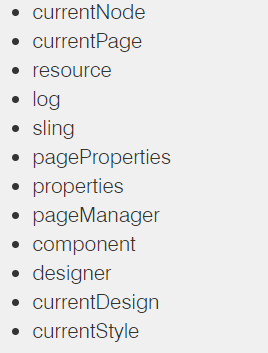
JavaScript Use API

<https://docs.adobe.com/content/help/en/experience-manager-htl/using/htl/use-api-javascript.html>

AEM has implemented [Rhino](https://developer.mozilla.org/en-US/docs/Mozilla/Projects/Rhino). According to the Rhino documentation, “Rhino is an open-source implementation of JavaScript written entirely in Java.” In a nutshell, Rhino takes JavaScript and converts it to Java.

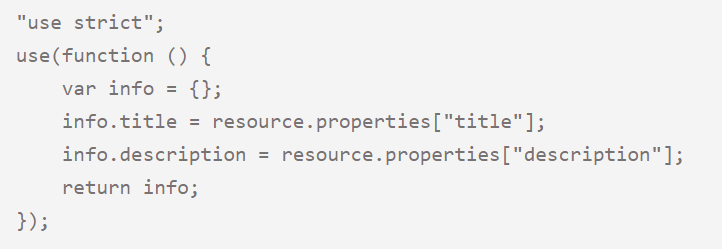
The HTML Template Langugae (HTL) JavaScript Use-API enables a HTL file to access helper code written in JavaScript. This allows all complex business logic to be encapsulated in the JavaScript code, while the HTL code deals only with direct markup production.

**Implicit objects available.**

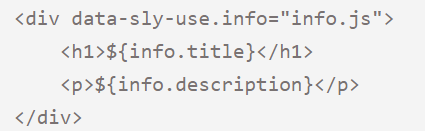


**Example1:**

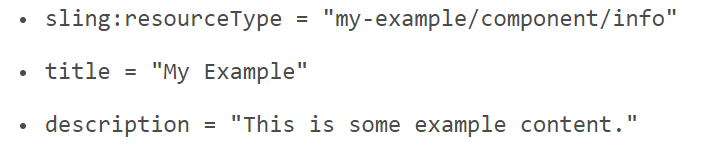
**/apps/my-example/component/info/info.js**



**/apps/my-example/component/info/info.html**



**/content/my-example**



**Example2:**

Using implicit objects and methods



## **Dependencies**

Let's imagine that we have a utility class that is already equipped with smart features, like the default logic for the navigation title or nicely cutting a string to a certain length:



## **Extending**

The dependency pattern can also be used to extend the logic of another component (which typically is the sling:resourceSuperType of the current component).

Imagine that the parent component already provides the title, and we want to add a description as well:



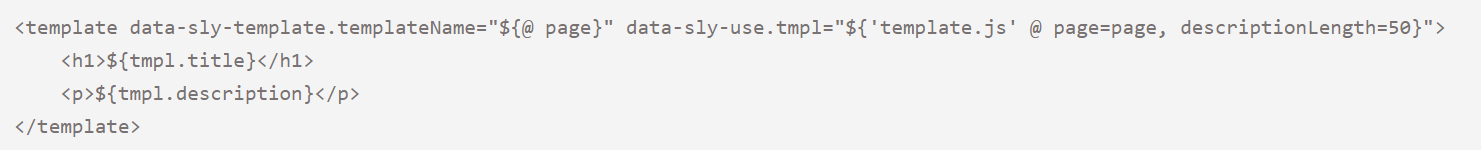
## **Passing Parameters to a Template**

In the case of data-sly-template statements that can be independent from components, it can be useful to pass parameters to the associated Use-API.

So in our component let's call a template that is located in a different file:



Then this is the template located in template.html:

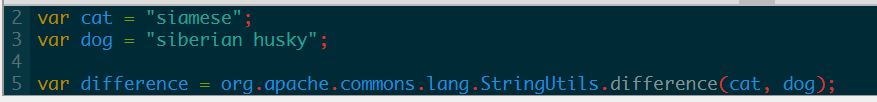


The corresponding logic can be written using following server-side JavaScript, located in a template.js file right next to the template file:



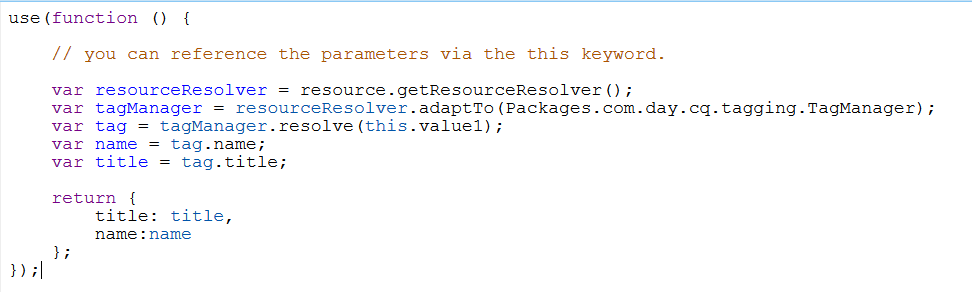
## **Java in the JavaScript**

Java classes are accessible through our JavaScript. In JavaScript, there is no import statement for bringing in additional classes. Instead, to access a class, simply use its class path. For example, to take advantage of the difference method in Java’s StringUtils, I would write the following:



## **Using java APIs in JavaScript**

Getting tag details, by passing the tag path



Front end html code looks like below.

**<sly data-sly-use.params="${'comp.js' @value=value}"></sly>**

### Pushing objects in to array