UI5


MVC
Framework
Data Binding
Rich UI
Controls(500+)
Responsive
Components
Models
Flexibility
Extensibility
UX(Fiori)
Routing
Events
i18n
RTL
Themes
HTML5
CSS3

SAPUI5 VS OPENUI5

"Difference is the license and proprietary features"

SAPUI5 is not separate product, its integrate with other SAP products like OCB

SAP UI5 has more additional controls and libraries like chart, visualization, smart and flexibility etc

BOOTSTRAPPING

Loading of framework is called as "Bootstrapping"

```
<script>
  id="sap-ui-bootstrap"
  src="resources/sap-ui-core.js"
  data-sap-ui-theme="sap_belize"
  data-sap-ui-libs="sap.m"
  data-sap-ui-preload="async" >
</script>
```

```
sap.ui.getCore().attachInit(function () {
  console.log("Initialization callback!");
});
```

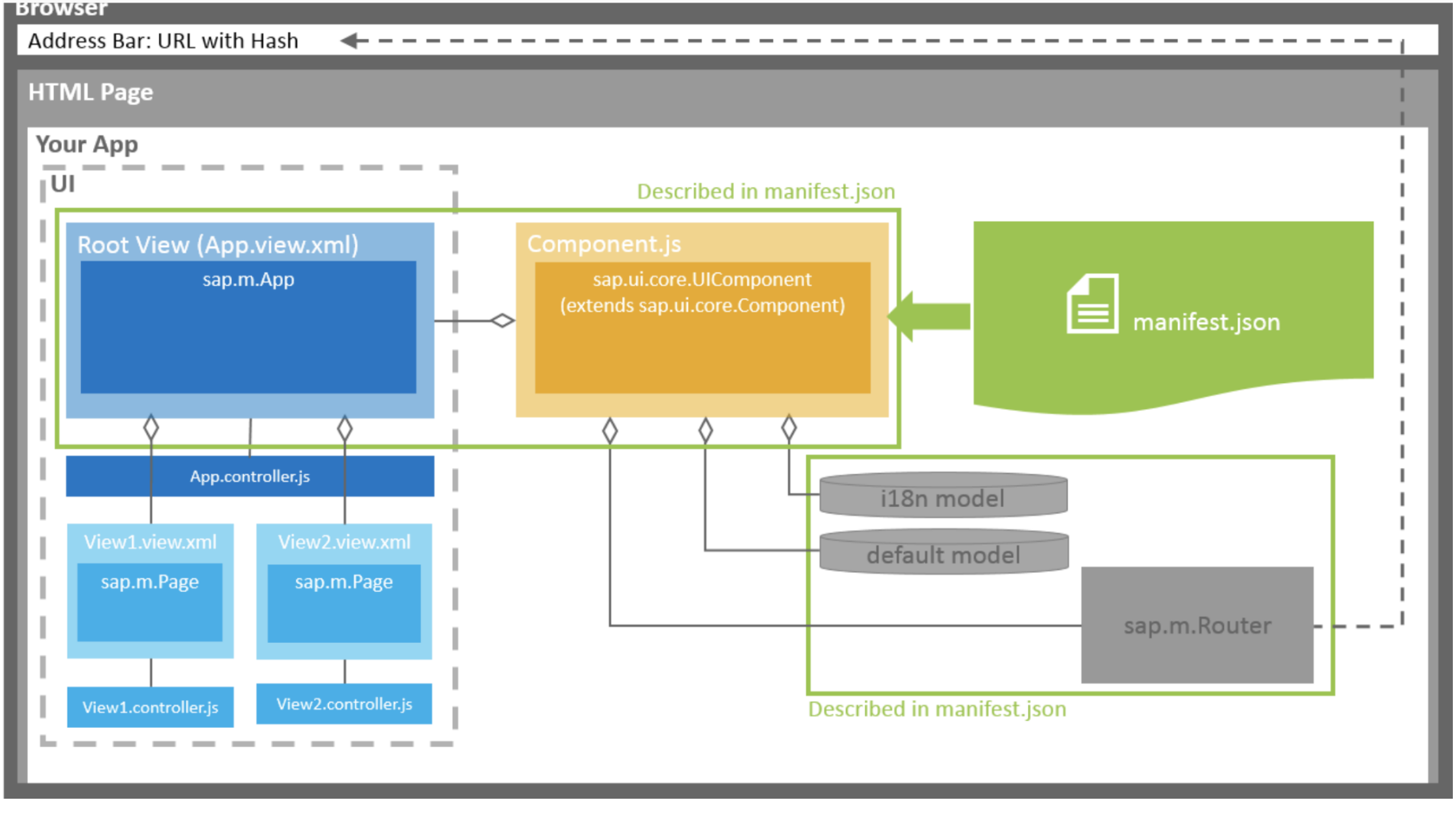
DEVELOPMENT ENVIRONMENT

Editor: VSCode, Sublime, Eclipse

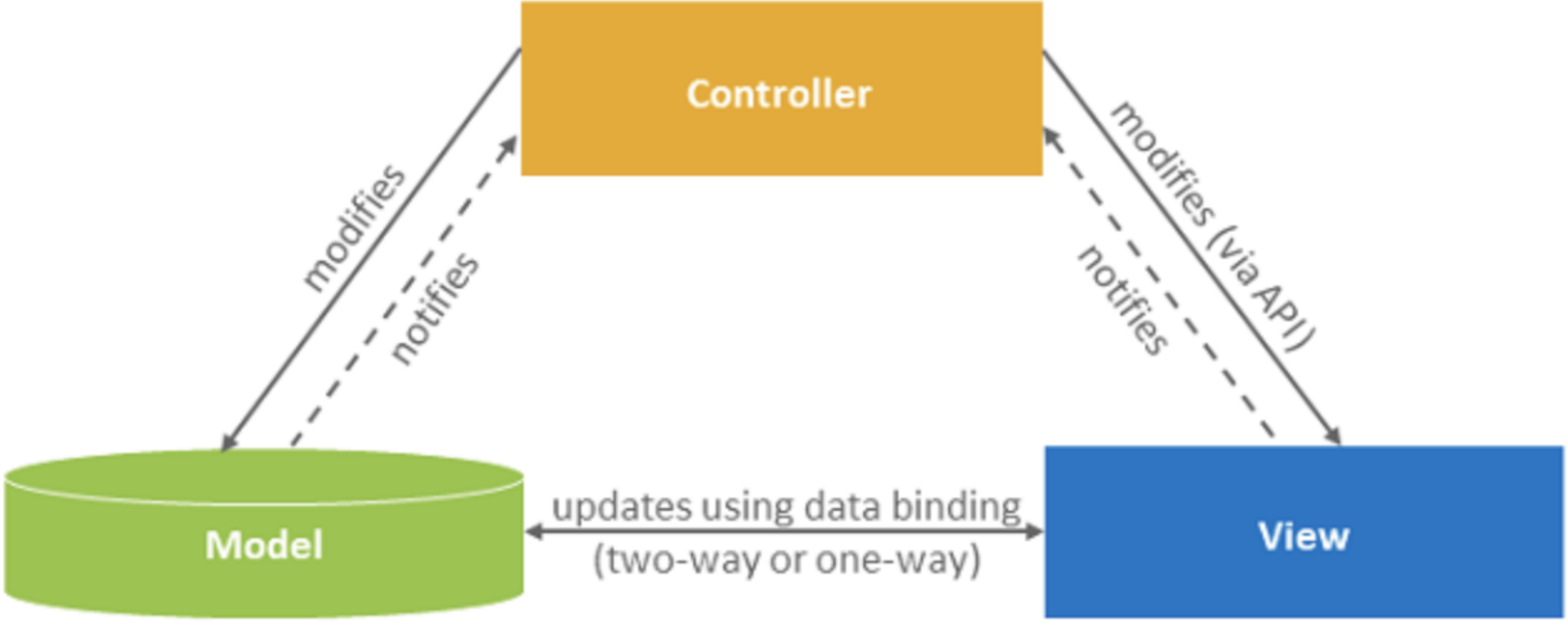
Server: Node.js, Apache, any HTTP Server

You can also develop app using WebIDE (Cloud Editor)

APPLICATION STRUCTURE



MVC


Controller
modifies
notifies
modifies (via API)
notifies
Model
updates using data binding
(two-way or one-way)
View

VIEWS

"View represents the virtual visual boundary where the related UI elements will be placed"

VIEW

- Helps to keep representation separate from controlling unit
- UI5 supports 4 different types (XML|JS|JSON|HTML)
- Views will be transpiled down to dom elements

XML VIEW

```
<mvc:View
    controllerName="sap.m.sample.ListCounter.List"
    xmlns:mvc="sap.ui.core.mvc"
    xmlns="sap.m">
    <List
        headerText="Products"
        items="{
            path: '/ProductCollection'
        }" >
        <StandardListItem
            title="{Name}"
            counter="{Quantity}"/>
    </List>
</mvc:View>
```

DOM REPRESENTATION

```
▼<div id="__xmlview10" data-sap-ui="__xmlview10" data-sap-ui-preserve="__xmlview10" style="width:100%"
class="sapUiView sapUiXMLView">
  ▼<div id="__list6" data-sap-ui="__list6" data-sap-ui-fastnavgroup="true" style="width:100%" class=
  "sapMList sapMListBGSolid">
      <header class="sapMListHdr sapMListHdrText" id="__list6-header">Products</header> == $0
      <div id="__list6-before" tabindex="-1" class="sapMListDummyArea"></div>
    ▼<ul role="listbox" id="__list6-listUl" tabindex="0" class="sapMListItems sapMListModeNone
    sapMListShowSeparatorsAll sapMListUl">
      ▶<li id="__item166-__list6-0" data-sap-ui="__item166-__list6-0" tabindex="-1" role="option" class=
      "sapMLIB sapMLIB-CTX sapMLIBFocusable sapMLIBShowSeparator sapMLIBTypeInactive sapMSLI">…</li>
      ▶<li id="__item166-__list6-1" data-sap-ui="__item166-__list6-1" tabindex="-1" role="option" class=
      "sapMLIB sapMLIB-CTX sapMLIBFocusable sapMLIBShowSeparator sapMLIBTypeInactive sapMSLI">…</li>
      ▶<li id="__item166-__list6-2" data-sap-ui="__item166-__list6-2" tabindex="-1" role="option" class=
      "sapMLIB sapMLIB-CTX sapMLIBFocusable sapMLIBShowSeparator sapMLIBTypeInactive sapMSLI">…</li>
      ▶<li id="__item166-__list6-3" data-sap-ui="__item166-__list6-3" tabindex="-1" role="option" class=
      "sapMLIB sapMLIB-CTX sapMLIBFocusable sapMLIBShowSeparator sapMLIBTypeInactive sapMSLI">…</li>
      ▶<li id="__item166-__list6-4" data-sap-ui="__item166-__list6-4" tabindex="-1" role="option" class=
      "sapMLIB sapMLIB-CTX sapMLIBFocusable sapMLIBShowSeparator sapMLIBTypeInactive sapMSLI">…</li>
      ▶<li id="__item166-__list6-5" data-sap-ui="__item166-__list6-5" tabindex="-1" role="option" class=
      "sapMLIB sapMLIB-CTX sapMLIBFocusable sapMLIBShowSeparator sapMLIBTypeInactive sapMSLI">…</li>
```

CONTROLLERS

"Controllers are responsible for controlling UI, react to user events, fetch data, updating the state of UI"

CONTROLLER EXAMPLE

```
sap.ui.define([
    "sap/ui/core/mvc/Controller"
], function (Controller) {
    "use strict";
    return Controller.extend("demo.controller.App", {
      onInit: function () {}, // initialization e.g service
      onBeforeRendering: function () {}, // Before getting
      onAfterRendering: function () {}, // After getting re
      onExit: function () {}, // Place to cleanup
      onSubmitPress : function () {
        // Event handler
      }
    });
});
```

CONTROLLER EXAMPLE

```
sap.ui.define([
    "sap/ui/core/mvc/Controller"
], function (Controller) {
    "use strict";
    return Controller.extend("demo.controller.App", {
      onInit: function () {}, // initialization e.g service
      onBeforeRendering: function () {}, // Before getting
      onAfterRendering: function () {}, // After getting re
      onExit: function () {}, // Place to cleanup
      onSubmitPress : function () {
        // Event handler
      }
    });
});
```

CONTROLLER EXAMPLE

```
sap.ui.define([
    "sap/ui/core/mvc/Controller"
], function (Controller) {
    "use strict";
    return Controller.extend("demo.controller.App", {
      onInit: function () {}, // initialization e.g service
      onBeforeRendering: function () {}, // Before getting
      onAfterRendering: function () {}, // After getting re
      onExit: function () {}, // Place to cleanup
      onSubmitPress : function () {
        // Event handler
      }
    });
});
```

CONTROLLER EXAMPLE

```
sap.ui.define([
    "sap/ui/core/mvc/Controller"
], function (Controller) {
    "use strict";
    return Controller.extend("demo.controller.App", {
      onInit: function () {}, // initialization e.g service
      onBeforeRendering: function () {}, // Before getting
      onAfterRendering: function () {}, // After getting re
      onExit: function () {}, // Place to cleanup
      onSubmitPress : function () {
        // Event handler
      }
    });
});
```

CONTROLLER EXAMPLE

```
1  sap.ui.define([
2      "sap/ui/core/mvc/Controller"
3  ], function (Controller) {
4      "use strict";
5      return Controller.extend("demo.controller.App", {
6        onInit: function () {}, // initialization e.g service
7        onBeforeRendering: function () {}, // Before getting
8        onAfterRendering: function () {}, // After getting re
9        onExit: function () {}, // Place to cleanup
10       onSubmitPress : function () {
11         // Event handler
12       }
13     });
14 });
```

CONTROLLER EXAMPLE

```
sap.ui.define([
    "sap/ui/core/mvc/Controller"
], function (Controller) {
    "use strict";
    return Controller.extend("demo.controller.App", {
      onInit: function () {}, // initialization e.g service
      onBeforeRendering: function () {}, // Before getting
      onAfterRendering: function () {}, // After getting re
      onExit: function () {}, // Place to cleanup
      onSubmitPress : function () {
        // Event handler
      }
    });
});
```

CONTROLLER EXAMPLE

```
sap.ui.define([
    "sap/ui/core/mvc/Controller"
], function (Controller) {
    "use strict";
    return Controller.extend("demo.controller.App", {
      onInit: function () {}, // initialization e.g service
      onBeforeRendering: function () {}, // Before getting
      onAfterRendering: function () {}, // After getting re
      onExit: function () {}, // Place to cleanup
      onSubmitPress : function () {
        // Event handler
      }
    });
});
```

MODELS

"A model is an object which holds the state an also knows about the state changes"

MODELS

- UI5 provides models like JSON, OData, Resource Model
- Models support two way binding with UI elements
- When bound with two way, It collects the data from UI Elements when control property changes
- When model is updated it also updates the UI elements on the view

JSON MODEL

```
sap.ui.define([
    "sap/ui/core/mvc/Controller",
    "sap/ui/model/json/JSONModel"
], function (Controller, JSONModel) {
    "use strict";
    return Controller.extend("demo.controller.App", {
        onInit : function () {
            const oData = {
                firstName : "",
                lastName: "",
            };
            const oModel = new JSONModel(oData);
            this.getView().setModel(oModel);
        }
    });
```

JSON MODEL

```
sap.ui.define([
    "sap/ui/core/mvc/Controller",
    "sap/ui/model/json/JSONModel"
], function (Controller, JSONModel) {
    "use strict";
    return Controller.extend("demo.controller.App", {
        onInit : function () {
            const oData = {
                firstName : "",
                lastName: "",
            };
            const oModel = new JSONModel(oData);
            this.getView().setModel(oModel);
        }
    });
```

JSON MODEL

```
2     "sap/ui/core/mvc/Controller",
3     "sap/ui/model/json/JSONModel"
4 ], function (Controller, JSONModel) {
5     "use strict";
6     return Controller.extend("demo.controller.App", {
7         onInit : function () {
8             const oData = {
9                 firstName : "",
10                lastName: "",
11            };
12            const oModel = new JSONModel(oData);
13            this.getView().setModel(oModel);
14        }
15    });
16 });
```

JSON MODEL

```
    "sap/ui/core/mvc/Controller",
    "sap/ui/model/json/JSONModel"
], function (Controller, JSONModel) {
    "use strict";
    return Controller.extend("demo.controller.App", {
        onInit : function () {
            const oData = {
                firstName : "",
                lastName: "",
            };
            const oModel = new JSONModel(oData);
            this.getView().setModel(oModel);
        }
    });
});
```

JSON MODEL

```
2     "sap/ui/core/mvc/Controller",
3     "sap/ui/model/json/JSONModel"
4  ], function (Controller, JSONModel) {
5     "use strict";
6     return Controller.extend("demo.controller.App", {
7        onInit : function () {
8           const oData = {
9              firstName : "",
10             lastName: "",
11          };
12          const oModel = new JSONModel(oData);
13          this.getView().setModel(oModel);
14       }
15    });
16 });
```

ODATA MODEL

"A server connected model"

With OData model we can call REST OData Services

```
const oModel = new sap.ui.model.odata.v2.ODataModel({
    serviceUrl: "{ServiceEndPoint}",
    headers: {
        "x-custom-header" : "{headerValue}",
    }
});
```

CRUD Operations

```
1 const oTransfer = {}
2
3 oModel.create("/Transfers", oData, {
4   success: () => {},
5   error: () => {}
6 });
7 oModel.read("/Transfers(123)", {success: ()=>{}, error: ()=>
8 oModel.update("/Transfers(123)", oData, {success: ()=>{}, er
9 oModel.remove("/Transfers(123)", {success: ()=>{}, error: ()
```

CRUD Operations

```
1 const oTransfer = {}
2
3 oModel.create("/Transfers", oData, {
4   success: () => {},
5   error: () => {}
6 });
7 oModel.read("/Transfers(123)", {success: ()=>{}, error: ()=>
8 oModel.update("/Transfers(123)", oData, {success: ()=>{}, er
9 oModel.remove("/Transfers(123)", {success: ()=>{}, error: ()
```

CRUD Operations

```
1 const oTransfer = {}
2
3 oModel.create("/Transfers", oData, {
4   success: () => {},
5   error: () => {}
6 });
7 oModel.read("/Transfers(123)", {success: ()=>{}, error: ()=>
8 oModel.update("/Transfers(123)", oData, {success: ()=>{}, er
9 oModel.remove("/Transfers(123)", {success: ()=>{}, error: ()
```

CRUD Operations

```
1 const oTransfer = {}
2
3 oModel.create("/Transfers", oData, {
4   success: () => {},
5   error: () => {}
6 });
7 oModel.read("/Transfers(123)", {success: ()=>{}, error: ()=>
8 oModel.update("/Transfers(123)", oData, {success: ()=>{}, er
9 oModel.remove("/Transfers(123)", {success: ()=>{}, error: ()
```

CRUD Operations

```
1 const oTransfer = {}
2
3 oModel.create("/Transfers", oData, {
4   success: () => {},
5   error: () => {}
6 });
7 oModel.read("/Transfers(123)", {success: ()=>{}, error: ()=>
8 oModel.update("/Transfers(123)", oData, {success: ()=>{}, er
9 oModel.remove("/Transfers(123)", {success: ()=>{}, error: ()
```

CONTROLS

"Primary building blocks of UI"

CONTROLS

- UI5 is having rich set of controls with more than 500 controls in library categorized as mobile, desktop, chart, maps, smart etc
- Every control will have state (properties), events and representation
- The representation in DOM will reflect its current state

Instance definition in XML

```
<button type="Accept" text="Accept" press="onAcceptButtonPress
</button>
```

SMART CONTROLS

UI have very specialized controls which can render to specific type based on property type which is bound to control

MODULES

"UI5 has built in support for modularizing applications, Instead of writing big bundle we can split in to smaller logical units"

MODULES

- UI5 modules should follow AMD pattern (define, require)
- A module can be represented in a separate javascript file
- We should clearly separate the dependencies

```
1 sap.ui.define([], function() {
2   "use strict";
3
4   const BankingService = function name(params) {
5   };
6
7    return BankingService;
8 });
```

MODULES

- UI5 modules should follow AMD pattern (define, require)
- A module can be represented in a separate javascript file
- We should clearly separate the dependencies

```
1 sap.ui.define([], function() {
2   "use strict";
3
4   const BankingService = function name(params) {
5   };
6
7    return BankingService;
8 });
```

COMPONENTS

"Reusable independent parts of UI5 application"

COMPONENTS

- Can be embedded in other applications or can run independently
- Component can have its own views, controllers, models can follow its own MVC
- Component meta information is stored in manifest.json
- Component can have their own state

REUSABILITY

"Don't Repeat Yourself (DRY), A basic principal useful for software development"

REUSABILITY IN UI5

There are multiple ways to reuse UI components

FRAGMENTS

- Are reusable elements which shares same representation
- Fragments can be embedded in views, other fragments as well as programmatically inside controllers

```
<core:fragmentdefinition xmlns="sap.m" xmlns:core="sap.ui.core
    <dialog id="messageDialog" title="{/message}">
    </dialog>
</core:fragmentdefinition>
```

CUSTOM CONTROLS

A custom control will allow you to reuse certain functionality e.g Lookup, CustomPassword

COMPONENTS

A component will allow to reuse the UI representation in more independent way example

```
<core:componentcontainer name="common.account" settings="{
    accountId: '1001'
  }">
</core:componentcontainer>

<core:componentcontainer name="common.account" settings="{
    accountId: '1002'
  }">
</core:componentcontainer>
```

INHERITANCE

An object oriented approach for reuse, UI5 components, controls are fully extensible e.g All UI5 Objects classes can be further extended by javascript prototype based inheritance

DATA BINDING

"Data binding is the way where data can be bound to UI elements without need of writing manual code"

Why data binding is useful?

- Minimizes the logic/code to set and extract data to/from UI elements
- Monitors the change in UI elements and keeps track of it
- With 2-way binding data can flow from model to UI elements and vice versa

TYPES OF DATA BINDING

There are 3 different types of data binding (Property|Element|Aggregation)

PROPERTY BINDING

A UI element's property can be bound to a property from model e.g

```
<input value="{/firstName}">
<input value="{userModel>/firstName}">
<text visible="{= ${uiModel>deviceSize} === 's' ? false : true
```

ELEMENT BINDING (CONTEXT BINDING)

Allows to bind elements to specific object in model

```
1    var panel = this.byId("panel");
2    panel.bindElement("/accounts/0");
```

AGGREGATION BINDING (LIST BINDING)

Aggregation binding is useful when the control has multiple elements e.g VBox, List etc.

It requires model and template (representation for every element in list)

```
const accountItemTemplate = new sap.ui.core.ListItem({text
const accountsBox = new sap.m.VBOx({
  items: {
    path: "/accounts",
    template: oItemTemplate
  }
});
accountsBox.setModel({accounts: []});
```

AGGREGATION BINDING (LIST BINDING)

Aggregation binding is useful when the control has multiple elements e.g VBox, List etc.

It requires model and template (representation for every element in list)

```
const accountItemTemplate = new sap.ui.core.ListItem({text
const accountsBox = new sap.m.VBOx({
  items: {
    path: "/accounts",
    template: oItemTemplate
  }
});
accountsBox.setModel({accounts: []});
```

AGGREGATION BINDING (LIST BINDING)

Aggregation binding is useful when the control has multiple elements e.g VBox, List etc.

It requires model and template (representation for every element in list)

```
const accountItemTemplate = new sap.ui.core.ListItem({text
const accountsBox = new sap.m.VBOx({
  items: {
    path: "/accounts",
    template: oItemTemplate
  }
});
accountsBox.setModel({accounts: []});
```

FILTERING AND SORTING

UI5 provides convenient methods for filtering and sorting

- Filtering and sorting is triggered via binding
- Easy API’s for creating filters and sorter

Example of Filtering

```
1  // build filters
2  const filters = [];
3  filters.push(new Filter("ProductName",
4  FilterOperator.Contains, "iPhone"));
5
6  const productList = this.byId("productList");
7  const binding = productList.getBinding("items");
8
9   // filter bindings
10 binding.filter(filters);
```

Example of Filtering

```
1  // build filters
2  const filters = [];
3  filters.push(new Filter("ProductName",
4  FilterOperator.Contains, "iPhone"));
5
6  const productList = this.byId("productList");
7  const binding = productList.getBinding("items");
8
9   // filter bindings
10 binding.filter(filters);
```

Example of Filtering

```
// build filters
const filters = [];
filters.push(new Filter("ProductName",
FilterOperator.Contains, "iPhone"));

const productList = this.byId("productList");
const binding = productList.getBinding("items");

 // filter bindings
binding.filter(filters);
```

SORTING

Similar to filtering UI5 provides a way for sorting For OData connected model it can trigger backend filtering via OData calls

```
<list id="productList" class="sapUiResponsiveMargin" width="
        path : 'product>/Products',
        sorter : {
            path : 'ProductName'
        }
    }"></list>
```

ROUTING

*Routing is mechanism to manage views via a central component called "**Router**"*

Its modern front end development concept for building SPA (Single Page Applications)

Router Definition

```
{ // manifest.json
  ...
    "routing": {
      "config": {
        "routerClass": "sap.m.routing.Router",
        "viewType": "XML",
        "viewPath": "demoapp.view",
        "controlId": "app",
        "controlAggregation": "pages",
        "async": true
      },
      "routes": [
        {
          "pattern": "",
          "name": "overview",
```

Invoking navigation via router

```
sap.ui.require([
    "sap/ui/core/UIComponent",
], function(UIComponent {
    sap.ui.controller("Default", {
        onLinkPressed: function() {
            const oRouter = this.getOwnerComponent().getRou
            oRouter.navTo("product", {
                id: "5",
                productId: "3"
            });
        }
    });
});
```

Invoking navigation via router

```
sap.ui.require([
    "sap/ui/core/UIComponent",
], function(UIComponent {
    sap.ui.controller("Default", {
        onLinkPressed: function() {
            const oRouter = this.getOwnerComponent().getRou
            oRouter.navTo("product", {
                id: "5",
                productId: "3"
            });
        }
    });
});
```

Invoking navigation via router

```
sap.ui.require([
    "sap/ui/core/UIComponent",
], function(UIComponent {
    sap.ui.controller("Default", {
        onLinkPressed: function() {
            const oRouter = this.getOwnerComponent().getRou
            oRouter.navTo("product", {
                id: "5",
                productId: "3"
            });
        }
    });
});
```

Receiving router's navigation event

```
sap.ui.controller("Product", {
    onInit: function() {
      const oRouter = this.getOwnerComponent().getRouter();
      oRouter.getRoute("product")
      .attachMatched((oEvent) => {
        const productId = oEvent.getParameter("arguments").
        this.showProductDetails(productId)
      });
    },

    showProductDetails : function(productId) {
        //implementation
    }
}
```

Receiving router's navigation event

```
sap.ui.controller("Product", {
    onInit: function() {
      const oRouter = this.getOwnerComponent().getRouter();
      oRouter.getRoute("product")
      .attachMatched((oEvent) => {
        const productId = oEvent.getParameter("arguments").
        this.showProductDetails(productId)
      });
    },

    showProductDetails : function(productId) {
        //implementation
    }
}
```

Receiving router's navigation event

```
sap.ui.controller("Product", {
    onInit: function() {
      const oRouter = this.getOwnerComponent().getRouter();
      oRouter.getRoute("product")
      .attachMatched((oEvent) => {
        const productId = oEvent.getParameter("arguments").
        this.showProductDetails(productId)
      });
    },

    showProductDetails : function(productId) {
        //implementation
    }
}
```

RESPONSIVENESS

UI5 is responsive framework, it has responsive controls which can automatically scale to device form factors. sap.m library have all mobile friendly controls which works well on desktop e.g sap.m.Popover, sap.m.Table

Single code base for all form factors

Responsive Table Example

Desktop

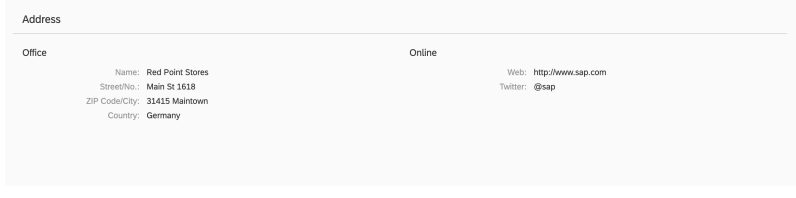
Product	Supplier	Dimensions	Weight	Price
10" Portable DVD player HT-2001	Titanium	24 x 19.5 x 29 cm	**0.84** KG	**449.99** EUR
7" Widescreen Portable DVD Player w MP3 HT-2000	Titanium	21.4 x 19 x 27.6 cm	**0.79** KG	**249.99** EUR
Astro Laptop 1516 HT-1251	Ultrasonic United	30 x 18 x 3 cm	**4.2** KG	**989.00** EUR
Astro Phone 6 HT-1252	Ultrasonic United	8 x 6 x 1.5 cm	**0.75** KG	**649.00** EUR
Audio/Video Cable Kit - 4m HT-2026	Titanium	21 x 10.2 x 13 cm	**0.2** KG	**29.99** EUR
Beam Breaker B-1 HT-6100	Titanium	30.4 x 23.1 x 23 cm	**1.7** KG	**469.00** EUR

Mobile

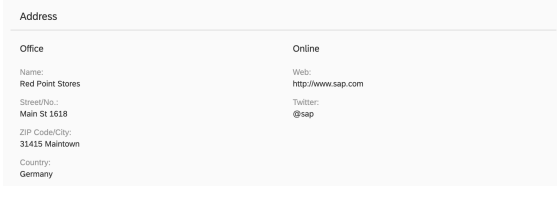
Product	Price
10" Portable DVD player HT-2001 Supplier: Titanium Dimensions: 24 x 19.5 x 29 cm Weight: **0.84** KG	**449.99** EUR
7" Widescreen Portable DVD Player w MP3 HT-2000 Supplier: Titanium Dimensions: 21.4 x 19 x 27.6 cm Weight: **0.79** KG	**249.99** EUR
Astro Laptop 1516 HT-1251 Supplier: Ultrasonic United Dimensions: 30 x 18 x 3 cm Weight: **4.2** KG	**989.00** EUR

Responsive Form Example

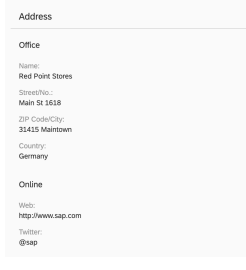
Desktop



Tablet



Mobile



Simple form XML Example

```
<core:fragmentdefinition xmlns="sap.m" xmlns:l="sap.ui.layout"
    <vbox class="sapUiSmallMargin">
        <f:simpleform id="SimpleFormDisplay480_12120Dual" edit
            <f:content>
                <core:title text="Office">
                <label text="Name">
                <text text="{SupplierName}">
                <label text="Street/No.">
                <text text="{Street} {HouseNumber}">
                <label text="ZIP Code/City">
                <text text="{ZIPCode} {City}">
                <label text="Country">
                <text text="{Country}">
                <core:title text="Online">
                <label text="Web">
```

MARGINS AND PADDINGS

"Use standard margin and padding classes"

sapUiTinyMargin, sapUiTinyMarginTop, sapUiTinyMarginBeginEnd, sapUiResponsiveMargin, sapUiContentPadding

Predefined Margin and Padding Classes

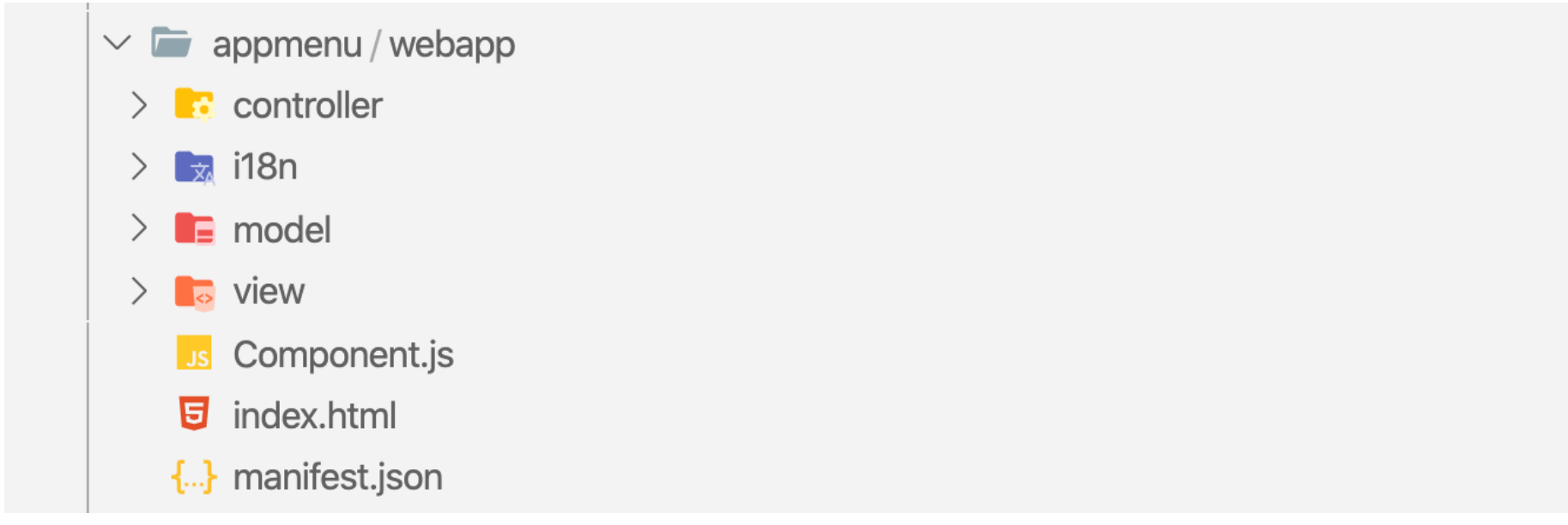
DEVELOPING NEW UI5 APPLICATION

How to start developing new UI5 app

Template Apps

COMPONENT DEVELOPMENT

Structure of Component



Component.js File

```
sap.ui.define([
  "sap/ui/core/UIComponent"
], function (UIComponent) {
  "use strict";
  const Component = UIComponent.extend("Component", {
    constructor: function (sId, mSettings) {
      UIComponent.apply(this, arguments);
    },
    metadata: {
      manifest: "json"
    },
    init: () => {},
    createContent: () => {}
  });
  return Component;
```

Component.js File

```
    "sap/ui/core/UIComponent"
], function (UIComponent) {
    "use strict";
    const Component = UIComponent.extend("Component", {
        constructor: function (sId, mSettings) {
            UIComponent.apply(this, arguments);
        },
        metadata: {
            manifest: "json"
        },
        init: () => {},
        createContent: () => {}
    });
    return Component;
}, true /* bExport */ );
```

Component.js File

```
sap.ui.define([
  "sap/ui/core/UIComponent"
], function (UIComponent) {
  "use strict";
  const Component = UIComponent.extend("Component", {
    constructor: function (sId, mSettings) {
      UIComponent.apply(this, arguments);
    },
    metadata: {
      manifest: "json"
    },
    init: () => {},
    createContent: () => {}
  });
  return Component;
```

Component.js File

```
    "sap/ui/core/UIComponent"
], function (UIComponent) {
    "use strict";
    const Component = UIComponent.extend("Component", {
        constructor: function (sId, mSettings) {
            UIComponent.apply(this, arguments);
        },
        metadata: {
            manifest: "json"
        },
        init: () => {},
        createContent: () => {}
    });
    return Component;
}, true /* bExport */ );
```

Component.js File

```
    "sap/ui/core/UIComponent"
], function (UIComponent) {
    "use strict";
    const Component = UIComponent.extend("Component", {
        constructor: function (sId, mSettings) {
            UIComponent.apply(this, arguments);
        },
        metadata: {
            manifest: "json"
        },
        init: () => {},
        createContent: () => {}
    });
    return Component;
}, true /* bExport */ );
```

MANIFEST

The setting for component are stored in special file called manifest.

```
{
    "_version": "1.8.0",
    "sap.app": {
        "id": "common.appmenu",
        "type": "application",
        "i18n": "i18n/i18n.properties",
        "applicationVersion": {
            "version": "1.0.0"
        }
    },

    "sap.ui": {
        "technology": "UI5",
        "icons": {
            "icon": "",
```

LOADING COMPONENTS

Asynchronously loading

```
sap.ui.component({
        name: "demo.product",
        async: true
  })
.then(function(oComponent) {
    const oContainer = new sap.ui.core.ComponentContainer({
      component: oComponent
    });
});
```

Asynchronously loading using component container

```
const oContainer = new sap.ui.core.ComponentContainer({
  name: "demo.product",
  async: true
});
oContainer.placeAt("target");
```

COMPONENT STATE

Component will have state which is stored in component properties

```
sap.ui.define([
  "sap/ui/core/UIComponent"
], function (UIComponent) {
  "use strict";
  const Component = UIComponent.extend("Component", {
    metadata: {
      manifest: "json",
        "properties": {
            "productId": {
              "type":"string",
              "defaultValue":"Product"
            },
            "productName": {
              "type":"string",
              "defaultValue":"Credit Card"
```

COMPONENT STATE

Component will have state which is stored in component properties

```
], function (UIComponent) {
  "use strict";
  const Component = UIComponent.extend("Component", {
    metadata: {
      manifest: "json",
        "properties": {
            "productId": {
              "type":"string",
              "defaultValue":"Product"
            },
            "productName": {
              "type":"string",
              "defaultValue":"Credit Card"
            }
        }
    }
```

COMPONENT STATE

Component will have state which is stored in component properties

```
], function (UIComponent) {
  "use strict";
  const Component = UIComponent.extend("Component", {
    metadata: {
      manifest: "json",
        "properties": {
            "productId": {
              "type":"string",
              "defaultValue":"Product"
            },
            "productName": {
              "type":"string",
              "defaultValue":"Credit Card"
            }
        }
    }
```

COMPONENT EVENTS

Components can expose some events from its internal layer

```
sap.ui.define([
  "sap/ui/core/UIComponent"
], function (UIComponent) {
  "use strict";
  const Component = UIComponent.extend("Component", {
    metadata: {
      manifest: "json",
        "events": {
            "productLoaded": {},
            "error": {},
            "change": {}
        }
    }
  });
  return Component;
```

COMPONENT EVENTS

Components can expose some events from its internal layer

```
2    "sap/ui/core/UIComponent"
3  ], function (UIComponent) {
4    "use strict";
5    const Component = UIComponent.extend("Component", {
6      metadata: {
7        manifest: "json",
8          "events": {
9              "productLoaded": {},
10             "error": {},
11             "change": {}
12         }
13     }
14   });
15   return Component;
16 }, true /* bExport */ );
```

COMPONENT EVENTS

Components can expose some events from its internal layer

```
], function (UIComponent) {
  "use strict";
  const Component = UIComponent.extend("Component", {
    metadata: {
      manifest: "json",
        "events": {
            "productLoaded": {},
            "error": {},
            "change": {}
        }
    }
  });
  return Component;
}, true /* bExport */ );

```

COMPONENT EVENTS

Components can expose some events from its internal layer

```
4    "use strict";
5    const Component = UIComponent.extend("Component", {
6      metadata: {
7        manifest: "json",
8          "events": {
9              "productLoaded": {},
10             "error": {},
11             "change": {}
12         }
13     }
14   });
15   return Component;
16 }, true /* bExport */ );
17
18
```

COMPONENT EVENTS

Components can expose some events from its internal layer

```
14    });
15    return Component;
16 }, true /* bExport */ );
17
18
19 sap.ui.component({
20          name: "demo.product",
21          async: true
22    })
23 .then(function(oComponent) {
24       // Attach events
25       oComponent.attachProductLoaded(() => {});
26       oComponent.attachError(() => {});
27       oComponent.attachChange(() => {});
28 });
```

THEMING

UI5 have standard themes as well as we can create our own themes

- UI5 uses LESS and CSS to maintain the stylesheet classes for its controls
- Maintainable and modular structure for all UI5 controls
- LESS files are compiled to CSS

How to design new theme?

- Adapt the existing theme using theme designer
- Create new theme using theme designer
- Adapt the existing theme by custom.css

Theme Designer

LIBRARY

Library is organized group of common controls, We can create your own library using framework

```
sap.ui.define(function() {

“use strict”;

var common = sap.ui.getCore().initLibrary({


// library version and additional params

});


// Common interfaces, types to be exposed

common.ValueColor = {
```

PERFORMANCE

> *"Performance is critical for web applications"*

As framework UI5 supports ways to improve it

LAZY INITIALIZATION

```
sap.ui.require(['load/my/dependency'], function(Dependency) {
  new Dependency().doSomething(); // Lazily load and execute
});
```

Useful in cases where views do not need such functions, controls all upfront e.g dialogs

DEFINE PATTERN

```
sap.ui.define(['./Helper', 'sap/m/Bar'], function(Helper,Bar

});
```

PRELOADS

Preloads are the packaged files optimized by build tool, Every component/library should have its own preload file

CHECKLIST

A UI5 performance checklist

DEBUGGING

- You can debug the UI5 applications using **Chrome Dev Tool**
- Use chrome's console, source, inspect, network tab to further debug
- **Ctrl + shift + alt + p** to open Technical info dialog
- Use Support Assistant to analyse the application
- Chrome plugin for UI5, helps in finding current state of controls

RESOURCES

- DemoKit
- Documentation
- API's
- Controls
- Icons
- Best Practices

DEMO

Weather Application