Checking the Admin Role in UI Builder `evaluateProperty`

There are two main ways to check a user's roles in a UI Builder scripted property: (a) **preloading the data** (e.g. via session context or a UI Builder Data Resource) and checking it locally, or (b) **fetching roles on-the-fly** via a server call (e.g. using `api.getData`). Below we outline both approaches with sample code and discuss their trade-offs.

1. Preloaded Data Resource (Client-side check)

UI Builder automatically provides the session user's info (including roles) in the context. You can use that directly in your `evaluateProperty`. For example, the following script reads the roles array from `api.context.session.user.roles` and checks for `'admin'`:

```
function evaluateProperty({ api, helpers }) { // Approach 1: use preloaded session data / data resource // This assumes a Data Resource (or the session context) has already loaded the current user's roles. const roles = api.context.session.user.roles; // e.g. ['admin','itil_user'] return Array.isArray(roles) && roles.includes('admin'); // true if 'admin' is present }
```

Here `api.context.session.user.roles` is an array of role names for the logged-in user, made available by UI Builder servicenow.com. (Alternatively, you can create a *Data Resource* on, say, the `sys_user_has_role` table filtered by the current user, and then use `api.data.myUserRoles` to read those records.) For instance, if you define a Data Resource named `userRoles` that queries `sys_user_has_role` for the current user, the code would be:

```
function evaluateProperty({ api, helpers }) { const userRoles = api.data.userRoles; // preloaded records from sys_user_has_role // Each record has a .role.name field. Check if any record's role is 'admin'. return userRoles && userRoles.some(r => r.role.name === 'admin'); }
```

Using preloaded data (context or Data Resources) means the roles are fetched **before** rendering. UI Builder Data Resources "fetch data from the instance to the page" and make it available to components developer.servicenow.com. This lets you check the role locally without a new network call in the property script.

2. Server-side Synchronous API Call

The alternative is to call the server at evaluation time. For example, you could invoke the Table API via `api.getData`. The script might look like this:

```
js                                                    Copy    Edit

function evaluateProperty({ api, helpers }) { // Approach 2: synchronous server call
inside evaluateProperty const userId = api.context.session.user.sysId; // Query
sys_user_has_role for current user & 'admin' role const result =
api.getData('sys_user_has_role', { sysparm_query: `user=${userId}^role.name=admin` }); //
If any records returned, the user has the 'admin' role return (result.records ||
[]).length > 0; }
```

This code uses `api.getData` (a UI Builder API) to synchronously query the `sys_user_has_role` table for the current user's admin role. If the query returns one or more records, the user has the admin role. (You could also use a Script Include or Transform Data Resource on the server and call it similarly.)

Comparison: Preloaded Data vs. Runtime API Call

- **Performance:** Preloading roles (via context or a UI Builder Data Resource) is generally faster. UI Builder can cache and reuse data – "multiple components can hook into a shared state" so data is fetched only once servicenow.com. In contrast, an `api.getData` call runs at evaluation time and blocks the UI until the server responds, which can slow page load or cause flicker. Data Resources are designed to "de-dupe data and only fetch it once" servicenow.com, whereas ad-hoc calls in `evaluateProperty` bypass those optimizations.
- **User Experience:** With preloaded data, the UI can render immediately using the already-loaded roles. The synchronous approach may delay component rendering or require additional loading states. A blocking call in `evaluateProperty` could make the page feel sluggish, especially if done repeatedly.
- **Best Practices:** ServiceNow recommends using Data Resources and binding data over custom scripting whenever possible. UI Builder's architecture encourages separating data fetching from presentation developer.servicenow.com servicenow.com. Fetching data in a scripted property is allowed but should be minimized. In particular, avoid heavy or repeated server calls in `evaluateProperty`. Instead, leverage the built-in session context or a preconfigured data resource.
- **Security & Reliability:** Both methods respect security (user can only see roles they have). Preloading via the session context or data resource uses ServiceNow's standard access controls. A custom script call could likewise enforce ACLs. However, adding custom server calls introduces more moving parts (REST APIs, query correctness, error handling).

Key takeaway: the *preloaded data* method (solution 1) is cleaner and more performant. For example, the UI Builder community shows using `api.context.session.user.roles.includes('admin')` directly servicenow.com . Using Data Resources also aligns with the UI Builder model of centralizing data fetch logic developer.servicenow.com servicenow.com . The runtime API call (solution 2) works, but should be used sparingly since it negates those built-in optimizations.

Recommendation: For production use, prefer the data-resource (or session context) approach. Define a user-roles resource or use `api.context.session.user.roles` and check `includes('admin')`. This leverages UI Builder's asynchronous data loading and caching servicenow.com . Use the direct server call approach only if absolutely necessary (e.g. no other way to get the data), and be mindful of the performance impact.

References: The examples above follow documented patterns in UI Builder for accessing user roles servicenow.com and for using data resources to fetch instance data developer.servicenow.com servicenow.com . These sources explain that UI Builder data resources are optimized for single-fetch, shared use cases, whereas custom script calls run at evaluation time.

Citations

Solved: Hide a component on workspace based on role - ServiceNow Community
https://www.servicenow.com/community/developer-forum/hide-a-component-on-workspace-based-on-role/m-p/2357022

UI Builder - Data Resources
https://developer.servicenow.com/blog.do?p=/post/quebec-ui-builder-data-resources/

All About Data Resources in UI Builder - ServiceNow Community
https://www.servicenow.com/community/next-experience-articles/all-about-data-resources-in-ui-builder/ta-p/2360643