SharePoint JSOM Basics

Contents

Deployment Patters for On-Premise & Office 365

What is JSOM

- JSOM is an inofficial but commonly used term for the JavaScript implementation of the SharePoint Client Object Model
- Stands for JavaScript Object Model
- Implemented in sp.js
- SP namespace documented @ https://msdn.microsoft.com/en-us/library/office/jj246996.aspx
- Many additional files for
  - Base SharePoint
  - Service Applicaitons

Out-Of-Box References

- SP.js – Client side object model
- Core.js - Include dropdown menu items, page layout manipulation, expand/collapse behavior on list views, etc.
- Menu.js – Core menu object
- Callout.js - Callouts
- Sharing.js – Sharing & Permissions
- Init.js – contains helper objects like SP.ScriptUtiltity

Deployment Locations

Serveral options for deployments of JavaScript based solutions

On Premise

- Site Assets or any other DocumentLibrary
- /_layouts/15/1033/Subfolder
- /_layouts/15/ApplicationPageFolder

Office 365

- Site Assets or any other DocumentLibrary

Registering & Loading JavaScript

ScriptLink

- Point to Default Location

  {SharePointRoot}\Template\LAYOUTS\1033

- Localizable=„False“

  {SharePointRoot}\Template\LAYOUTS

- Uses SPUtility.MakeBrowserCacheSafeLayoutsUr method to prevent Caching

```
<script src="/_layouts/myscript.js?rev=7KqI9%2FoL9hClomz1RdzTqg%3D%3D" type="text/javascript"></script>
```

ScriptLink OnDemand

Delays Loading of Script using RegisterSod

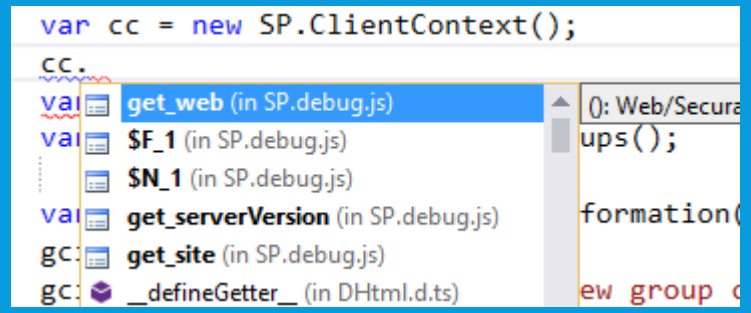
Used with LoadSod or LoadSodByKey

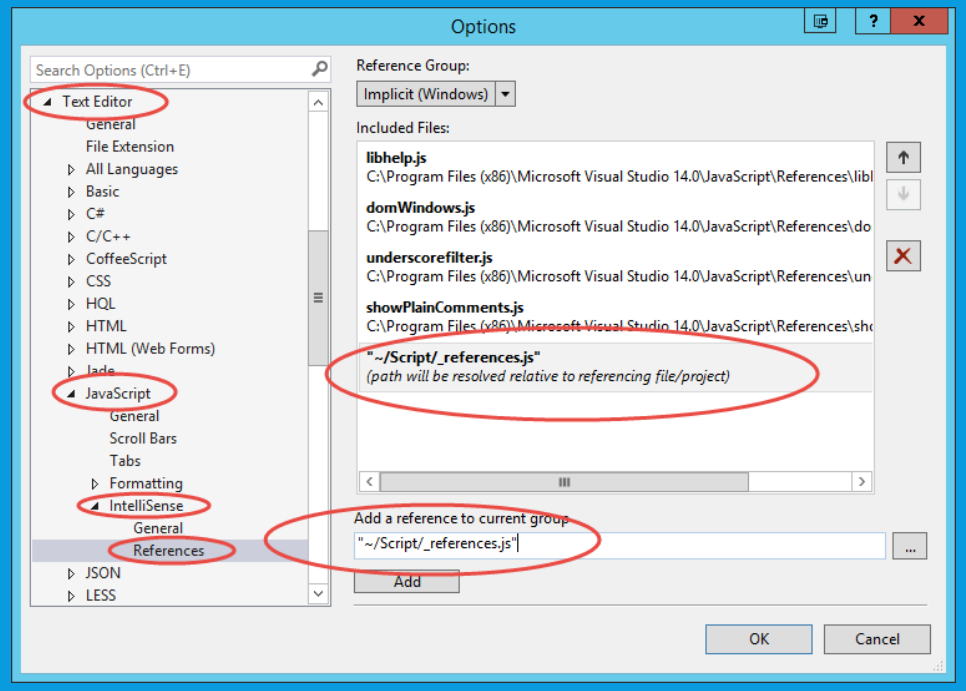
```
<SharePoint:ScriptLink ID="ondemand" Name="ondemand.js" runat="server" OnDemand="True">
```

```
LoadSodByKey("ondemand.js", function () {
    console.log("script loaded");
    delayedLog();
});
```

JavaScript Intellisense

- In order to have proper JavaScript Intellisense create a "_references.js" file
- Create local copies of the files you want – ie sp.debug.js
- /// <reference path="SP.debug.js" />

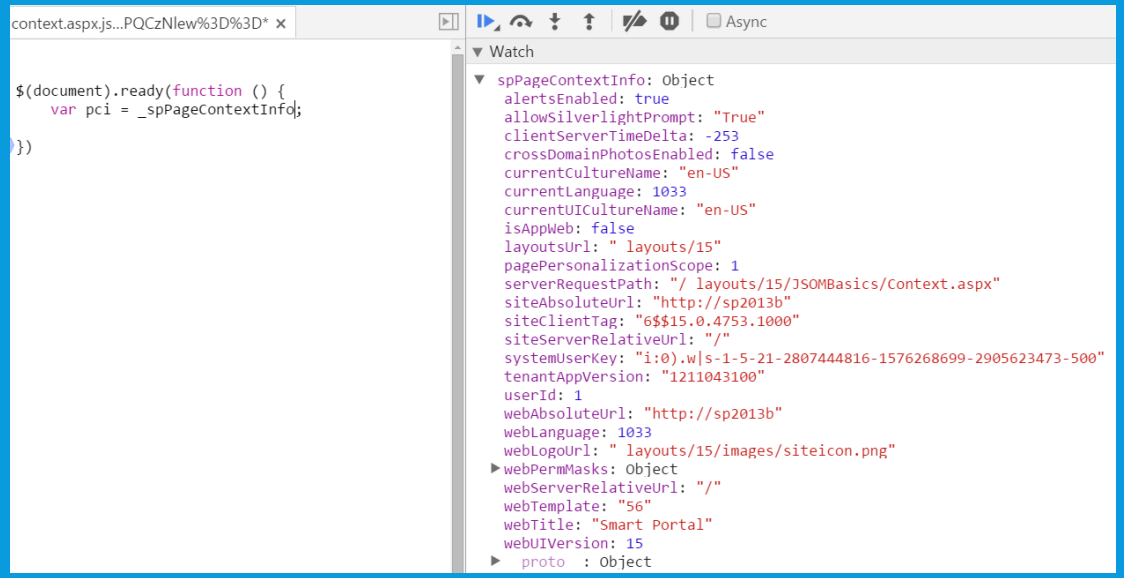




SharePoint Variables & Refs

_spPageContextInfo

- Available in every SharePoint Page
- _spPageContextInfoCulture/Locale information
  - Server-relative URL for site
  - Absolute URL for Site
  - Current page relative URL
  - Pages ListID
  - Web Title
  - Web UI Version



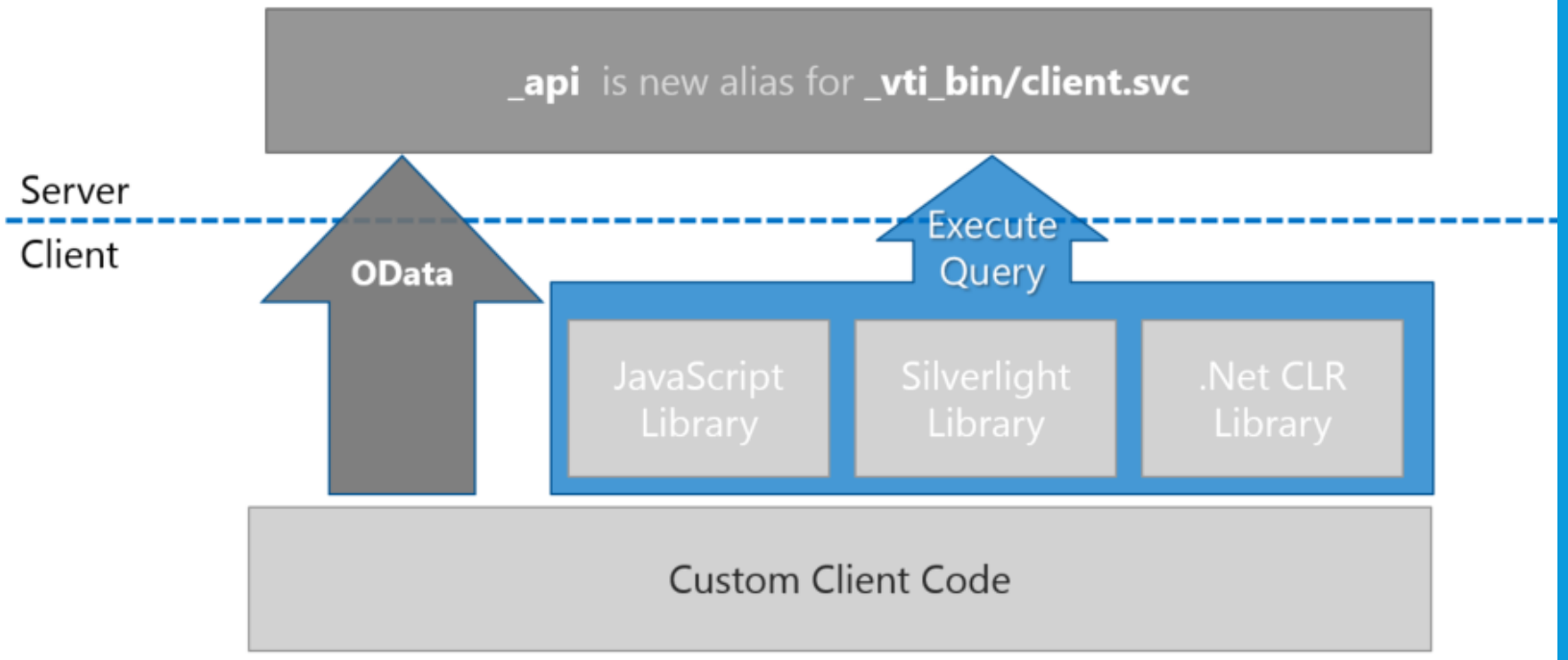
ctx in Global Namespace

- Available in List-Related Pages
- ListDataListViewsonly
  - Row collection property:Column values
  - Row is 0-based, based on display, not item ID
  - ContentTypeId
  - FSObjType(0=ListItem, 1=Folder)


```
▼ Watch
▼ ctx: ContextInfo
    AllowCreateFolder: true
    AllowGridMode: true
  ▶ BasePermissions: Object
    BaseViewID: 1
    CascadeDeleteWarningMessage: null
    ContentTypesEnabled: false
    ControlMode: 4
    CurrentCultureName: "en-US"
    CurrentLanguage: 1033
    CurrentSelectedItems: null
    CurrentUICultureName: "en-US"
    CurrentUserId: 1
    CurrentUserIsSiteAdmin: true
    EnableMinorVersions: false
    ExternalDataList: false
    HasRelatedCascadeLists: 0
    HttpPath: "http://sp2013b/_vti_bin/owssvr.dll?CS=65001"
    HttpRoot: "http://sp2013b"
    IsAppWeb: false
    IsClientRendering: true
    LastSelectableRowIdx: null
  ▶ ListData: Object
    ListDataJSONItemsKey: "Row"
  ▶ ListSchema: Object
    ListTemplateType: 101
    ListTitle: "Documents"
    ModerationStatus: 0
    NavigateForFormsPages: true
    OfficialFileName: ""
```

Context, Batching, Loading

SharePoint 2013 Remote API



ClientContext

- Represents the context for objects and operations
- Reference the required libraries
- Avoid ctx as a variable
- Limited to current site colletion without CDL

```
<script
 type="text/javascript"
 src="//ajax.aspnetcdn.com/ajax/4.0/1/MicrosoftAjax.js">
</script>
<script type="text/javascript" src="_layouts/15/sp.runtime.js"></script>
<script type="text/javascript" src="_layouts/15/sp.js"></script>
<script type="text/javascript">
 // Continue your program flow here.
</script>
```

SP.ClientContext object

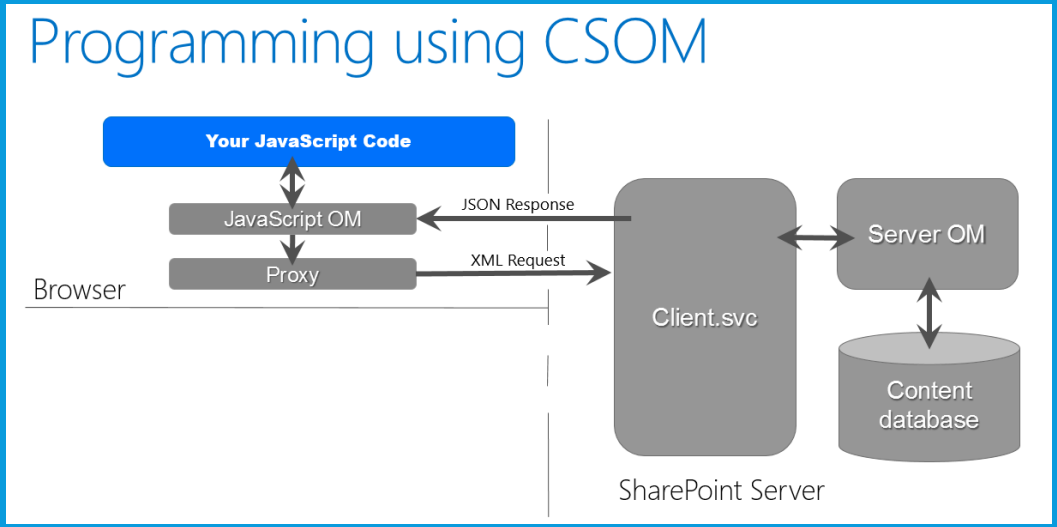
- Initializes a new instance of the ClientContext object for the specified SharePoint site
- Avoid calling it ctx because of ctx in ListViews
- Overloaded constructor
- Cannot span multible Site Collections
- Copy between two ClientContext objects possible

```
var cctx;
cctx = new SP.ClientContext();                        //Connect to the default context
cctx = new SP.ClientContext("http://sp2013c/8085"); //Connect to a specific Url
cctx = new SP.ClientContext("/sales");                //Connect to a subweb
```

Batching

- Commands are sent to the server in batches

```
var cctx = new SP.ClientContext();
this.web = cctx.get_web();
cctx.load(web, 'Title', 'Created');
cctx.load(lists);
cctx.executeQueryAsync(function () {
        console.log("Successfully loaded" + web.get_title());
    }, onErr);
```



Loading

- Objects and their properties must be loaded explicitly
- Three Options:
  - Load All: ctxCurrent.load(web)
  - Load Explicit: ctxCurrent.load(owner, 'CreatedBy', 'Title')
  - Load Collection: ctxCurrent.load(<objectCollection>,'Include(<Field1>,<Field2>,<Field3>'));

Excption Handling

- Exception handling is done using SP.ExceptionHandlingScope

```
function sampleExceptionHandler() {
    var currCtx = new SP.ClientContext();
    var scope=new SP.ExceptionHandlingScope(currCtx);
    var startScope=scope.startScope();
    var tb=scope.startTry();
    //attempt something that causes an error
    var lists=currCtx.get_web().get_lists();
    var badList=lists.getByTitle("badListName");
    tb.dispose();
    var cb=scope.startCatch();
    //*server-side* steps to fix the error
    cb.dispose();
    var fb=scope.startFinally();
    //**server-side actions that will always occur
    fb.dispose();
    startScope.dispose();
    currCtx.executeQueryAsync(onSucceed, onFail);
}
```

Site &Site Collections

SP.Site object

- SP.Site does not allow creating of Site Collections
- SpoOperation class - Represents an operation on a site collection in an Office 365 tenant
- Site Collection operations is not supported by JSOM in an on premise installation
- As a fallback you can use Admin.asmx

SP.Web object

- Represents a Microsoft SharePoint Foundation Web site.
- Main entry point for client side access
- Has most of the properties and methods of the server side equivalent

```
var clientContext = new SP.ClientContext();
this.web = clientContext.get_web();
clientContext.load(web, 'Title', 'Created');
```

Create a Web

- SP.WebCreationInformation() holds the parameter to create a Web

```
var clientContext = new SP.ClientContext.get_current();
var web = clientContext.get_web();
var webCreationInfo = new SP.WebCreationInformation();
webCreationInfo.set_title('My JSOM Web Site');
webCreationInfo.set_description('Description of new Web site...');
webCreationInfo.set_language(1033);
webCreationInfo.set_url('MyJSOMWebSite');
webCreationInfo.set_useSamePermissionsAsParentSite(true);
webCreationInfo.set_webTemplate('STS#0');
web.get_webs().add(webCreationInfo);
web.update();

clientContext.executeQueryAsync(function () { console.log("JSOM Web created"); }, onQueryFailed);
```

Update / Delete Web

- Update

```
var clientContext = new SP.ClientContext.get_current();
var web = clientContext.get_web();
web.set_title('Updated Web Site');
web.set_description('This is an updated Web site.');
web.update();
clientContext.load(web, 'Title', 'Description');
clientContext.executeQueryAsync(function () {
    console.log('Title: ' + web.get_title() + ' Description: ' + web.get_description());
}, onQueryFailed);
```

- Delete

```
var web = site.openWeb("/MyJSOMWebSite");
web.deleteObject();
clientContext.load(site);
clientContext.load(web);
clientContext.executeQueryAsync(function() {
```

Use Properts Bag

- A dictionary of key – value to store custom values for your app
- Write to property bag

```
var clientContext = new SP.ClientContext.get_current();
var web = clientContext.get_web();
this.properties = web.get_allProperties();
this.properties.set_item("myCustomProperty", "myCustomValue");
clientContext.load(web);
web.update();
clientContext.executeQueryAsync(Function.createDelegate(this, getWebProperty), onQueryFailed);
```

- Get value from property bag

```
var val = web.properties.get_item("myCustomProperty");
```

Manage Lists

List Basics

- Lists can be accessed using an instance of the List object
- Items in a list are represented by ListItem object
- To assign a value to a column use
  - ListItem[„Fieldname“] =*Value*
  - ListItem.Update()

Create List

- Create list

```
var clientContext = new SP.ClientContext(siteUrl);
var web = clientContext.get_web();
var listCreationInfo = new SP.ListCreationInformation();
listCreationInfo.set_title(listName);
listCreationInfo.set_templateType(SP.ListTemplateType.announcements);
var list = web.get_lists().add(listCreationInfo);
clientContext.load(list);
clientContext.executeQueryAsync(function () { console.log("Create list done"); }, onQueryFailed);
```

- Find list using a specific template

```
listItemEnumerator = lists.getEnumerator();
var discussionBoardLists = [];
while (listItemEnumerator.moveNext()) {
    oListItem = listItemEnumerator.get_current();
    if (oListItem.get_baseTemplate() == 108) {
        var listname = oListItem.get_title();
        discussionBoardLists.push(listname);
    }
}
```

Update / Delete List

- Update

```
var list = clientContext.get_web().get_lists().getByTitle("News");
list.set_description("The very cool Announcments list");
list.update();
clientContext.load(list);
clientContext.executeQueryAsync(function () { console.log("Update list done"); }, onQueryFailed);
```

- Delete

```
var list = web.get_lists().getByTitle("News");
list.deleteObject();
clientContext.executeQueryAsync(function () { console.log("Delete list done"); }, onQueryFailed);
```

Manipulate List

- Add field to list

```
var list = clientContext.get_web().get_lists().getByTitle(listName);
this.fld = list.get_fields().addFieldAsXml('<Field DisplayName=\'MyField\' Type=\'Number\' />', true,
SP.AddFieldOptions.defaultValue);
var fieldNumber = clientContext.castTo(fld, SP.FieldNumber);
fieldNumber.set_maximumValue(100);
fieldNumber.set_minimumValue(35);
fieldNumber.update();
clientContext.load(fld);
clientContext.executeQueryAsync(function () { console.log("Add field to list done"); }, onQueryFailed);
```

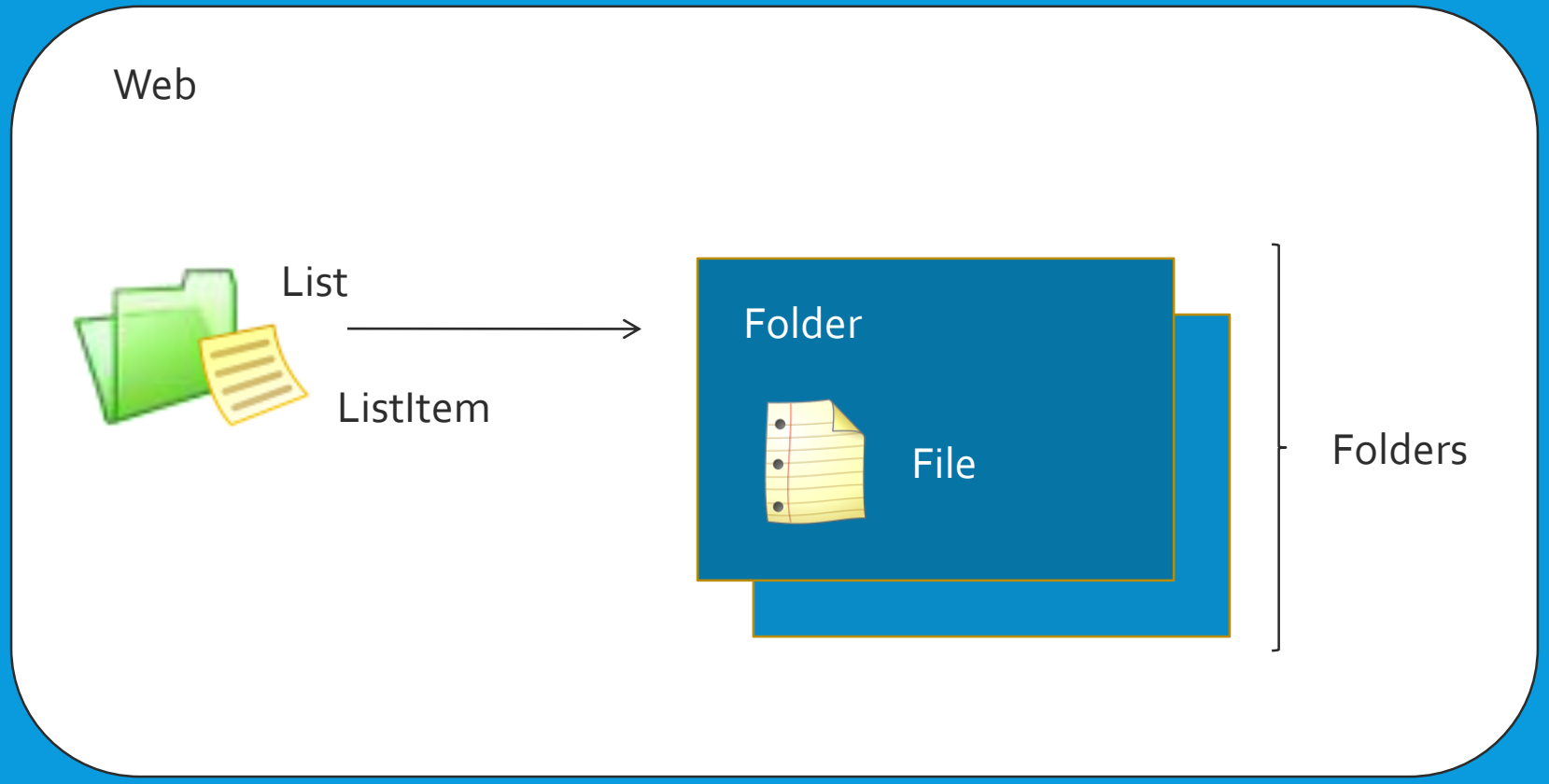
- Set choice values

```
fields = web.get_fields();
var fieldExpiryDate = context.castTo(fields.getInternalNameOrTitle("ProductionType"),
SP.FieldChoice);
var choices = Array("Phase 1 Trial", "Phase 2 Trial", "Phase 3 Trial", "Production");
fieldExpiryDate.set_choices(choices);
context.ExecuteQueryAsync(onUpdateFieldSuccess, onUpdateFieldFail);
```

Document Library Basics

- Document Libraries are instances of the SPList Class
- Items in a Document Library are instances of the SPListItem Class
- Each Library has an associated SPFolder in which the files (SPFile) of the list are stored

Document Storage in SharePoint



Working with files

- Create

```
list = web.get_lists().getByTitle("Shared Documents");
fci = new SP.FileCreationInformation();
fci.set_url("my new file.txt");
fci.set_content(new SP.Base64EncodedByteArray());
fileContent = "The content of my new file";
for (var i = 0; i < fileContent.length; i++) {

        fci.get_content().append(fileContent.charCodeAt(i));
}
var newFile = list.get_rootFolder().get_files().add(fci);
clientContext.load(this.newFile);
```

ListItem CRUD

Simple Read

- List item is represented by SP.ListItem
- "Title" is represented displayName
- Access to properties is done using "get_PROPERTY()"

```
var clientContext = new SP.ClientContext();
var list = clientContext.get_web().get_lists().getByTitle('News');
var li = list.getItemById(itemId);
clientContext.executeQueryAsync(function() {
    console.log(li.get_displayName());
});
```

CAML Query

- Used to search for one or more item
- <ViewFields> corresponds to "Select xx from"
- <Where> corresponds to condition

```
var camlQuery = new SP.CamlQuery();
camlQuery.set_viewXml('<Query><Where><Eq><FieldRef Name="ID" /><Value Type="Counter">'
         + itemId + '1</Value></Eq></Where></Query><View><RowLimit>100</RowLimit></View>');
var collListItem = list.getItems(camlQuery);
clientContext.load(collListItem, 'Include(Id, DisplayName, HasUniqueRoleAssignments)');
clientContext.executeQueryAsync(
            function() {
                var liInfo = '';
                var listItemEnumerator = collListItem.getEnumerator();
                while (listItemEnumerator.moveNext()) {
                    var li = listItemEnumerator.get_current();
                    …
                }
```

Simple Create

- XXCreationInformation classes are used to set the param sent to server

```
var clientContext = new SP.ClientContext();
var list = clientContext.get_web().get_lists().getByTitle('News');
var itemCreateInfo = new SP.ListItemCreationInformation();
this.li = list.addItem(itemCreateInfo);
li.set_item('Title', 'My New Item!');
li.set_item('Body', 'Hello World!');
li.update();

clientContext.load(li);
clientContext.executeQueryAsync(function () {
            itemId = li.get_id();
            alert('Item created: ' + itemId);
},function(){..})
```

Complex Field Values

- Complex Field Values have to be set using objects
  - FieldGeoLocationValue
  - FieldLookupValue
  - FieldUrlValue
  - FieldUserValue

```
var urlValue = new SP.FieldUrlValue();
urlValue.set_url("http://www.example.com");
urlValue.set_description("test link");
myItem.set_item("TestURL", urlValue);
```

Lookups

- Call fld.get_lookupValue() the get the Value
- Use SP.FieldLookupValue() and pass unique ID of the lookup item to write

```
var ctx = SP.ClientContext.get_current();
var web = ctx.get_web();
var lists = web.get_lists();
var listNews = lists.getByTitle("News");
var firstNews = listNews.getItemById(1);
ctx.load(firstNews);
ctx.executeQueryAsync(function() {
    var lookupField = firstNews.get_item("Writer");
    var lookupTitle = lookupField.get_lookupValue();
    console.log("title of lookedup field is " + lookupTitle);
}, onQueryFailed);
```

Managed Metadata

- Implemented in sp.taxonomy.js
- Documentation @ https://msdn.microsoft.com/en-us/library/office/jj857114.aspx

```
var ctx = SP.ClientContext.get_current();
   var web = ctx.get_web();
   var lists = web.get_lists();
   var listNews = lists.getByTitle("News");
   var firstNews = listNews.getItemById(1);
   ctx.load(firstNews);
   ctx.executeQueryAsync(function () {
       var mmField = firstNews.get_item("Topic");
       console.log("title of Managed Metadata field is " + mmField.Label);
   }, onQueryFailed);
```