EXERCISE 10B - Get Set to Use SAP Cloud SDK for JavaScript

SAP Partner Workshop	**15 min**

Description

In this exercise, you'll learn how

- How to scaffold your application
- About the project's structure
- How to run the application locally

Target group

- Developers
- People interested in learning about S/4HANA Cloud SDK for JavaScript.

Goal

The goal of this exercise is to set up the system to use SAP Cloud SDK for JavaScript.

Prerequisites

- Node.js, npm, SAP Cloud Platform Trial Account.
- Exercise 10A must be completed.

Steps

Step 1: Scaffold an application

Personally, we are fans of TypeScript and recommend using it for most applications. However, we also offer a version of our application scaffolding for plain JavaScript. If a step in a tutorial contains instructions or code that is specific to TypeScript or JavaScript, you can toggle between whatever you prefer at the top of each step.

The main differences you will notice between TypeScript and JavaScript are the type annotations and module definitions - ES6 modules in TypeScript vs. commonJS modules in JavaScript. To migrate a TypeScript file to JavaScript, you only need to change the file extension from .ts to .js, remove all type annotations and change the imports and exports.

To create an Express.js application that already contains all the files and configuration you need to use the SAP Cloud SDK for JavaScript, simply clone our TypeScript scaffolding application as follows.

```
git clone --single-branch --branch scaffolding-ts --origin scaffolding https://github.com/SAP/cloud-s4-sdk-examples.git <path/to/your/project>

cd <path/to/your/project>
```

Step 2: Get familiar with the project

The project contains the following files and folders, among others, to get you started with the SAP Cloud SDK for JavaScript:

NPM / Project

- **package.json**: Specifies dependencies, metadata and user-defined scripts. The provided scaffolding comes with some predefined scripts and dependencies, that will be explained in detail in the course of this group of tutorials.
- **.npmrc**: The **npm** configuration file. In the scaffolding we specify the registry for the @sap scope, where the SAP Cloud SDK libraries are published.

TypeScript

- **tsconfig.json**: Configuration file for TypeScript. This is not present in the plain JavaScript version.

Continuous Delivery

- **Jenkinsfile**: Jenkins pipeline definition file for quality assurance. It uses the SAP Cloud SDK's Continuous Delivery Toolkit.
- **pipeline_config.yml**: Pipeline configuration file for the Jenkins pipeline.
- **cx-server/**: A directory containing scripts to quickly deploy and start your own CI / CD server based on Jenkins.

Cloud Foundry

- **manifest.yml**: The deployment descriptor file for Cloud Foundry in SAP Cloud Platform.

Local development

- **initialize.js**: Script for initial setup.
- **src/**: Source code for an initial hello world express application.

Step 3: Run the application

Before you can run the app, you should run the initialization script. This will install your dependencies and replace some placeholders with your application name. This is only necessary once at the start of development. Feel free to delete this file afterwards.

```
npm run init -- <YOUR-APPLICATION-NAME>
npm run start:local
```

Go to **http://localhost:8080/hello** and you should get a ‘Hello, World!’ in response.

Step 4: Use the SDK in an existing project (Optional)

If you already have an existing project, you will need to specify the registry for the @sap scope, in order to make the SAP Cloud SDK for JavaScript libraries available. If it does not yet exist create a .npmrc file in the root folder of your project. Paste the following line into it:

```
@sap:registry=https://npm.sap.com
```

Now you can install the necessary libraries, first of all the @sap/cloud-sdk-core, the heart of the SAP Cloud SDK for JavaScript and basis for the service libraries you might want to use.

```
npm install @sap/cloud-sdk-core
```

We recommend to also take a look at the continuous delivery artifacts in the scaffold application and adopt those along with the respective **npm** scripts.

That’s it! You should now have a running application that is ready to be integrated with SAP S/4HANA Cloud. You can proceed to exercise 11A.