**<script**

**src=""**

-> core library path

**theme**

-> supports themes

**libs**

-> UI library "sap.m"

**compatVersion**

-> "edge" for most recent funcs

**async "true"**

-> process of "bootstrapping" to be async

**onInit**

-> module loaded initially, in a declarative way, to avoid directly executable JS code in HTML file (more secure)

**resourceroots sap.ui.demo.walkthrough": "./"**

-> we tell that resources are located in the same folder index.html

**> </script>**

**index.js contains the app logic**, which will be called by index.html, where we define it as a module in a declarative way to avoid having executable code directly in the HTML file. (security)

/\* ----------------------------- \*/

The **class sapUiBody** adds additional theme-dependent styles for displaying SAPUI5 apps.

The name of the control is prefixed by the namespace of its control library **sap/m/** and the options are passed to the constructor **with a JavaScript object**.

**placeAt** that is used to place SAPUI5 controls inside a node of the document object model (DOM) or any other SAPUI5 control instance. We pass the ID of a DOM node as an argument.

**! Note**

*Only instances of sap.ui.core.Control or their subclasses can be rendered stand-alone and have a placeAt function. Each control extends sap.ui.core.Element that can only be rendered inside controls. Check the API reference to learn more about the inheritance hierarchy of controls. The API documentation of each control refers to the directly known subclasses.*

/\* ----------------------------- \*/

/\* XML View \*/

Using XML to force to separate the view declaration from the controller logic.

The root node of the XML structure is the view. Here, we reference the default namespace sap.m where the majority of our UI assets are located.

We define an additional sap.ui.core.mvc namespace with alias mvc, where the SAPUI5 views and all other Model-View-Controller (MVC) assets are located.

**Note**

*The namespace identifies all resources of the project and has to be unique. If you develop your own application code or controls, you cannot use the namespace prefix sap, because this namespace is reserved for SAP resources. Instead, simply define your own unique namespace (for example, myCompany.myApp).*

/\* ----------------------------- \*/

*/\* CONTROLLER \*/*

A view does not necessarily need an explicitly assigned controller. You do not have to create a controller if the view is just displaying information and no additional functionality is required. If a controller is specified, it is instantiated after the view is loaded.

We create the folder webapp/controller and a new file App.controller.js inside.

We define the app controller in its own file by extending the Controller object of the SAPUI5 core.

**Conventions**

1. Controller names are capitalized
2. Controllers carry the same name as the related view (if there is a 1:1 Relationship)
3. Event handlers are prefixed with *on*
4. Controller names always end with “.controller.js”

/\* ----------------------------- \*/