Relativity Forms Cheat Sheet

This document contains information briefly describing Relativity Forms, a suggested approach for getting started, advice on ramping-up, links to help, and documentation.

Relativity Forms

Relativity Forms is a new application that replaces the traditional form rendering pages for Relativity Dynamic Objects (RDOs). From an application developer's perspective, the most significant changes are to the page structure itself and to the behavior of event handlers typically used to alter the page. API documentation for Relativity Forms (including a getting started, with a quick step-by-step tutorial) can be found in the Platform Documentation for R1,
here: https://platform.relativity.com/RelativityOne/Content/Relativity_Forms/Relativity_Forms_API.htm

Ensuring your existing ADS application is ready for Relativity Forms

The two most common mechanisms used to customize the look-and-feel of a Forms page is through Page Interaction and Console event handlers. These two event handlers have undergone significant changes with Relativity Forms. This section describes the kinds of customizations that may exist in ADS Applications today that will need to be updated in order to be compatible with Relativity Forms.

Page Interaction Event Handlers

Relativity Forms has an entirely new page structure and therefore Document Object Model (DOM) as well. Since classic forms pages with page interaction event handlers typically modified the DOM directly, these event handlers will have to be updated. Although developers may continue to modify the DOM directly, Relativity strongly recommends that all applications use the new Relativity Forms JavaScript APIs whenever possible. Using the APIs will insulate you from changes to the underlying DOM.

Console Event Handlers

Server-side Console Event Handlers will no longer fire with Relativity Forms. Instead, consoles must be built client-side from within a Page Interaction Event Handler. Developers can build consoles using the console functionality on the convenienceAPI from within the createConsole and updateConsole pipeline events. If your console requires logic that must execute on the server, the best practice is to create a new Kepler service with your application and to call that new service from within your Page Interaction Event Handler using the relativityHttpClient.

Default Values

With classic forms pages, Pre Load Event Handlers could be used to supply default values on layouts for

new objects. With Relativity Forms, Pre Load Event Handlers no longer fire on new objects – though they do continue to fire on existing objects. In order to supply default values for new objects on a layout, handle the replaceGetNewObjectInstance pipeline event and set your default values there. If your default value logic requires logic that must execute on the server, the best practice is to create a new Kepler service with your application and to call that new service from within your Page Interaction Event Handler using the relativityHttpClient.

Layout Redirection

With classic forms, you could redirect users to a particular layout from within a Pre Load Event Handler. This mechanism does not work with Relativity Forms. We do not have a solution to support this functionality within Relativity Forms today, but it will be part of an upcoming release of Relativity Forms.

JavaScript in Event Handler Messages

Previously, in some scenarios, JavaScript embedded into Messages returned from Pre Load, Pre Save, or Post Save Event Handlers may have been fired. This is no longer supported. The recommended approach in these instances is to review the capabilities of the Relativity Forms JavaScript APIs and implement the custom workflow required using those new APIs. If you're having trouble, please reach out devhelp.

Getting up to speed with front-end work, specifically JavaScript:

Familiarity with JavaScript (particularly es5.1 and earlier)

If you are new to front-end work, we suggest spending a little bit of time becoming familiar with JavaScript – specifically es5.1 and lower, as es6/es2015+/esNext/typescript are not supported in event handler JavaScript files.

Specific Technical Reference

The best technical reference on the internet for quick JavaScript syntax help is the MDN (Mozilla DeveloperNetwork) web docs located here: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference.

If the team is attempting to use a certain JavaScript feature, and the MDN documents aren't clear enough about whether this feature is supported in our supported browsers (this is rare, but can happen), another good resource to check is https://caniuse.com/.

General Education

There are a lot of bad sources of information for JavaScript out there. Douglas Crockford's, "JavaScript: The Good Parts" is an excellent book on JavaScript, and it is still very relevant

now, despite its age. This is a very dry read, however, and it must be read serially, as each chapter depends upon context from every preceding chapter. For something easier to digest, the **"You Don't Know JS"** series said to be quite good, and it's 100% free online: https://github.com/getify/You-Dont-Know-JS

Promises

Also important is the concept of Promises in JavaScript. Relativity Forms makes use of them, and documentation references them in many places. It's not too difficult a concept, but it'd be a good idea to have read about them, and to have some familiarity with them:
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Using_promises
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise
http://promisesaplus.com/

Preparing your existing ADS application for Relativity Forms

1. Be familiar with the Platform's terminology, and concepts. If you are unfamiliar with Event Handlers (1, 2), Page Interaction Event Handlers, or Console Event Handlers, please read the platform documentation for these (linked in the text of this bullet).
2. Identify an RDO as part of your application that has customizations the form page. This is typically a Page Interaction and/or a Console Event Handler.
3. View an instance of that RDO in Relativity as it presently is.
4. Unlock the application.
5. Enable Relativity Forms for the type by setting the "**Use Relativity Forms**" to "Yes" and saving.
6. View that same RDO instance. Note that the Console and any Page Interaction Event Handler customizations are gone.
7. Following the landing page for Relativity Forms, create a basic event handler JavaScript file, alter the existing (.NET) PageInteractionEventHandler for the RDO to include the newly created JavaScript file in the value for the "ScriptFileNames" property. Once this is DLL is updated in Relativity, and the JavaScript file is uploaded to Relativity as a resource file, and associated with the application, the script file will be used by Relativity Forms. If the same stub code is used for this script file as in the code sample on the landing page, when you view the RDO instance in Relativity with a developer console open (F12 in Chrome), the message "Page load complete" will appear in the console, whether you're looking at the object in View, Edit, or Create mode.
8. With that basic, does-nothing-special code in place, you will then be able to edit your event handler script file to do much more powerful things. The last section in the Relativity Forms API landing page details updating the event handler JavaScript file, and the rest of the platform documentation within the Relativity Forms API tree gives detail of high-level concepts of Relativity Forms, and of the APIs available to you as you expand the capabilities of the object type.
9. Possible gotcha: Object Rules still work in Relativity Forms exactly as they do in Classic, so if the RDO has Object Rules attached which for example override URLS for Create/View/Edit, it's quite possible you wouldn't be seeing Relativity Forms in those circumstances.
10. Above (in point 2), finding an RDO with both PageInteractionEventHandler (PIEH) and ConsoleEventHandler is recommended. This is because:
    a. If the PIEH exists already, you only have to alter some existing code rather than write from scratch, and
    b. If there's a ConsoleEventHandler, it'll be very clear by looking at an object instance in View mode that there is no Console in step 6.

Assuming you're looking at this kind of an RDO scenario, we suggest making the first *real* customization within RelativityForms be the addition of a Console within Relativity Forms. This is accomplished in the JavaScript by implementing a handler for the eventNames.CREATE_CONSOLE event (and if needed, the eventNames.UPDATE_CONSOLE event). Particularly relevant to creating consoles is the console object on the convenienceApi.