In case you are using SAP Web IDE, you can right-click the project and select New  HTML5 Application Descriptor to make the /resources… reference work. This creates the neo-app.json file, which configures a URL mapping for this path.

* The src attribute of the <script> tag tells the browser where to find the OpenUI5 core library – it initializes the OpenUI5 runtime and loads additional resources, such as the libraries specified in the data-sap-ui-libs attribute.
* The OpenUI5 controls support different themes, we choose sap\_belize as our default theme.
* We specify the required UI library sap.m containing the UI controls we need for this tutorial.
* To make use of the most recent functionality of OpenUI5 we define the compatibility version as edge.
* We configure the process of “bootstrapping” to run asynchronously.

This means that the OpenUI5 resources can be loaded simultaneously in the background for performance reasons.

* We define the module to be loaded initially in a declarative way. With this, we avoid directly executable JavaScript code in the HTML file. This makes your app more secure. We will create the script that this references to further down in this step.
* We tell OpenUI5 core that resources in the sap.ui.demo.walkthrough namespace are located in the same folder as index.html. This is, for example, necessary for apps that run in the SAP Fiori launchpad.

**When all resources and libraries are loaded, the OpenUI5 runtime fires the global init event to signal that the library is ready. It is a good practice to listen for this event in order to trigger your application logic only after the event has been fired.**

In the example above, we get a reference to the OpenUI5 core by calling **sap.ui.getCore()** and register an anonymous callback function for the init event by calling **attachInit(…)** on the core

index**.js** script that will contain the **application logic** for this tutorial step

## Conventions

* Controller names are capitalized
* Controllers carry the same name as the related view (if there is a 1:1 relationship)
* Event handlers are prefixed with on
* Controller names always end with \*.controller.js
* Use sap.ui.define for controllers and all other JavaScript modules to define a global namespace. With the namespace, the object can be addressed throughout the application.
* Use sap.ui.require for asynchronously loading dependencies but without declaring a namespace, for example code that just needs to be executed, but does not need to be called from other code.
* Use the name of the artifact to load for naming the function parameters (without namespace).