SESUG 2020 Paper 167

Building SAS®-Powered Web Apps with the SASjs Framework

Krishna Acondy, Allan Bowe, Analytium Group Ltd

ABSTRACT

Build SAS®-Powered HTML5 Web Apps faster than ever before, with the SASjs Framework - a set of tools and utilities for frontend, backend, build, deployment, documentation and more. The framework consists of a JS library, a SAS Macro library, and an Node Package Manager (NPM) command line tool - all of which can be used independently, but together provide a serious accelerator to web development projects.

This paper provides an overview of the framework and a quickstart guide to help you build and deploy a HTML5 app on SAS 9 or Viya. Prerequisites for using these tools for web development projects include SAS 9.3 or Viya, and NPM.

INTRODUCTION

The great thing about using SAS for enterprise web application development is that everything you need is available "out of the box" - the SAS Web Server, SAS Application Server (Stored Process on SAS 9 or Compute on Viya), database access engines, and authentication integration (SSO, LDAP).

There is nothing further to provision! You can just start coding. However - if you are serious about building many maintainable, multi-user, macro driven SAS Apps – the SASjs framework will smooth over the trickier bits and keep you focused on delivering valuable SAS-Powered interfaces.

Note that not all open source libraries are without encumbrances (eg GNU). Rest assured that the SASjs framework is (and will remain) fully MIT Open Source and free for commercial use.

THE SASJS FRAMEWORK

SASjs, as the name implies, is a set of SAS and Javascript libraries that provide an entry point into the SAS ecosystem.

SASjs includes:

- @sasjs/adapter—an isomorphic JavaScript library for interfacing with SAS servers from Node.js or the browser. Written in TypeScript.
- @sasjs/cli—a command-line NPM tool that automates creation, build and deployment of SAS-based web apps on both SAS 9 and Viya.
- @sasjs/core—a library of reusable SAS macros (not a JS library, but interfaces with the adapter). Referenced by the CLI when building & deploying SAS Services.
- @sasjs/test-framework—a React-based library that provides abstractions for testing SASjs services against a given server. By running tests directly within SAS, the need for whitelisting external test servers is hereby avoided.

Additionally, the following seed projects are also available for developers to use as the starting point for their app development journey:

- @sasjs/react-seed-app
- @sasjs/angular-seed-app
- @sasjs/minimal-seed-app

INSTALLATION

SASjs CLI is the gateway to the SASjs platform. Although it is possible to use the platform without the CLI, using it affords a number of advantages including the ability to scaffold out SASjs projects, create, build and deploy services and web apps, and the added benefit of being able to script the commands into CI/CD infrastructure.

SASjs components depend on Node.js and NPM, so these will have to be installed prior to starting with SASjs.

The CLI can then be installed globally using:

```
npm install -g @sasjs/cli
```

Once this is installed, the `sasjs` command will be available on the command line.

SCAFFOLDING AN HTML/CSS/JAVASCRIPT APP

You can use the SASjs CLI to generate a brand new HTML/CSS/JavaScript app as a starting point for your own app:

```
sasjs create <your-app-name> --template minimal
```

Or:

```
sasjs create <your-app-name> -t minimal
```

Running this command creates a folder with your app name. This folder contains:

- A `package.json` file that specifies the NPM package dependencies and includes a deploy script. This app only has one dependency - the `@sasjs/core` macro library.
- A `src` folder containing an HTML/CSS/JavaScript example app that shows how to use the SASjs adapter.
- A `sasjs` folder containing all the SAS services and macros necessary for the example app, along with a `sasjsconfig.json` file. We will look at the configuration in greater detail in the next section.
- A `node_modules` system folder, which contains all the third party dependencies for your app. This folder is managed by npm, there's no need to touch it.

The provided frontend app is simple and minimal. It includes three files:

- An `index.html` file that contains the HTML markup, a script tag that pulls in the `@sasjs/adapter` dependency, and a piece of JavaScript that initialises the adapter.

- A `scripts.js` file that shows how the adapter can be used to call SAS services on your server and handle the returned data.
- A `style.css` file with basic CSS styling.

CONFIGURATION

There are two main points of configuration in the generated app - the `index.html` file where the adapter is configured, and the `sasjsconfig.json` file which specifies the configuration for building and deploying SAS services using the CLI. Additional configuration is made in the `package.json` file if using the supplied template for synchronising the local build with the SAS Web Server.

SASJS ADAPTER CONFIGURATION

In the `index.html` file, the adapter needs to be configured to point to the right server to execute SAS code on.

It can include the following parameters:

- `serverUrl` - a string indicating URL of the SAS server (and port) to execute SAS services. If left blank, it is assumed to be the same as the server that the app is deployed on.
- `appLoc` - a string containing the path to the SAS folder (Metadata on SAS 9, or SAS Drive folder on Viya) in which the SAS services will be deployed.
- `serverType` - either SASVIYA or SAS9.
- `debug` - a flag that indicates whether jobs should be run in debug mode, which when set to true will produce additional debug output from job executions.

The SASjs adapter supports job execution via three approaches:

1. Via the SAS Stored Process web app (SAS 9 only)
2. Via the Job Execution Service API (SAS Viya only)
3. Via the Compute API (SAS Viya only)

To use the compute API, the additional parameters `contextName` (name of the compute context) and `useComputeApi` (true) must be set in the configuration. A detailed description of all the various configuration options for the adapter can be found in the documentation.

Once you have configured the adapter, you can deploy the app to your server either manually, or using the provided NPM `deploy` script and access it via the browser.

SASJS CLI CONFIGURATION

If you'd like to compile, build and deploy SAS services, some additional configuration is required.

The `sasjsconfig.json` file contains an array of targets. Each target represents a SAS server with properties such as `serverUrl` and `appLoc` that we saw in the section above. In addition, it also includes build-specific information such as deploy scripts, output file names and asset paths.

You can add a target via the CLI using the following command:

```
sasjs add
```

This will prompt for several items including the `appLoc`, server details, and client ID/secret if SASVIYA is chosen as the `serverType`. You can get a client ID and secret from your SAS administrator. If you are an administrator, you can use the SASjs Viya Token Generator to generate new client registrations. Be sure to select "Local" config to define the target within the current project.

The generated target will be saved to the `sasjsconfig.json` file. This file also contains a `cmnServices` array. This is a list of folder paths where the build process will look for SAS services. If you are creating a new service, it has to be placed in a subfolder in the services folder, and the path has to be included in the `cmnServices` array.

PACKAGE.JSON CONFIGURATION

The "package.json" file is used universally by NPM projects for managing third party dependencies. It also allows the execution of scripts (npm run scriptname) within the project.

You can configure the deployment of your frontend using the template NPM deploy script provided in package.json. This script requires two environment variables to be set (eg in .env):

- SSH_ACCOUNT — your SSH account, this is of the form username@domain.com
- DEPLOY_PATH — the path on the server where the app will be deployed to, typically /var/www/html/<some-subfolder> in SAS Viya.

More information on finding your static content folder is available in the deployment documentation.

You can run the script like so:

```
SSH_ACCOUNT=me@my-sas-server.com
DEPLOY_PATH=/var/www/html/my-folder/myapp
npm run deploy
```

This will lift & shift your frontend from your local (or build) environment to the SAS Web Server (or wherever you need your frontend to be served from).

C.B.D. LIFTOFF

SASJS COMPILE

Now we are all configured, the fun can begin! From the terminal, within your app project, run the following command:

```
sasjs compile
```

This command will loop through all of the .sas programs in the folders defined in the `cmnServices` array in the `sasjsconfig.json` file, and copy them over to a temporary `sasjsbuild/services/<subfolder>` directory, along with the following items:

- App specific macro variables
- Service specific macro dependencies
- App specific service initialisation & termination code

We now have one file per SAS service, and each service is fully self contained with all dependencies (zero code deployed to file system) – see Figure 1 below.

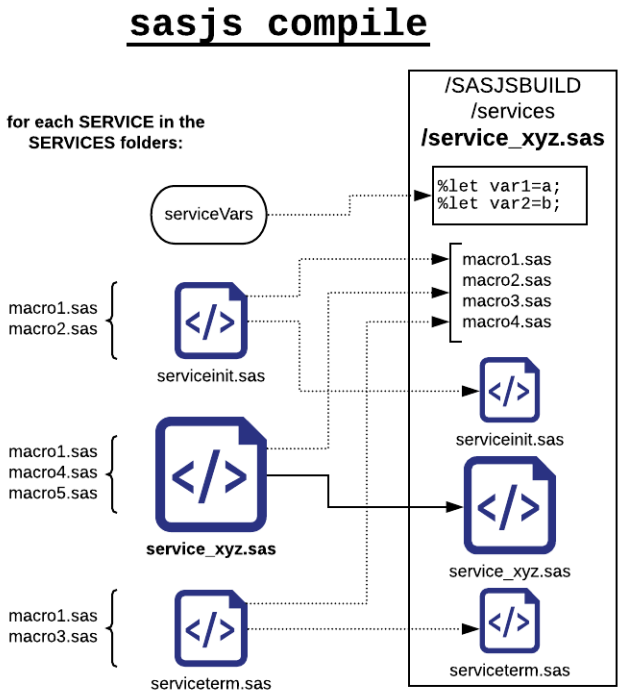

Figure 1 – Compiling SAS Services using SASjs

SASJS BUILD

Whilst it's nice and handy to have one file per SAS service, we still need to think about how we might get all of, say, 100 services for a particular SAS app ready for deployment. That's where the build command comes in:

```
sasjs build
```

This command takes all of the services and creates two more outputs in the temporary `sasjsbuild` folder:

1. A SAS program (named per `buildOutputFileName`) that can be executed in SAS Studio to create ALL services
2. A JSON file that can be used by `sasjs deploy` to create all services directly using the Viya REST APIs

The SAS program is very handy, as it can be used in SAS 9 to deploy all the services in one go by simply executing the code. No need for SAS Management Console, DI Studio, or the

enhanced "stored process creation" capability in Enterprise Guide! You do of course need write permissions on the target metadata folder.

The `sasjs build` command will automatically run `sasjs compile` if the `sasjsbuild` *folder* does not exist. Both commands can also be run together using the `sasjs cb` alias.

The build process not only allows you to create the services, you can also run pre and post build scripts to prepare the environment and deployment process, as illustrated in Figure 2 below.

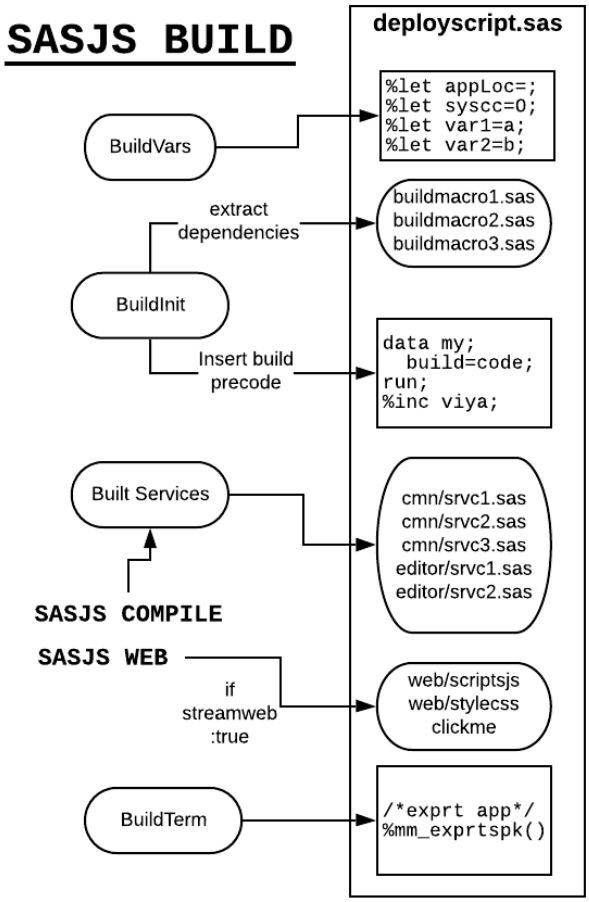

Figure 2 – Building SAS Services using SASjs

SASJS DEPLOY

Everything is set – now to deploy! Whilst the output of sasjs build can be copy pasted to create the SAS services, this can get tedious after a few deploys. Hence the command:

```
sasjs deploy
```

If the target `serverType` is Viya, the first action will be to deploy the services defined in the aforementioned JSON file. Then the command will sequentially execute the scripts listed in the `tgtDeployScripts` array. If the script is a SAS program, it will execute on the SAS server. If it is a shell or any other script type, it will execute locally in the build environment.

By placing the commands for frontend deployment in a script referenced here, both backend and frontend can be deployed with the same command.

To make this even easier / faster, you can compile / build / deploy in a single step – the alias is: `sasjs cbd`

CONCLUSION

We have explored some of the basic functionality of the SASjs framework, however there is yet more to learn. There is tooling in the CLI for Viya Compute context management, running arbitrary SAS code from commandline, executing web services from commandline, deploying databases, and compiling the frontend into services so they can be streamed without any need for accessing the SAS web server.

There are utilities for base64 encoding binary files, interfaces with the SAS 9 REST API and an upcoming plugin for VSCode. The entire framework underpins commercial software offerings (eg Data Controller for SAS and SAS Apps) and so will continue to be supported and improved.

We'd love to help you take your SASjs skills to the next level - join us on a free or paid training course, or engage our professional services to provide you with a quick start to your SAS App development project, STP Web App to Viya migration, or SAS AF/SCL Modernisation effort.

Happy coding!

RECOMMENDED READING

- *https://developer.sas.com/apis/rest*
- *https://sasjs.io/resources*
- *https://medium.com/@krishna.acondy/building-sas-apps-rapidly-with-sasjs-60ba2be0c4d4*

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Krishna Acondy - https://www.linkedin.com/in/krishna-acondy
Allan Bowe - https://www.linkedin.com/in/allanbowe