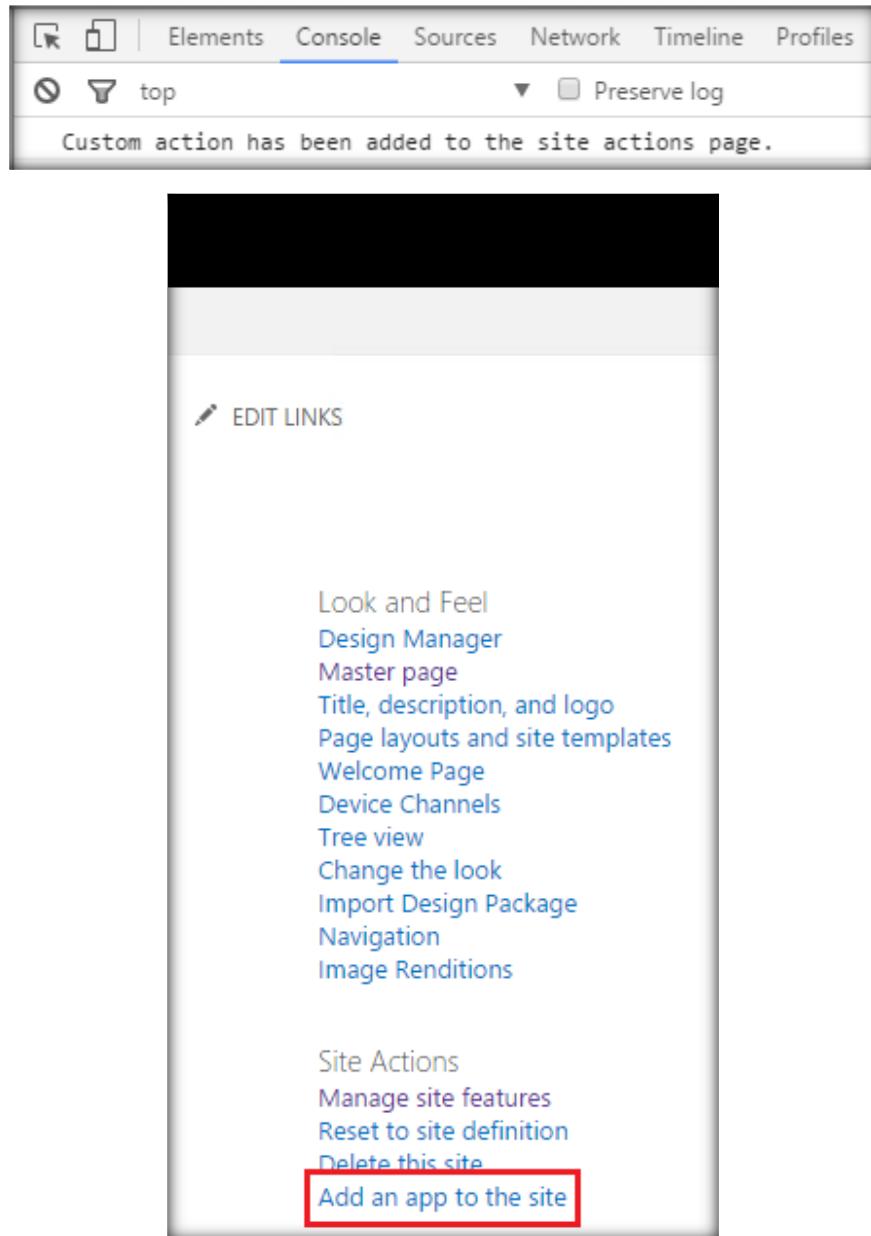
function QuerySuccess() {
    console.log("Custom action has been added to the site actions page.");
}

```

```
function QueryFailure() {
    console.log(args.get_message());
}

</script>
```

## Output:



## E. Get Available List Custom Actions

### Steps:

- Add Content Editor web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    SP.SOD.executeOrDelayUntilScriptLoaded(EditCustomUserAction, "sp.js");
});

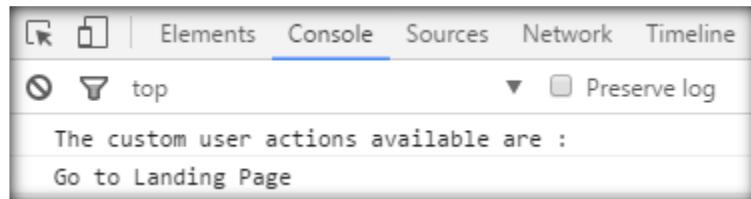
var oList, item, userCustomActionColl, clientContext;
function EditCustomUserAction() {
    //Get the client context, web and list object
    clientContext = new SP.ClientContext();
    var oWeb = clientContext.get_web();
    var oList = oWeb.get_lists().getByTitle('Test List');

    //Get the custom user action collection
    userCustomActionColl = oList.get_userCustomActions();

    //Load the client context and execute the batch
    clientContext.load(userCustomActionColl);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {
    //Enumerate the custom user actions
    var customActionEnumerator = userCustomActionColl.getEnumerator();
    console.log("The custom user actions available are :");
    while (customActionEnumerator.moveNext())
    {
        var oUserCustomAction = customActionEnumerator.get_current();
        console.log(oUserCustomAction.get_title());
    }
}

function QueryFailure() {
    console.log('Request failed');
}
</script>
```

**Output:**

The custom user actions available are :

Go to Landing Page

Sites

HOL WebPartPage LandingPage EDIT LINKS

## Test List

[⊕ new item or edit this list](#)

All Items ... Find an item

✓ Title

✓ Test Item \*

...

Go to Landing Page

Edit Item Go to Landing Page

Delete Item

View Item

Advanced ▾

A screenshot of a SharePoint site titled "Test List". In the browser's developer tools console, it shows the output of custom user actions: "The custom user actions available are :" followed by "Go to Landing Page". On the SharePoint page, a context menu is open over an item named "Test Item". The menu includes options like "Go to Landing Page", "Edit Item", "Delete Item", "View Item", and "Advanced".

## F. Edit Custom Action

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```
<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    SP.SOD.executeOrDelayUntilScriptLoaded(EditCustomUserAction, "sp.js");
});

var oListItem,userCustomActionColl,clientContext;
function EditCustomUserAction() {

    //Get the client context, web and list object
    clientContext = new SP.ClientContext();
    var oWebsite = clientContext.get_web();
    var oList = oWebsite.get_lists().getByTitle('Test List');

    //Get the custom user action collection
    userCustomActionColl = oList.get_userCustomActions();

    //Load the client context and execute the batch
    clientContext.load(userCustomActionColl);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

    //Enumerate the custom user actions
    var customActionEnumerator = userCustomActionColl.getEnumerator();

    while (customActionEnumerator.moveNext())
    {
        var oUserCustomAction = customActionEnumerator.get_current();

        //Edit the custom action title
        if (oUserCustomAction.get_title() == "Go to Landing Page")
        {
            oUserCustomAction.set_title('Take me to Landing Page');
            oUserCustomAction.update();
        }
    }
}

```

```

//Load the client context and execute the batch
clientContext.load(oUserCustomAction);
clientContext.executeQueryAsync(OnSuccess,OnFailure);
}

}

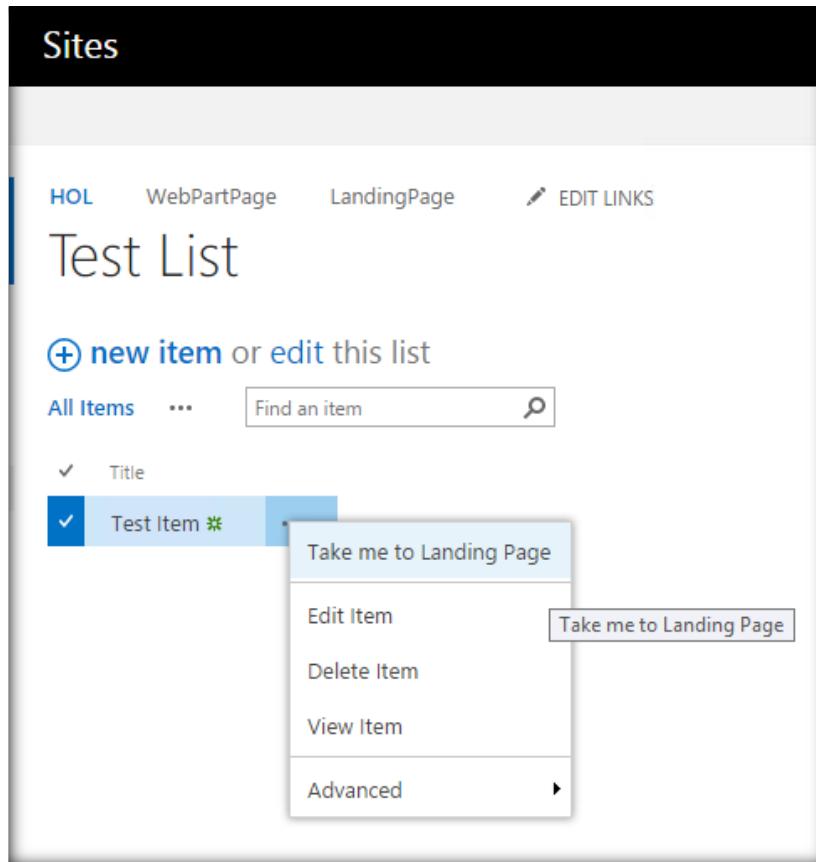
function QueryFailure() {
    console.log('Request failed');
}

function OnSuccess() {
    console.log('Request succeeded in editing the custom action');
}

function OnFailure() {
    console.log('Request failed');
}
</script>

```

### Output:



## G. Delete Custom Action

### Steps:

- Add Content Editor Web part (CEWP) to the SharePoint page.
- Save the below script as a text file and upload it to Site Assets page.
- Refer the script file from the CEWP.

### Script:

```

<script language="javascript" type="text/javascript"
src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>

<script language="javascript" type="text/javascript">
$(document).ready(function() {
    SP.SOD.executeOrDelayUntilScriptLoaded(EditCustomUserAction, "sp.js");
});

var oListItem,userCustomActionColl,clientContext;
function EditCustomUserAction() {

    //Get the client context, web and list object
    clientContext = new SP.ClientContext();
    var oWebsite = clientContext.get_web();
    var oList = oWebsite.get_lists().getByTitle('Test List');

    //Get the custom user action collection
    userCustomActionColl = oList.get_userCustomActions();

    //Load the client context and execute the batch
    clientContext.load(userCustomActionColl);
    clientContext.executeQueryAsync(QuerySuccess, QueryFailure);
}

function QuerySuccess() {

    //Enumerate the custom user actions
    var customActionEnumerator = userCustomActionColl.getEnumerator();
    var itemsToDelete= new Array();
    while (customActionEnumerator.moveNext())
    {
        var oUserCustomAction = customActionEnumerator.get_current();
        //Select the custom action to delete
        if (oUserCustomAction.get_title() == "Take me to Landing Page")
        {
            itemsToDelete.push(oUserCustomAction);
        }
    }
}

```

```

}

//Delete the user action
for (var i = itemsToDelete.length-1; i >= 0 ; i--) {
    itemsToDelete[i].deleteObject();
    //Execute the batch
    clientContext.executeQueryAsync(OnSuccess,OnFailure);
}
}

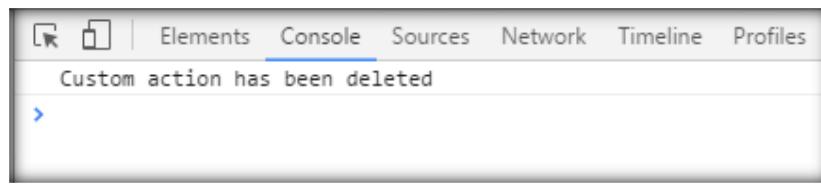
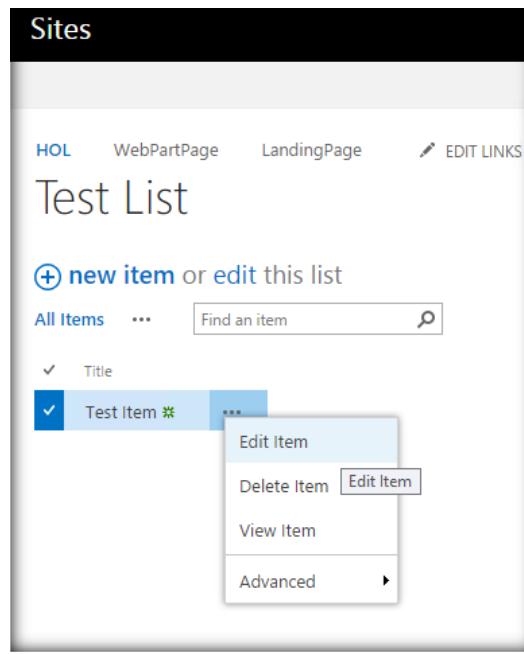
function QueryFailure() {
    console.log(args.get_message());
}

function OnSuccess() {
    console.log("Custom action has been deleted");
}

function OnFailure() {
    console.log(args.get_message());
}
</script>

```

## Output:

The screenshot shows a SharePoint site titled "Sites" with a sub-page named "Test List". At the top, there are navigation links: HOL, WebPartPage, LandingPage, and EDIT LINKS. Below the header, there's a search bar with the placeholder "Find an item" and a magnifying glass icon. A blue button labeled "+ new item or edit this list" is visible. In the main content area, a list item titled "Test Item \*" is selected. A context menu is open over this item, displaying the following options: "Edit Item", "Delete Item", "Edit Item" (disabled), "View Item", and "Advanced".

## XXIV. Summary

Thus, we have seen how to program against SharePoint, using JavaScript Object Model. Close to 300 JavaScript Object Model scripts, that are used commonly for operating with SharePoint, have been discussed through out the book. This is just a jump start to client side programming with SharePoint. Bill Baer, Senior Product Marketing Manager at Microsoft has promised that SharePoint 2016 will not be the last iteration of On Premise SharePoint. SharePoint is here to stay and will live really long. So, let's get our hands dirty and do some SharePointing!