Extend Dynamics 365 - Deploy Typescript based Web Application in Web Resources

Cognitive Convergence
http://www.cognitiveconvergence.com
+1 4242530744
shahzad@cognitiveconvergence.com

Cognitive Convergence is Subject Matter Expert in Office 365, Dynamics 365, SharePoint, Project Server, Power Platform: Power Apps-Power BI-Power Automate-Power Virtual Agents. Our Microsoft Office 365 Consulting, add-in Development, Customization, Integration services and solutions, can help companies maximize business performance, overcoming market challenges, achieving profitability, and providing best customer service.

Contents

OBJECTIVE

This case study is written to give a brief introduction about development of Dynamics 365 using Typescript, node.js and other related technologies for front-end and back-end to create a full fledge web application. This pattern of development is used to extend Dynamics 365.

INTRODUCTION

Typescript is a way of writing code in the next generation of JavaScript before it is fully supported by all browsers. TypeScript is a superset of JavaScript. JavaScript is TypeScript without all the new features and strict compiler checks. We can convert your JavaScript into TypeScript with very little effort and then start to make use of the new features available in ES6 gradually over time.

TypeScript has several benefits including easier OOP methodology, common errors spotting at compile time, support for new ECMAScript standards, etc. By default, Dynamics 465 does not support TypeScript, but we can implement it through different methods. We'll see later how it works.

Typescript code is deployed as a website in web resources and development is done using Data verse entities.

BENEFITS AND VALUE PROPOSITIONS

If we develop through typescript, following capabilities are provided for Dynamics 365:

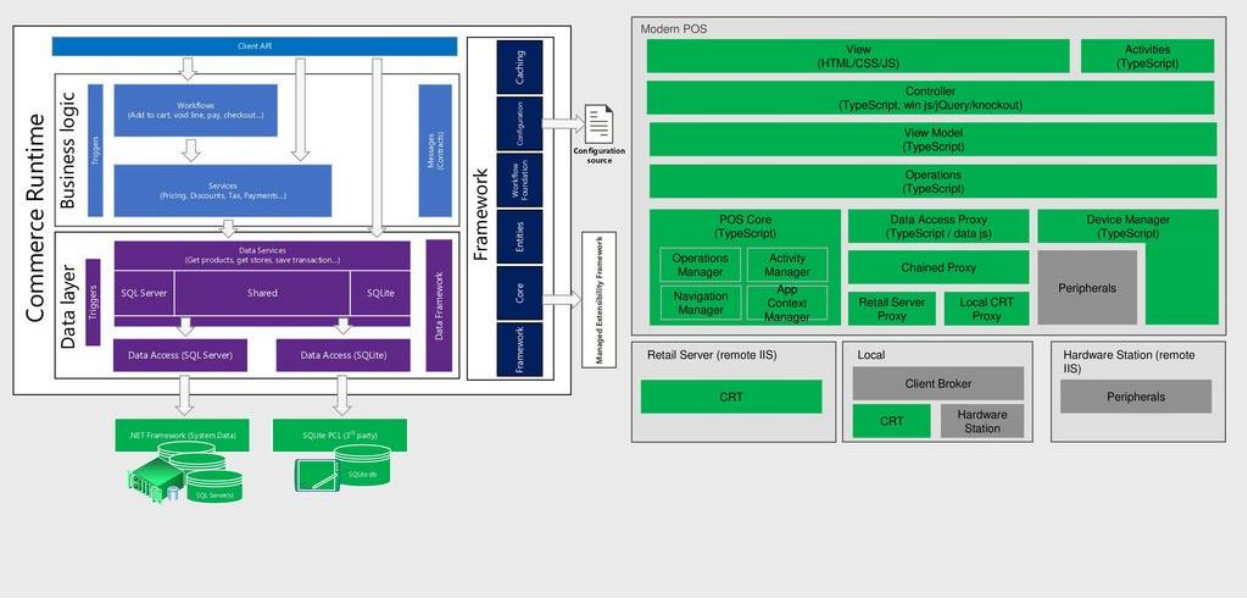
- **Open Source** - TypeScript is a free and open source programming language
- **Object-Oriented Programming** - adds optional static typing and class-based object-oriented programming to the JavaScript language
- **Large Applications** - TypeScript is designed for development of large applications and trans compiles to JavaScript.
- **Pure JavaScript Files** - Typescript creates pure JavaScript files so that the result is the same as if you wrote JavaScript.
- **Robust Software** - A large JavaScript project, adopting Typescript might result in more robust software, while still being deployable where a regular JavaScript application would run.
- **Error Detection** - All the type errors are found during runtime.

- **TypeScript is portable** - TypeScript is portable across browsers, devices, and operating systems. It can run on any environment that JavaScript runs on. Unlike its counterparts, TypeScript does not need a dedicated VM or a specific runtime environment to execute.
- **Dynamic Process of adding code in web resource -** Writing the website code in Visual Studio or any other compiler helps developers to access Dynamics 365 entities/tables outside the environment and embedding in web resources make it a dynamic process. It can be added in any solution's web resource providing the access credentials.

SYSTEM REQUIREMENTES FOR DYNAMICS 365 TYPESCRIPT BASED DEVELOPMENT

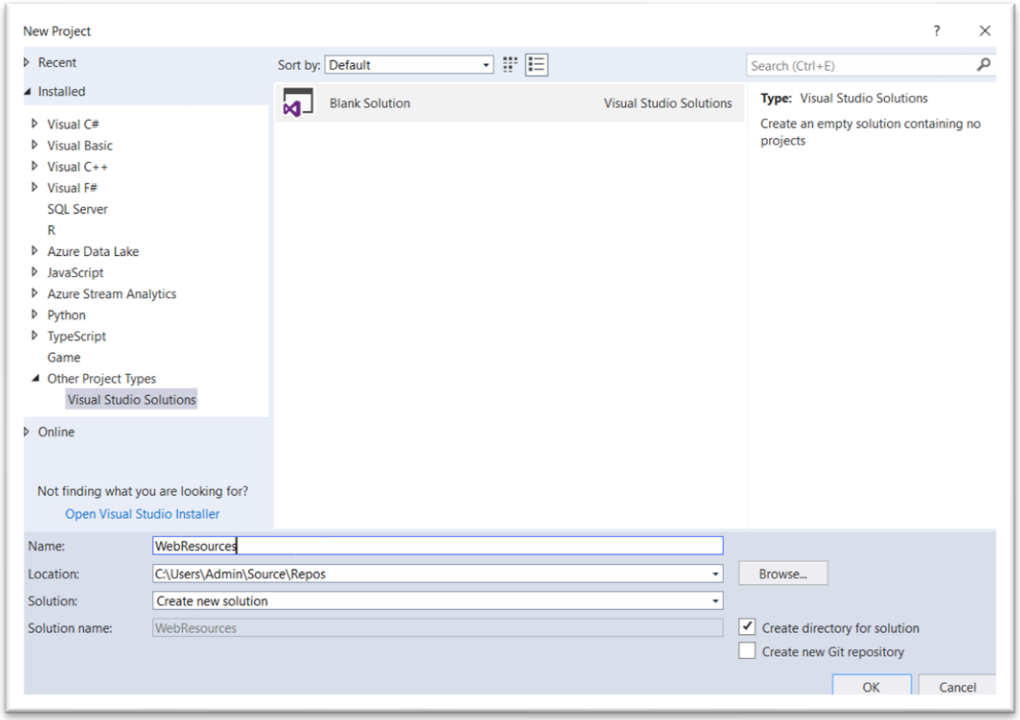
This development pattern has some system requirements based on which the development can be done.

- **Dynamics 365** - Dynamics 365 (online)
- **Code Editor** – Visual Studio 2017 (Preferred)
- **Node.js –** Latest version of node available

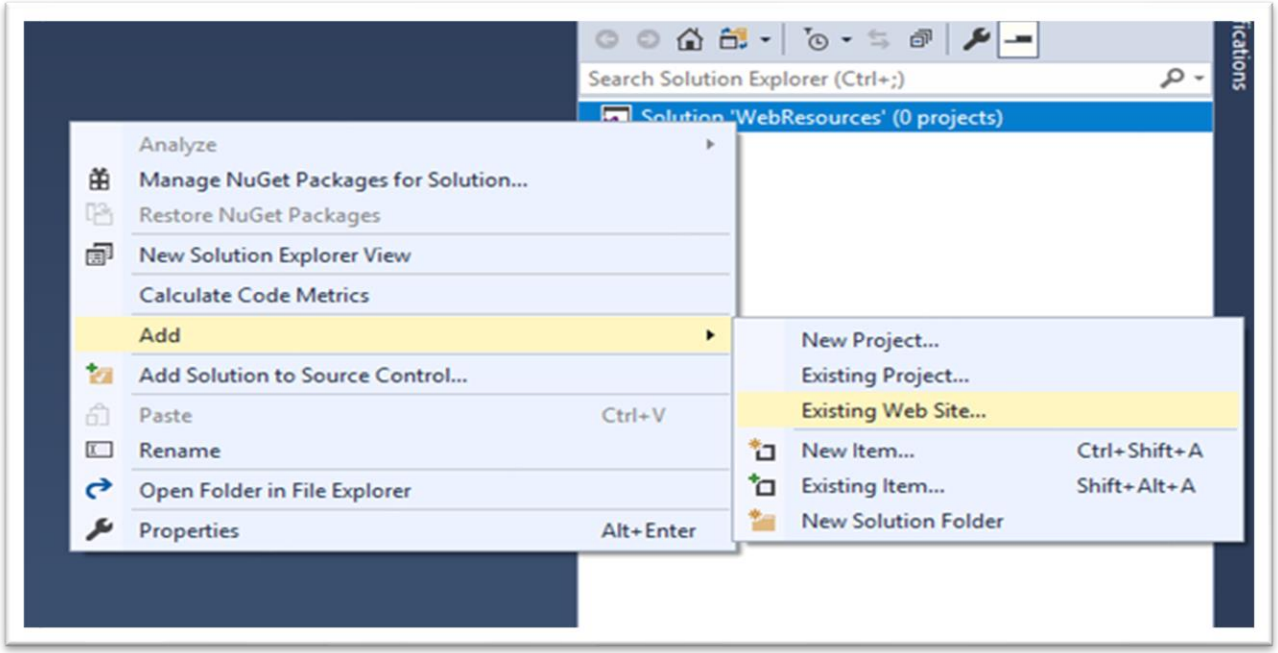


CREATING TYPESCRIPT ENVIRONMENT

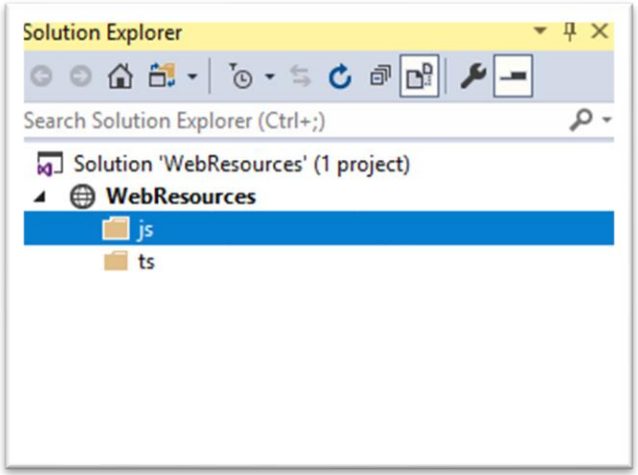
There are plenty of approaches to set up the environment for typescript development and start writing web resource for dynamics 365. We start from blank solution of Visual Studio.



In our case, we create a new folder “WebResouces” in C:\Users\Admin\Source\Repos\WebResources\



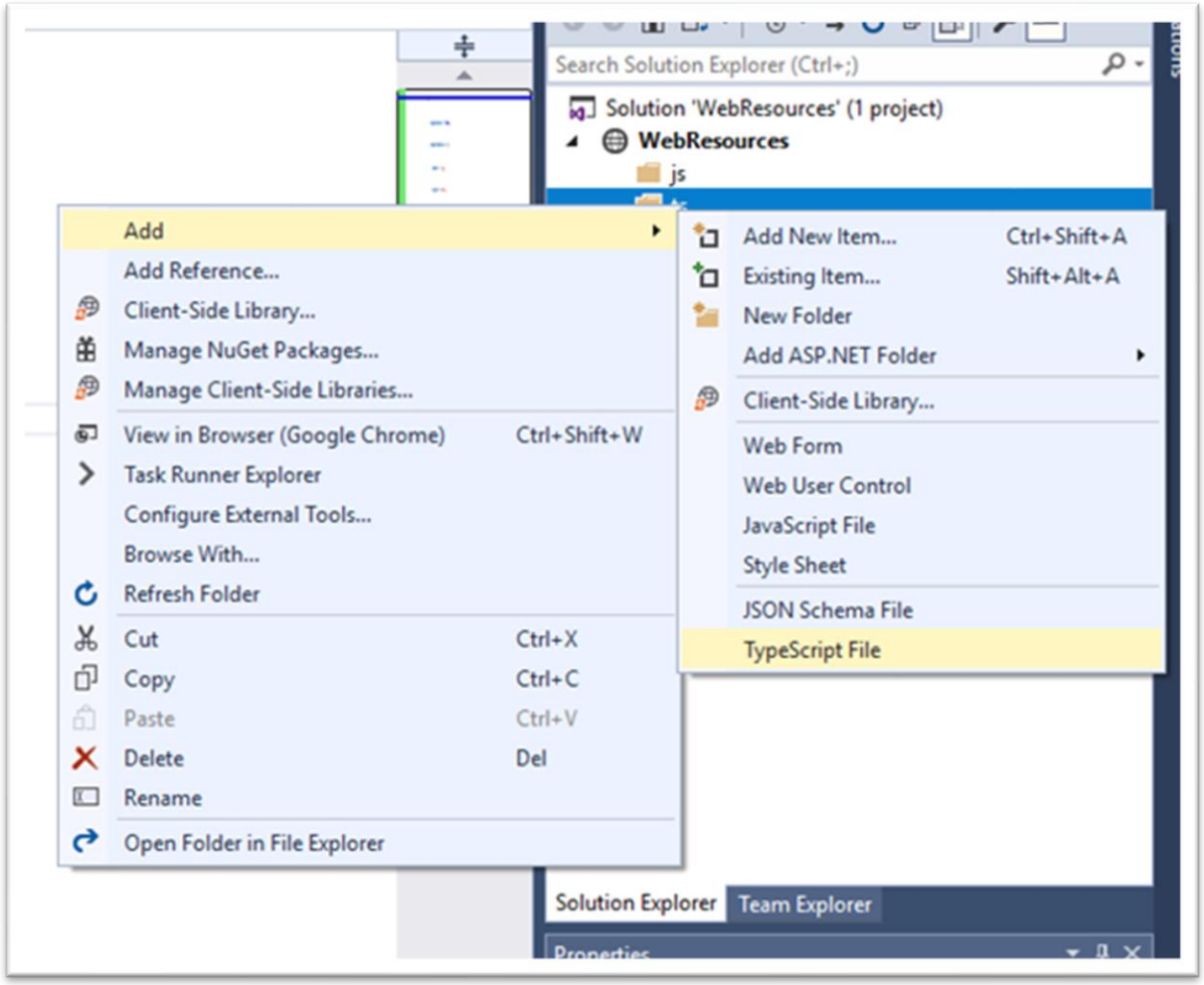
WebResources project has the folders where we will store our TypeScript and compiled JavaScript files.



In tsconfig.json file,

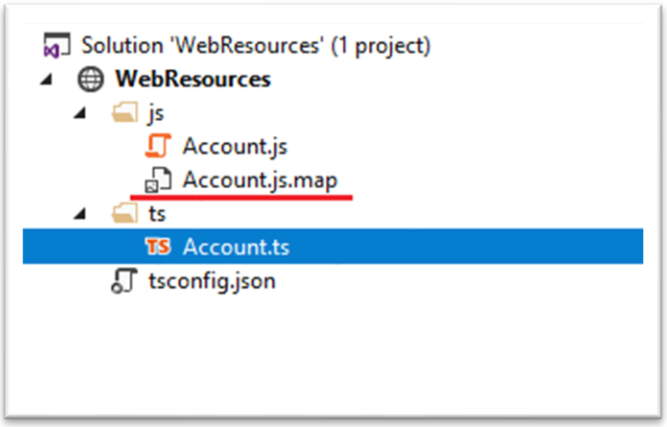
```
{
  "compileOnSave": true,
  "compilerOptions": {
    "outDir": "js",
    "target": "es5",
    "sourceMap": true
  },
  "include": [
    "./ts/**/*"
  ]
}
```

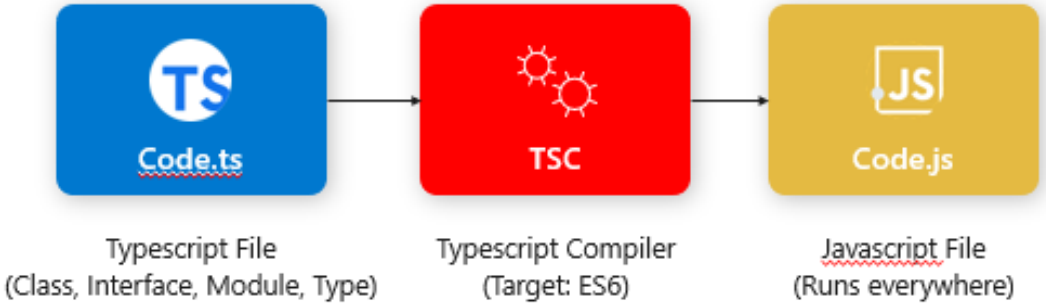
The “Account.ts” file has all the TypeScript code.



Typescript file mapped to JavaScript

The Account.ts file it automatically generates JavaScript files under the js folder. This is controlled by compilation options in the tsconfig.json file.





Adding reference

A reference to the index.d.ts file in the Account.ts file is added.

```
/// <reference path="../lib/@types/xrm/index.d.ts" />
namespace Account {
    export function getClientUrl() {
        var url = Xrm.Page.context.getClientUrl();
        alert(url);
    }
}
```

There can be several patterns for the UI development, and we can add other components in the project too.

COMPLETE WEB APPLICATION

Dynamics 365 web resources mainly support HTML, CSS and JavaScript language. The Typescript code is deployed as JavaScript once the tsconfig.json file configures the typescript code and convert it into JavaScript.

There can be multiple JavaScript, HTML and CSS files in web resources that are related to each other through URLs. The HTML file may contain the link to CSS and JavaScript files in style and script tags, respectively. For example,

```
<link rel="stylesheet" type="text/css" href="../styles/styles.css" />
```

```
<script src="Script/Script.js" type="text/javascript"></script>
```

The UI can be developed using various supported client-side UI Frameworks. These frameworks make the User Interface friendly and easy to use.

Adding web pages in web resources:

The created web pages are added in web resources of dynamics 365 solution during bundling and packaging of the solution. The naming convention is done with a "/" in their names which is used to identify, differentiate, and categorize the web pages. The following sample shows the style of URL you can use to view web resources.

<Microsoft CRM URL>/WebResources/<name of web resource>

For example,

https://MyOrganization.crm.dynamics.com/WebResources/new_/test/test.html

Adding images in web resources:

Web resources are virtual files in multiple formats, such as html files, JavaScript, and Silverlight applications, that are stored in the system and can be retrieved by using a unique URL address. There are some formats which are supported by Dynamics 365 in web resources. The supported image resources are of type PNG, JPG, GIF and ICO. Images can be included to be used in CSS files or stylesheets in the web application.

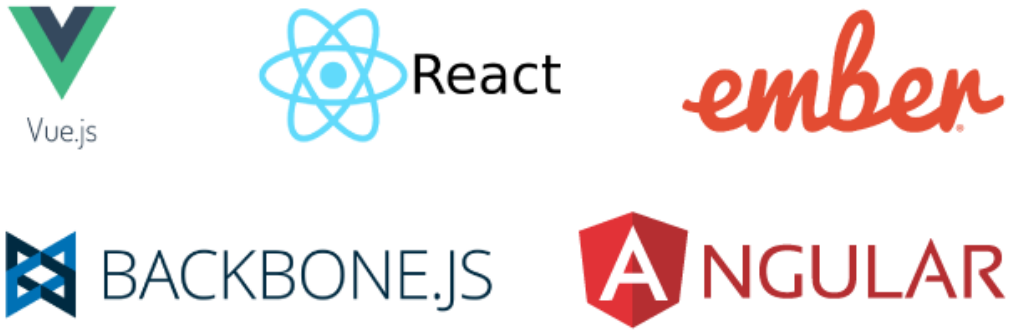
Supported Web Resources:

All the formats that are supported in web resources are listed below:

File	File extensions
Webpage (HTML)	.htm, .html
Style Sheet (CSS)	.css
Script (JScript)	.js
Data (XML)	.xml
Image (PNG)	.png
Image (JPG)	.jpg
Image (GIF)	.gif
Silverlight (XAP)	.xap
StyleSheet (XSL)	.xsl, .xslt
Image (ICO)	.ico
Vector format (SVG)	.svg
String (RESX)	.resx

CLIENT-SIDE UI FRAMEWORKS

Client-side web parts are client-side components that run in the context of a Dynamics 365 Solution Web resources that will be deployed using this pattern. Modern JavaScript web frameworks, tools, and libraries can be used to build them.



Some of the modern front-end frameworks that can be used are:

React.js

React is an open-source JavaScript library for building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications. React makes it painless to create interactive UIs. Design simple views for each state in the application and React will efficiently update and render just the right components when the data changes. Build encapsulated components that manage their state, then compose them to make complex UIs.

Angular.js

AngularJS is a JavaScript-based open-source front-end web framework mainly maintained by Google and by a community of individuals and corporations to address many of the challenges encountered in developing single-page applications. Angular is unique with its two-way data binding feature. It means there is a real-time synchronization between the model and the view, where any change in the model reflects instantly on the view and vice versa.

Vue.JS

Vue.js is an open-source model–view–viewmodel JavaScript framework for building user interfaces and single-page applications. It was created by Evan You and is maintained by him and the rest of the active core team members coming from various companies such as Netlify and Netguru. It is a progressive JavaScript framework used to develop interactive web interfaces. Focus is more on the view part, which is the front end. It is very easy to integrate with other projects and libraries. The installation of Vue.JS is fairly simple, and beginners can easily understand and start building their user interfaces. The content is divided into various chapters that contain related topics with simple and useful examples.

Ember.JS

mber.js is an open-source, free JavaScript client-side framework used for developing web applications. It allows building client-side JavaScript applications by providing a complete solution that contains data management and application flow.

The original name of Ember.js was the *SproutCore MVC framework*. It was developed by *Yehuda Katz* and was initially released *in December 2011*. The stable release of Ember.js is 2.10.0 and this was released on November 28, 201. Ember.js uses templates that help to automatically update the model if the content of applications gets changed

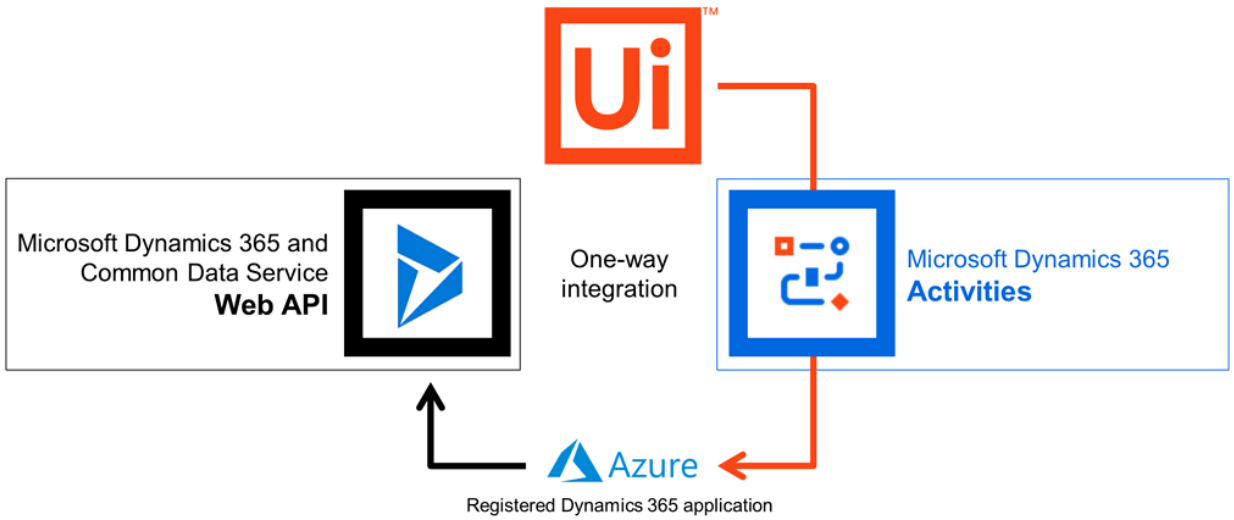
Backbone.js

One of the easiest frameworks out there, Backbone.js allows us to develop single-page applications swiftly. It is a framework which is based on MVC architecture. Like a Controller, MVC architecture's View allows the implementation of component logic. The platform also allows us to develop projects that require different types of users, where the arrays can be used to distinguish the models. So, regardless of whether you intend to use Backbone.js for frontend or backend, this is an ideal choice as its REST API compatibility provides seamless synchronization between the two.

ACCESS DYNAMICS 365 ENTITIES/TABLES IN TYPESCRIPT

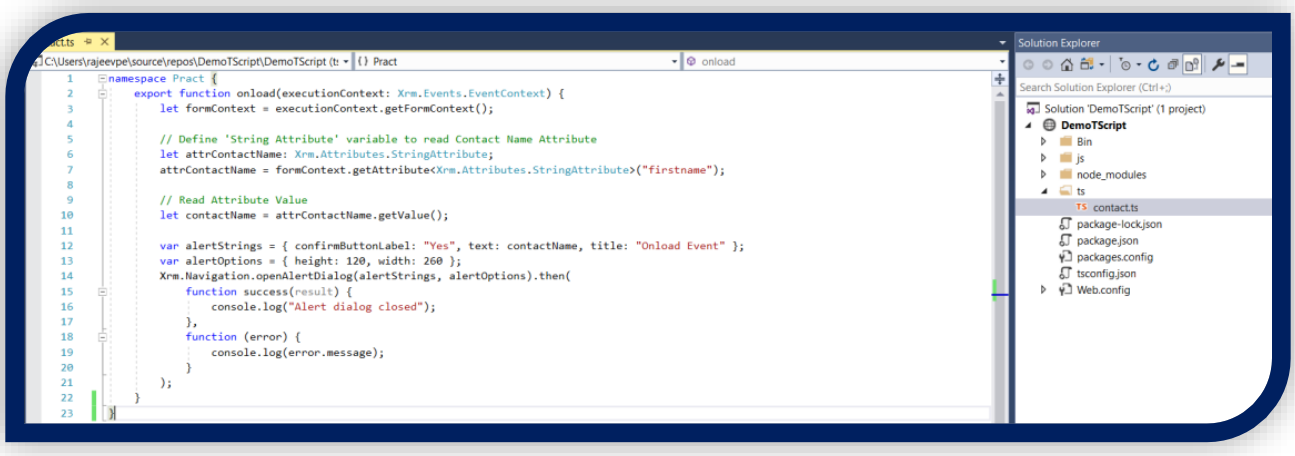
There are 2 methods using which the dynamics 365 entities and form attributes can be accessed and actions can be performed.

- Form context
- XRM Web API



Form Context

The Client API form context (formContext) provides a reference to the form or to an item on the form, such as, a quick view control or a row in an editable grid, against which the current code is executed.




```
namespace Pract {
    export function onload(executionContext: Xrm.Events.EventContext) {
        let formContext = executionContext.getFormContext();

        // Define 'String Attribute' variable to read Contact Name Attribute
        let attrContactName: Xrm.Attributes.StringAttribute;
        attrContactName = formContext.getAttribute<Xrm.Attributes.StringAttribute>("firstname");

        // Read Attribute Value
        let contactName = attrContactName.getValue();

        var alertStrings = { confirmButtonLabel: "Yes", text: contactName, title: "Onload Event" };
        var alertOptions = { height: 120, width: 260 };
        Xrm.Navigation.openAlertDialog(alertStrings, alertOptions).then(
            function success(result) {
                console.log("Alert dialog closed");
            },
            function (error) {
                console.log(error.message);
            }
        );
    }
}
```

This typescript code is using execution context's getFormContext to get attributes of an entity.

XRM Web API

XRM Web API provides properties and methods to use Web API to create and manage records and execute Web API actions and functions in model-driven apps. This Web API provides the following methods:

- createRecord
- retrieveRecord
- retrieveMultipleRecords
- deleteRecord
- UpdateRecord
- isAvailableOffline
- execute
- executeMultiple

Perform CRUD using XRM Web API

Create Record:

```
function createRecord(executionContext){
    var dataAccount=
    {
        "name":"Faiqa",
        "msdyn_travelcharge":20,
    }

    Xrm.WebApi.createRecord("account",dataAccount).then(
        function success(result)
        {
            alert("Record created"+ result.id);
        },
        function (error){
            alert(error.message);
        }
    );
}
```

Delete Record:

Deletes record with record ID "0bccc980-ae0e-eb11-a813-000d3a9c7d8e"

```
function deleteRecord()
{
    Xrm.WebApi.deleteRecord("account", "0bccc980-ae0e-eb11-a813-000d3a9c7d8e").then(
        function success(result)
        {
            alert("Record deleted "+ result.name);
        },
        function (error){
            alert(error.message);
        }
    );
}
```

Retrieve Record:

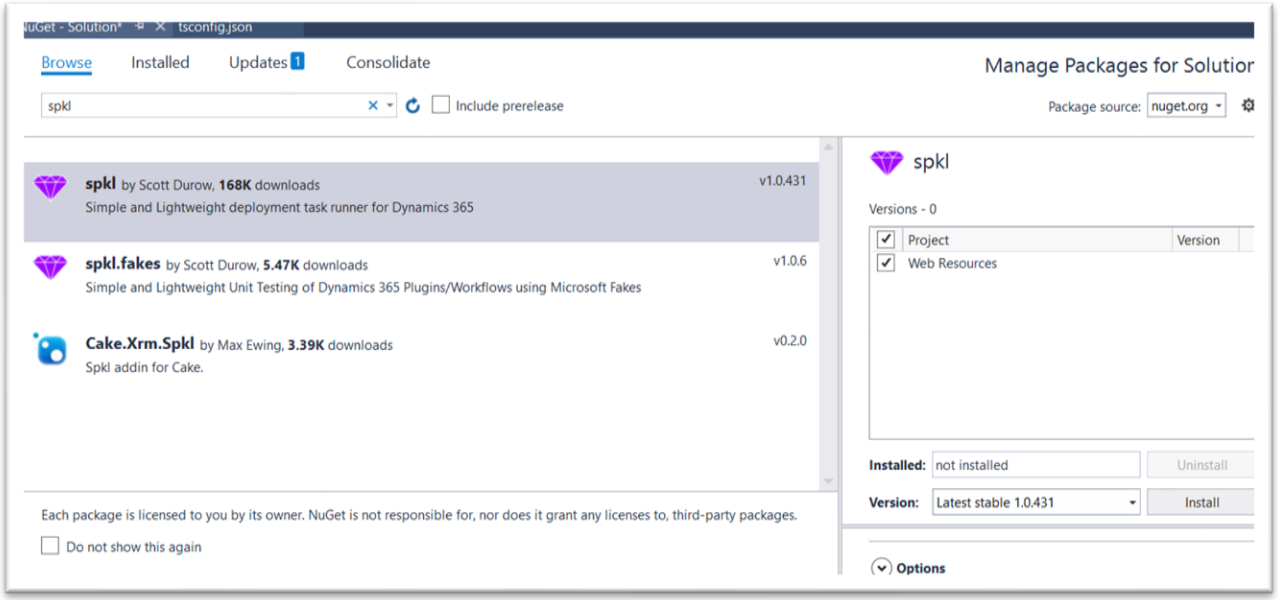
```
function retrieveRecord()
{
    Xrm.WebApi.retrieveRecord("account", "3ab0b0cb-ae0e-eb11-a813-000d3a9c7d8e", "?$select=name,telephone1").then(
        function success(result) {
            alert("Retrieved values: Name: " + result.name + ", Telephone: " + result.telephone1);
            // perform operations on record retrieval
        },
        function (error) {
            alert(error.message);
            // handle error conditions
        }
    );
}
```

Update Record:

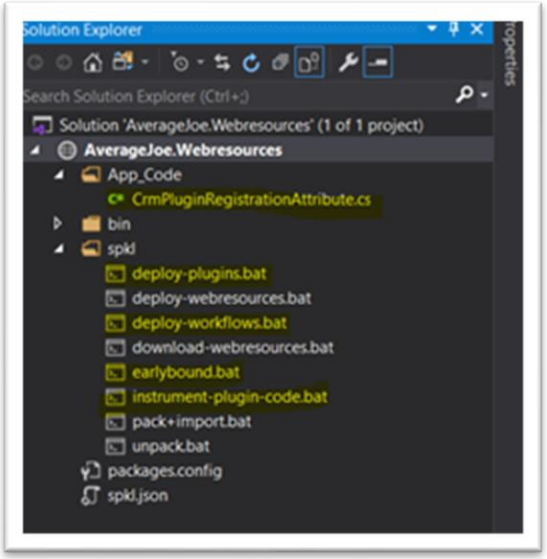
```
function updateRecord()
{
    var data =
    {
        "name": "Updated Sample Account ",
        "address1_city": "Sahiwal",
        "description": "This is the updated description of the sample account",
    }
// update the record
    Xrm.WebApi.updateRecord("account", "65badb47-b20e-eb11-a813-000d3a9c7d8e", data).then(
    function success(result) {
        alert("Account updated");
        // perform operations on record update
    },
    function (error) {
        alert(error.message);
        // handle error conditions
    }
);
}
```

DEPLOYING TYPESCRIPT IN DYNAMICS 365 WEB RESOURCES

There are different ways to deploy the typescript file in the web resources through visual studio. One of them is using spkl NuGet package. By installing and using this package, we can deploy our developed solution along with all the UI, Typescript, JavaScript, lib files that we created and added in the solution.



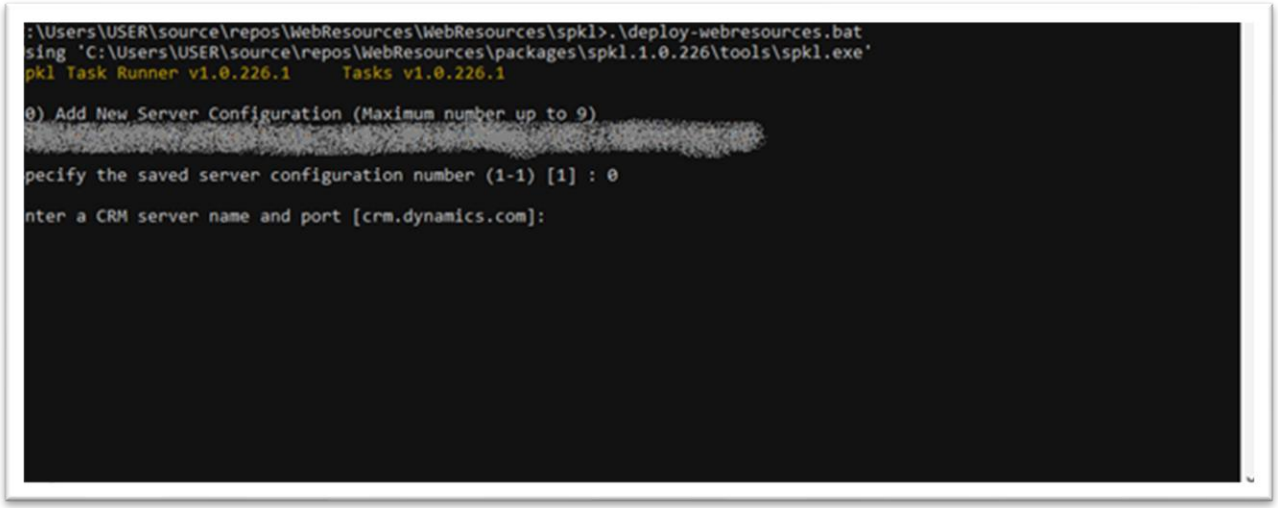
After the installation, several folders are added to the solution. *spkl* is not only web resource deployment tool. It has some additional features like plugin deployment, early bound class generations, etc.,



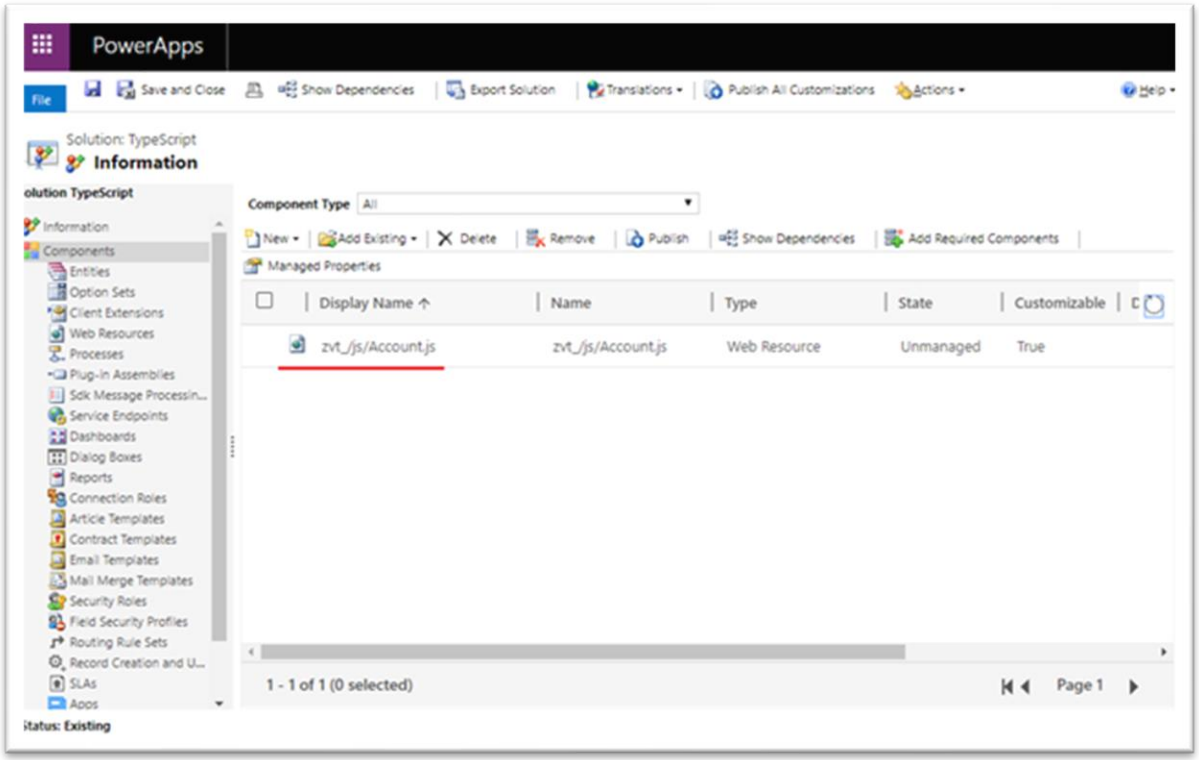
In spkl.json file, we provide solution name and other required things.

```
{
    "webresources": [
      {
        /*
            Option - profile - Provide a comma delimitered list of profile names that can be referenced
when calling spkl
            */
        "profile": "default,debug",

        /*
            Optional - root - Provide the relatative path of the webresources.
            */
        "root": "",

        /*
            Optional - solution - Add webresources to a solution when deploying
            */
        "solution": "TypeScript",

        /*
            Required - files - List the webresources to deploy relatative to the root of this file (or
the the root parameter above)
            */
        "files": [
          {
            "uniquename": "zvt_/js/Account.js",
            "file": "js\\Account.js",
            "description": ""
          }
        ]
      }
    ]
}
```

The spkl package also added spkl folder with few batch scripts files. We need the one called deploy.webresources.bat. Open the command prompt, navigate to the project root directory, and execute the batch script file.



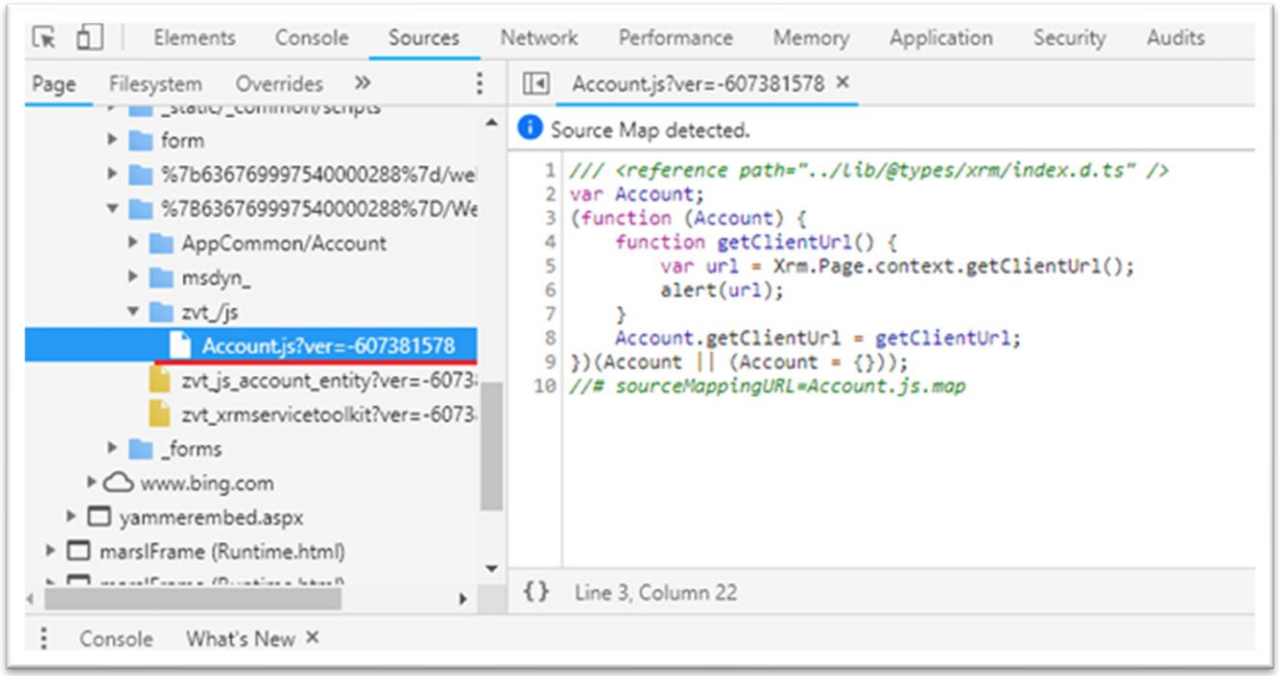
After providing the requirements, web resource will be uploaded to the instance as:



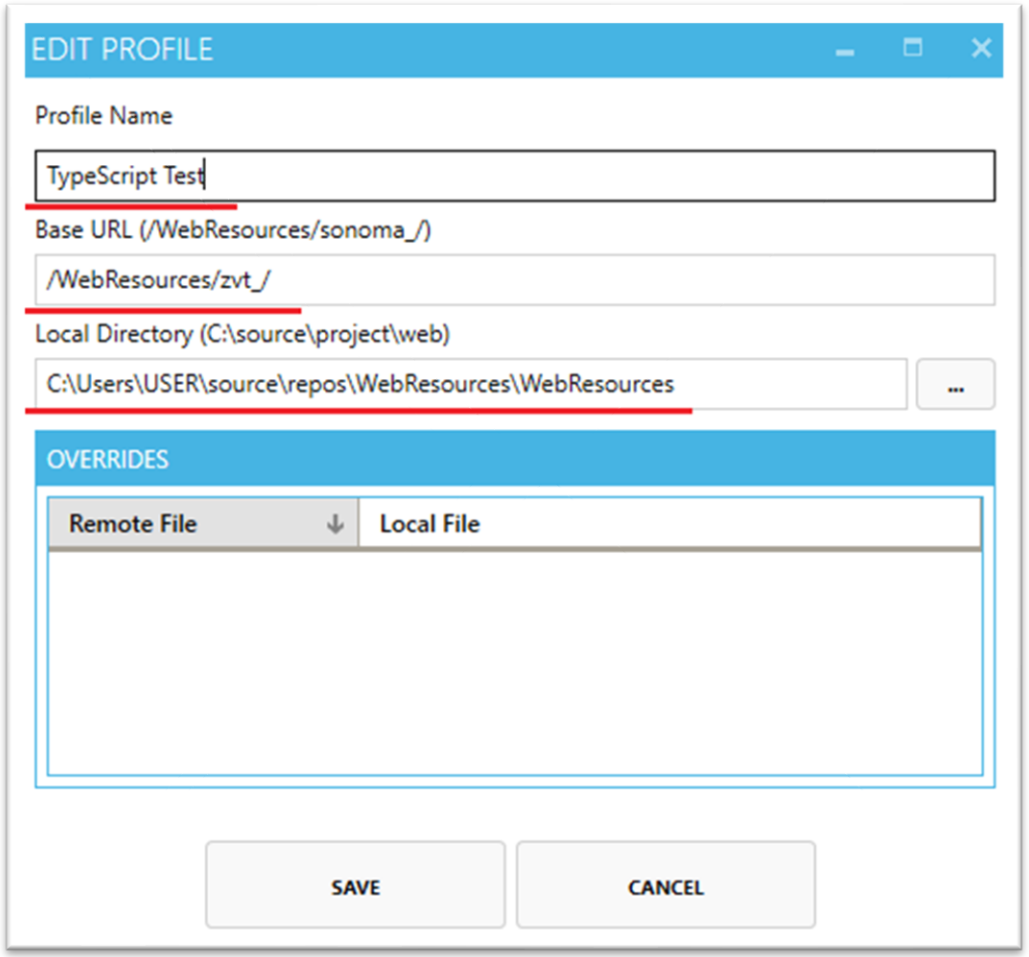
DEBUG TYPESCRIPT WITHIN DYNAMICS 365

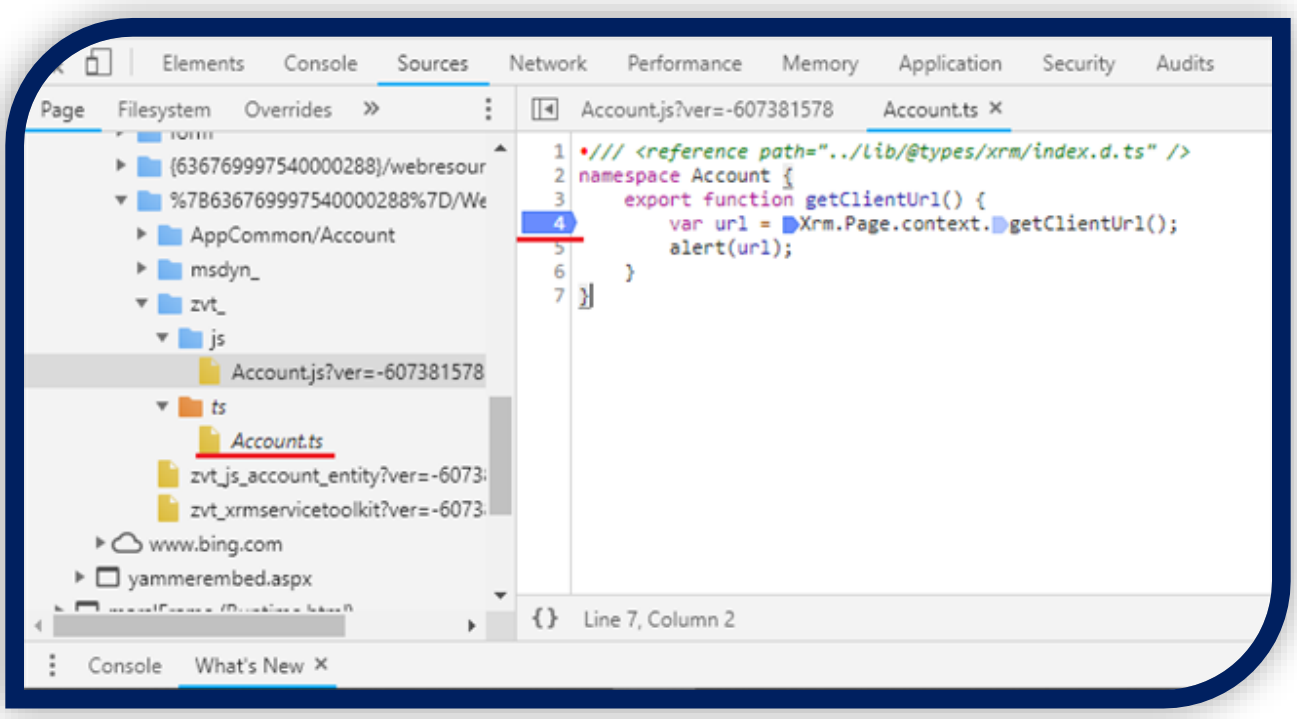
The typescript code is embedded in the dynamics 365 as JavaScript. The problem is that in the Dynamics 365 environment we have only the JavaScript file. We didn't deploy the TypeScript file, so we are not able to access that file via Developer Tools(F12).

We debug the typescript code withing the Dynamics 365.



We use Fiddler extension called imposter:





From here, we can debug your TypeScript code in Developer Tools(F12) or directly in your IDE. We can make changes in the TypeScript code and test it in the browser without deploying it into Dynamics 365. This is very powerful and can save us a lot of time.

PACKAGE AS A SOLUTION FOR OUTSOURCING

We can create a complete web application and provide it on App Source by packaging the solution in Dynamics 365. The Dynamics 365 solution will be available on App Source as an add-in under "Free Trial" and "Contact Me" listing options. The add-in can be installed as a free trial from App Source and can be used cross tenants.

Offer types and add-ins	**Contact Me**	**Free Trial**	**Get It Now (Free)**	**BYOL**	**Get It Now (Transact)**

Dynamics 365 for Customer Engagements & PowerApps	AppSource	AppSource	--	--	--

The packaging and bundling are done by adding the required components in the solution and adding all the web pages (TypeScript as JavaScript, html, CSS, Images, etc.) in the web resources. The web application can be used as an IFRAME in an entity's form, depending on the requirement or it can be available as a button in the ribbon which will navigate the user to the web application in a new window.

AUTHENTICATE TO MICROSOFT DATAVERSE WITH WEB API

When we use the Web API with JavaScript/TypeScript within HTML web resources, form scripts, or ribbon commands we do not need to include any code for authentication. In each of these cases the user is already authenticated by the application and authentication is managed by the application.

CONCLUSION

In this case study, a brief introduction about Dynamics 365 development through Typescript, its requirements, deploying typescript in Dynamics 365 and debugging the typescript inside dynamics 365 using the developer tools is discussed.

Our Dynamics 365 Consulting, add-in Development through Typescript can help companies maximize business performance, overcoming market challenges, achieving profitability, and providing the best customer care service.

Contact Us

Cognitive Convergence

http://www.cognitiveconvergence.com

+1 4242530744

shahzad@cognitiveconvergence.com