# SAP UI5 Interview Preparation Guide

## From Basics to 1 Year Experience

## Table of Contents

## 1. SAP UI5 Fundamentals

### What is SAP UI5?

SAP UI5 is an HTML5-based JavaScript framework for building enterprise-ready web applications with a responsive user interface. It follows the Model-View-Controller (MVC) pattern and provides a rich set of UI controls.

### Key Features

- **Responsive Design**: Works across devices (desktop, tablet, mobile)

- **Extensibility**: Can extend standard controls

- **Theming**: Built-in themes (SAP Fiori, Belize, Quartz, Horizon)

- **Data Binding**: Two-way, one-way, and one-time binding

- **i18n Support**: Internationalization and localization

- **Open Source**: OpenUI5 is the open-source version

### SAP UI5 vs OpenUI5

- **SAP UI5**: Commercial version with additional controls and libraries

- **OpenUI5**: Free, open-source version with core functionality

### Basic Project Structure

```
webapp/
 ├── Component.js        # Application component
 ├── manifest.json       # Application descriptor
 ├── index.html          # Entry point
 ├── controller/         # Controllers
 │   └── App.controller.js
 ├── view/               # Views
 │   └── App.view.xml
 ├── model/              # Models (if any)
 ├── i18n/               # Translation files
 │   └── i18n.properties
 ├── css/                # Custom styles
 └── util/               # Helper functions
```

### Component.js Example

```javascript
sap.ui.define([
    "sap/ui/core/UIComponent",
    "sap/ui/model/json/JSONModel"
], function (UIComponent, JSONModel) {
    "use strict";

    return UIComponent.extend("com.myapp.Component", {
        metadata: {
            manifest: "json"
        },

        init: function () {
            // Call the base component's init function
            UIComponent.prototype.init.apply(this, arguments);

            // Create the views based on the url/hash
            this.getRouter().initialize();

            // Set device model
            var oDeviceModel = new JSONModel(sap.ui.Device);
            oDeviceModel.setDefaultBindingMode("OneWay");
            this.setModel(oDeviceModel, "device");
        }
    });
});
```

## 2. MVC Architecture

### What is MVC in UI5?

MVC separates application logic into three interconnected components:

- **Model**: Manages data and business logic

- **View**: Displays data (XML, HTML, JSON, JavaScript)

- **Controller**: Handles user input and updates model

### View Types

#### XML View (Most Common)

```xml
<!-- App.view.xml -->
<mvc:View
  controllerName="com.myapp.controller.App"
  xmlns:mvc="sap.ui.core.mvc"
  xmlns="sap.m">

  <Page title="My Application">
    <content>
      <Input
        value="{/userName}"
        placeholder="Enter your name"/>
      <Button
        text="Submit"
        press=".onSubmit"/>
```

            <Text text="Hello, {/userName}!"/>
        </content>
    </Page>
</mvc:View>
~~~

Controller
```javascript
// App.controller.js
sap.ui.define([
    "sap/ui/core/mvc/Controller",
    "sap/ui/model/json/JSONModel"
], function (Controller, JSONModel) {
    "use strict";

    return Controller.extend("com.myapp.controller.App", {

        onInit: function () {
            // Initialize model
            var oModel = new JSONModel({
                userName: ""
            });
            this.getView().setModel(oModel);
        },

        onSubmit: function () {
            var sUserName = this.getView().getModel().getProperty("/userName");
            sap.m.MessageToast.show("Hello, " + sUserName);
~~~

```
      }
   });
});
```

### Controller Lifecycle Methods

```javascript
onInit: function() {
   // Called when controller is instantiated
},

onBeforeRendering: function() {
   // Called before the view is re-rendered
},

onAfterRendering: function() {
   // Called after the view has been rendered
},

onExit: function() {
   // Called when controller is destroyed
}
```

## 3. Data Binding

### Types of Data Binding

#### 1. One-Way Binding

Data flows from model to view only.

```xml
<Text text="{/productName}"/>
```

#### 2. Two-Way Binding

Data flows in both directions (default for input controls).

```xml
<Input value="{/userName}"/>
```

#### 3. One-Time Binding

Data is bound only once at initialization.

```xml
<Text text="{path: '/productName', mode: 'OneTime'}"/>
```

### Binding Modes

```javascript
// In controller or component
var oModel = new JSONModel();
oModel.setDefaultBindingMode("TwoWay"); // or "OneWay", "OneTime"
this.getView().setModel(oModel);
```

### Property Binding

```xml
<!-- Simple property binding -->
<Text text="{/firstName}"/>

<!-- With formatter -->
<Text text="{
   path: '/price',
   formatter: '.formatPrice'
}"/>
```

### Aggregation Binding

```xml
<List items="{/products}">
   <StandardListItem
      title="{name}"
      description="{description}"
      info="{price}"/>
</List>
```

### Element Binding

```xml
<Panel headerText="Product Details"
      binding="{/products/0}">
   <Text text="{name}"/>
   <Text text="{price}"/>
```

```
</Panel>
```

### Expression Binding

```xml
<Text
  text="{= ${/price} > 100 ? 'Expensive' : 'Affordable' }"
  visible="{= ${/stock} > 0 }"/>
```

### Context Binding

```javascript
// In controller
var oContext = oModel.createBindingContext("/products/0");
this.getView().setBindingContext(oContext);
```

## 4. Controls and Libraries

### Common Libraries

#### sap.m (Mobile Library)

Most commonly used library for responsive applications.

```xml
xmlns:m="sap.m"
```

```
<!-- Common Controls -->
<m:Button text="Click Me" press=".onPress"/>
<m:Input value="{/name}" placeholder="Enter name"/>
<m:Text text="Static Text"/>
<m:List items="{/items}">
  <m:StandardListItem title="{title}"/>
</m:List>
<m:Table items="{/products}">
  <m:columns>
    <m:Column><m:Text text="Name"/></m:Column>
    <m:Column><m:Text text="Price"/></m:Column>
  </m:columns>
  <m:items>
    <m:ColumnListItem>
      <m:Text text="{name}"/>
      <m:Text text="{price}"/>
    </m:ColumnListItem>
  </m:items>
</m:Table>
```

#### sap.ui.layout

```xml
xmlns:layout="sap.ui.layout"

<layout:VerticalLayout>
  <m:Text text="Item 1"/>
  <m:Text text="Item 2"/>
</layout:VerticalLayout>

<layout:HorizontalLayout>
  <m:Button text="Left"/>
  <m:Button text="Right"/>
</layout:HorizontalLayout>

<layout:Grid>
  <m:Panel>Panel 1</m:Panel>
  <m:Panel>Panel 2</m:Panel>
</layout:Grid>
```

#### sap.ui.table (Desktop Table)

For large datasets with scrolling.

```javascript
var oTable = new sap.ui.table.Table({
  visibleRowCount: 10,
  selectionMode: "Single",
  columns: [
    new sap.ui.table.Column({
      label: new sap.m.Label({text: "Name"}),
      template: new sap.m.Text({text: "{name}"})
    })
  ]
});
```

### Important Controls

#### Input Controls

```xml
<Input value="{/text}" placeholder="Enter text"/>
<TextArea value="{/description}" rows="5"/>
<CheckBox selected="{/isActive}" text="Active"/>
<RadioButton selected="{/option}" text="Option 1"/>
<DatePicker value="{/date}"/>
<TimePicker value="{/time}"/>
<ComboBox
  items="{/countries}"
  selectedKey="{/selectedCountry}">
  <core:Item key="{code}" text="{name}"/>
</ComboBox>
<Select
  items="{/items}"
  selectedKey="{/selected}">
  <core:Item key="{key}" text="{text}"/>
</Select>
```

#### Display Controls

```xml
<Text text="Simple Text"/>
<Label text="Label:" labelFor="input1"/>
<Title text="Section Title" level="H2"/>
```

```
<ObjectHeader
  title="{/productName}"
  number="{/price}"
  numberUnit="USD"/>
```

#### Container Controls

```xml
<Page title="Page Title">
  <content>
    <!-- Content here -->
  </content>
</Page>

<Panel headerText="Panel Header">
  <content>
    <!-- Content here -->
  </content>
</Panel>

<IconTabBar>
  <items>
    <IconTabFilter text="Tab 1" key="tab1">
      <!-- Tab 1 content -->
    </IconTabFilter>
    <IconTabFilter text="Tab 2" key="tab2">
      <!-- Tab 2 content -->
    </IconTabFilter>
  </items>
</IconTabBar>
```

## 5. Routing and Navigation

### manifest.json Routing Configuration
```json
{
  "sap.ui5": {
    "routing": {
      "config": {
        "routerClass": "sap.m.routing.Router",
        "viewType": "XML",
        "viewPath": "com.myapp.view",
        "controlId": "app",
        "controlAggregation": "pages",
        "transition": "slide"
      },
      "routes": [
        {
          "pattern": "",
          "name": "home",
          "target": "home"
        },
        {
          "pattern": "detail/{productId}",
          "name": "detail",
          "target": "detail"
        }
      ],
      "targets": {
        "home": {
          "viewName": "Home",
          "viewLevel": 1
        },
        "detail": {
          "viewName": "Detail",
          "viewLevel": 2
        }
      }
    }
  }
}
```

### Navigation in Controller

```javascript
// Navigate to a route
this.getRouter().navTo("detail", {
  productId: "123"
});

// Navigate back
```

```javascript
this.getRouter().navTo("home");

// Or use history
var oHistory = sap.ui.core.routing.History.getInstance();
var sPreviousHash = oHistory.getPreviousHash();

if (sPreviousHash !== undefined) {
  window.history.go(-1);
} else {
  this.getRouter().navTo("home", {}, true);
}
```

### Route Pattern Matching

```javascript
// In controller's onInit
this.getRouter().getRoute("detail").attachPatternMatched(this._onObjectMatched, this);

_onObjectMatched: function (oEvent) {
  var sProductId = oEvent.getParameter("arguments").productId;
  // Bind view to the product
  this.getView().bindElement({
    path: "/Products('" + sProductId + "')",
    model: "odata"
  });
}
```

## 6. OData Services

### What is OData?

OData (Open Data Protocol) is a REST-based protocol for querying and updating data. SAP systems extensively use OData V2 and V4.

### OData Model Initialization

```javascript
// In Component.js or controller
var oModel = new sap.ui.model.odata.v2.ODataModel("/sap/opu/odata/sap/SERVICE_NAME/");
this.setModel(oModel, "odata");
```

### CRUD Operations

#### Read (GET)

```javascript
// Read single entity
this.getView().getModel("odata").read("/Products('1')", {
  success: function(oData) {
    console.log(oData);
  },
  error: function(oError) {
    console.error(oError);
  }
```

~~~~
});

// Read entity set with filters
this.getView().getModel("odata").read("/Products", {
  filters: [
    new sap.ui.model.Filter("Price", "GT", 100)
  ],
  success: function(oData) {
    console.log(oData.results);
  }
});
```

#### Create (POST)
```javascript
var oModel = this.getView().getModel("odata");
var oEntry = {
  ProductID: "123",
  ProductName: "New Product",
  Price: 50
};

oModel.create("/Products", oEntry, {
  success: function(oData) {
    sap.m.MessageToast.show("Product created");
  },
  error: function(oError) {
    sap.m.MessageBox.error("Error creating product");
~~~~

```
    }
});
```

#### Update (PUT/PATCH)

```javascript
var oModel = this.getView().getModel("odata");
var oEntry = {
    ProductName: "Updated Product",
    Price: 75
};

oModel.update("/Products('123')", oEntry, {
    success: function() {
        sap.m.MessageToast.show("Product updated");
    },
    error: function(oError) {
        sap.m.MessageBox.error("Error updating product");
    }
});
```

#### Delete (DELETE)

```javascript
var oModel = this.getView().getModel("odata");

oModel.remove("/Products('123')", {
    success: function() {
```

```
    sap.m.MessageToast.show("Product deleted");
  },
  error: function(oError) {
    sap.m.MessageBox.error("Error deleting product");
  }
});
```

### Batch Requests

```javascript
oModel.setUseBatch(true);
oModel.setDeferredGroups(["myGroup"]);

// Add changes to batch
oModel.create("/Products", oEntry1, {groupId: "myGroup"});
oModel.create("/Products", oEntry2, {groupId: "myGroup"});

// Submit batch
oModel.submitChanges({
  groupId: "myGroup",
  success: function() {
    sap.m.MessageToast.show("Batch successful");
  }
});
```

### URL Parameters and Filters

```javascript
```

```
// $filter
var aFilters = [
  new sap.ui.model.Filter("Category", "EQ", "Electronics")
];

// $expand
this.getView().bindElement({
  path: "/Products('1')",
  parameters: {
    expand: "Supplier,Category"
  }
});

// $select
oModel.read("/Products", {
  urlParameters: {
    "$select": "ProductID,ProductName,Price"
  }
});

// $top and $skip (pagination)
oModel.read("/Products", {
  urlParameters: {
    "$top": 10,
    "$skip": 20
  }
});
```

## 7. Manifest.json

### What is manifest.json?

The application descriptor file that contains all application metadata and configuration.

### Complete Example

```json
{
  "_version": "1.12.0",
  "sap.app": {
    "id": "com.myapp",
    "type": "application",
    "i18n": "i18n/i18n.properties",
    "title": "{{appTitle}}",
    "description": "{{appDescription}}",
    "applicationVersion": {
      "version": "1.0.0"
    },
    "dataSources": {
      "mainService": {
        "uri": "/sap/opu/odata/sap/ZPRODUCT_SRV/",
        "type": "OData",
        "settings": {
          "odataVersion": "2.0",
          "localUri": "localService/metadata.xml"
        }
      }
    }
  },
  "sap.ui": {
    "technology": "UI5",
    "deviceTypes": {
      "desktop": true,
      "tablet": true,
      "phone": true
    }
  },
  "sap.ui5": {
    "rootView": {
      "viewName": "com.myapp.view.App",
      "type": "XML",
      "id": "app"
    },
    "dependencies": {
      "minUI5Version": "1.120.0",
      "libs": {
        "sap.ui.core": {},
        "sap.m": {},
        "sap.ui.layout": {}
      }
    },
    "models": {
      "i18n": {
        "type": "sap.ui.model.resource.ResourceModel",
        "settings": {
          "bundleName": "com.myapp.i18n.i18n"
        }
      },
      "": {
        "dataSource": "mainService",
        "preload": true,
        "settings": {
          "defaultBindingMode": "TwoWay",
          "defaultCountMode": "Inline"
        }
      }
    },
    "routing": {
      "config": {
        "routerClass": "sap.m.routing.Router",
        "viewType": "XML",
        "viewPath": "com.myapp.view",
        "controlId": "app",
        "controlAggregation": "pages"
      },
      "routes": [],
      "targets": {}
    }
  }
}
```

### Key Sections

#### sap.app

- Application metadata

- Data source definitions

- i18n configuration

#### sap.ui

- Technology and device support

#### sap.ui5

- Root view

- Dependencies

- Models

- Routing

## 8. Fragments and Dialogs

### What are Fragments?

Reusable UI parts that can be included in views. They don't have their own controller.

### Fragment Definition (XML)

```xml
<!-- ProductDialog.fragment.xml -->
<core:FragmentDefinition
```

```
  xmlns="sap.m"
  xmlns:core="sap.ui.core">

  <Dialog
    title="Product Details"
    contentWidth="400px">
    <content>
      <VBox>
        <Label text="Product Name"/>
        <Input value="{/productName}"/>
        <Label text="Price"/>
        <Input value="{/price}"/>
      </VBox>
    </content>
    <beginButton>
      <Button text="Save" press=".onSave"/>
    </beginButton>
    <endButton>
      <Button text="Cancel" press=".onCancel"/>
    </endButton>
  </Dialog>
</core:FragmentDefinition>
```

### Loading Fragment in Controller

```javascript
sap.ui.define([
  "sap/ui/core/mvc/Controller",
  "sap/ui/core/Fragment"
], function (Controller, Fragment) {
  "use strict";

  return Controller.extend("com.myapp.controller.Main", {

    onOpenDialog: function () {
      if (!this._oDialog) {
        Fragment.load({
          id: this.getView().getId(),
          name: "com.myapp.view.ProductDialog",
          controller: this
        }).then(function (oDialog) {
          this._oDialog = oDialog;
          this.getView().addDependent(this._oDialog);
          this._oDialog.open();
        }.bind(this));
      } else {
        this._oDialog.open();
      }
    },

    onSave: function () {
      // Handle save logic
      this._oDialog.close();
    },

    onCancel: function () {
            this._oDialog.close();
        },

        onExit: function () {
            if (this._oDialog) {
                this._oDialog.destroy();
            }
        }
    });
});
```

### Message Box

```javascript
// Simple message
sap.m.MessageBox.show("This is a message");

// Confirmation dialog
sap.m.MessageBox.confirm("Are you sure?", {
    onClose: function (oAction) {
        if (oAction === sap.m.MessageBox.Action.OK) {
            // User clicked OK
        }
    }
});

// Error message
sap.m.MessageBox.error("An error occurred");
```

```
// Warning
sap.m.MessageBox.warning("This is a warning");

// Information
sap.m.MessageBox.information("Information message");

// Success
sap.m.MessageBox.success("Operation successful");
```

### Message Toast

```javascript
sap.m.MessageToast.show("Quick message", {
  duration: 3000,
  width: "15em"
});
```

## 9. Formatters and Custom Logic

### Formatters in View

```xml
<Text text="{
  path: 'price',
  formatter: '.formatPrice'
}"/>

<Text text="{
   parts: ['firstName', 'lastName'],
   formatter: '.formatFullName'
}"/>
```

### Formatter Functions

```javascript
// In controller
formatPrice: function (sPrice) {
   if (!sPrice) {
      return "";
   }
   return parseFloat(sPrice).toFixed(2) + " USD";
},

formatFullName: function (sFirstName, sLastName) {
   return sFirstName + " " + sLastName;
},

formatDate: function (oDate) {
   if (!oDate) {
      return "";
   }
   var oDateFormat = sap.ui.core.format.DateFormat.getDateInstance({
      pattern: "dd/MM/yyyy"
```

```
    });
    return oDateFormat.format(new Date(oDate));
},

formatStatus: function (sStatus) {
    switch (sStatus) {
        case "A":
            return "Active";
        case "I":
            return "Inactive";
        default:
            return "Unknown";
    }
}
```

### Separate Formatter File

```javascript
// util/formatter.js
sap.ui.define([], function () {
    "use strict";

    return {
        formatPrice: function (sPrice) {
            if (!sPrice) {
                return "";
            }
            return parseFloat(sPrice).toFixed(2) + " USD";
        },

        statusText: function (sStatus) {
            var oResourceBundle = this.getView().getModel("i18n").getResourceBundle();
            switch (sStatus) {
                case "A":
                    return oResourceBundle.getText("statusActive");
                case "I":
                    return oResourceBundle.getText("statusInactive");
                default:
                    return "";
            }
        }
    };
});
```

### Using Formatter in Controller

```javascript
sap.ui.define([
    "sap/ui/core/mvc/Controller",
    "com/myapp/util/formatter"
], function (Controller, formatter) {
    "use strict";

    return Controller.extend("com.myapp.controller.Main", {
        formatter: formatter,
        // Rest of controller code
    });
});
```

### Custom Validation

```javascript
onValidateInput: function (oEvent) {
    var oInput = oEvent.getSource();
    var sValue = oInput.getValue();

    if (!sValue || sValue.length < 3) {
        oInput.setValueState("Error");
        oInput.setValueStateText("Minimum 3 characters required");
    } else {
        oInput.setValueState("None");
    }
}
```

## 10. Performance Optimization

### Best Practices

#### 1. Use OData Select and Expand

```javascript
```

// Bad - Fetches all fields
oModel.read("/Products");

// Good - Fetches only required fields
oModel.read("/Products", {
   urlParameters: {
      "$select": "ProductID,ProductName,Price"
   }
});

#### 2. Batch Requests

```javascript
oModel.setUseBatch(true);
```

#### 3. Lazy Loading

```javascript
// Use growing="true" for lists
<List
   items="{/Products}"
   growing="true"
   growingThreshold="20">
```

#### 4. Destroy Unused Objects

```javascript
onExit: function () {
  if (this._oDialog) {
    this._oDialog.destroy();
  }
}
```

#### 5. Use Expression Binding

```xml
<!-- Instead of formatter for simple conditions -->
<Text visible="{= ${stock} > 0 }"/>
```

#### 6. Debouncing Search

```javascript
onSearch: function (oEvent) {
  clearTimeout(this._searchTimeout);
  this._searchTimeout = setTimeout(function () {
    var sQuery = oEvent.getParameter("query");
    // Perform search
  }, 300);
}
```

#### 7. Component Preload

```json
// In manifest.json
"sap.ui5": {
  "dependencies": {
      "libs": {
        "sap.m": {
          "lazy": false
        }
      }
  }
}
```

## 11. Common Interview Questions

### Basic Level

Q1: What is SAP UI5?

A: SAP UI5 is an HTML5-based JavaScript framework for building responsive, enterprise-ready web applications. It follows MVC architecture and provides rich UI controls.

Q2: What are the different view types in UI5?

A: XML (most common), JSON, HTML, JavaScript, and Typed Views.

Q3: What is data binding?

A: Data binding connects UI controls to data models, allowing automatic synchronization. Types: One-way, Two-way, One-time.

Q4: What is the difference between JSON Model and OData Model?

A:

- **JSON Model**: Client-side model for static or local data

- **OData Model**: Server-side model for REST-based services with CRUD operations

Q5: What is manifest.json?

A: Application descriptor file containing metadata, configuration, dependencies, data sources, and routing information.

Q6: What are fragments?

A: Reusable UI components without their own controller, used for dialogs, forms, or repeated UI parts.

Q7: Explain Component.js

A: Main application component that initializes the app, sets up models, and starts routing.

Q8: What is the purpose of i18n?

A: Internationalization - supporting multiple languages by externalizing text in property files.

Q9: What are the lifecycle methods of a controller?

A: onInit, onBeforeRendering, onAfterRendering, onExit

Q10: How do you navigate between views?

A: Using Router: `this.getRouter().navTo("routeName", {params});`

### Intermediate Level

Q11: What is aggregation binding?

A: Binding a collection of data to a control's aggregation (like items in a List or Table).

Q12: How do you implement filtering in OData?

A:

```javascript
var aFilters = [
   new sap.ui.model.Filter("Price", "GT", 100)
];
oModel.read("/Products", { filters: aFilters });
```

Q13: What is the difference between attachPress and press?

A:

- `press=".onPress"`: XML view event binding
- `attachPress`: Programmatic event attachment in controller

Q14: How to implement master-detail pattern?

A: Use routing with two targets, pass ID in URL pattern, bind detail view to selected item.

Q15: What is expression binding?

A: Inline JavaScript expressions in XML views:

```xml
<Text text="{= ${price} > 100 ? 'High' : 'Low' }"/>
```

Q16: How to handle errors in OData calls?

A:

```javascript
oModel.read("/Products", {
   success: function(oData) {},
   error: function(oError) {
      sap.m.MessageBox.error(oError.message);
   }
});
```

Q17: What is the difference between setModel and setProperty?

A:

- `setModel`: Sets entire model to view/component

- `setProperty`: Updates specific property in existing model

Q18: How to implement custom validation?

A: Use ValueState and ValueStateText properties on input controls based on validation logic.

Q19: What is batch processing in OData?

A: Grouping multiple OData operations into a single HTTP request to improve performance.

Q20: How to create a custom control?

A:

```javascript
sap.ui.define([
   "sap/ui/core/Control"
], function (Control) {
   return Control.extend("com.myapp.CustomControl", {
      metadata: {
         properties: {
            "text": { type: "string", defaultValue: "" }
      }
    },
    renderer: function (oRM, oControl) {
      oRM.write("<div");
      oRM.writeControlData(oControl);
      oRM.write(">");
      oRM.writeEscaped(oControl.getText());
      oRM.write("</div>");
    }
  });
});
```

### Advanced Level

Q21: Explain event bus in UI5

A: Central event mechanism for cross-component communication:

```javascript
// Subscribe
sap.ui.getCore().getEventBus().subscribe("Channel", "Event", this.handler, this);

// Publish
sap.ui.getCore().getEventBus().publish("Channel", "Event", { data: value });
```

Q22: How to optimize OData performance?

A:

- Use $select to fetch only required fields

- Use $expand for related entities

- Enable batch requests

- Implement client-side filtering/sorting when possible

- Use growing lists for large datasets

Q23: What is Smart Controls?

A: Controls that automatically configure themselves based on OData metadata (SmartTable, SmartForm, SmartFilterBar).

Q24: How to implement deep insert in OData?

A:

```javascript
var oEntry = {
  ProductID: "1",
  ProductName: "Product",
  Supplier: {
    SupplierID: "S1",
    SupplierName: "Supplier"
  }
};
oModel.create("/Products", oEntry);
```

Q25: What is the difference between sap.m.Table and sap.ui.table.Table?

A:

- **sap.m.Table**: Responsive, mobile-optimized, all data loaded

- **sap.ui.table.Table**: Desktop-optimized, virtual scrolling for large datasets

Q26: How to implement draft handling?

A: Use OData V4 draft features or implement custom draft save/discard logic with separate entity sets.

Q27: What is metadata-driven development?

A: Using OData service metadata to automatically generate UI controls and validation rules.

Q28: How to handle concurrency in OData updates?

A: Use ETag for optimistic locking:

```javascript
oModel.update("/Products('1')", oEntry, {
  eTag: currentETag
});
```

Q29: Explain flexible column layout

A: SAP Fiori pattern with up to 3 columns (master-detail-detail) for complex navigation.

Q30: How to implement field help/value help?

A:

```xml
<Input
  showValueHelp="true"
  valueHelpRequest=".onValueHelp"/>
```

Then show a SelectDialog or TableSelectDialog in the handler.

## Additional Topics to Prepare

### 1. Debugging

- Chrome DevTools
- UI5 Diagnostics (Ctrl+Alt+Shift+S)
- Support Assistant
- Console logging

### 2. Testing

- QUnit for unit testing
- OPA5 for integration testing

### 3. Deployment

- SAP BTP (Business Technology Platform)
- SAP NetWeaver
- Standalone web server

### 4. Security

- CSRF tokens
- XSS prevention
- Authentication/Authorization

### 5. Fiori Design Guidelines

- SAP Fiori design principles
- Launchpad integration
- Tile configuration

## Practical Coding Exercises

### Exercise 1: Create a Product List with Search and Filter

Create a view with a list of products, search bar, and category filter.

### Exercise 2: Master-Detail Application

Implement a master-detail pattern with navigation.

### Exercise 3: Form with Validation

Create a form with input validation and error handling.

### Exercise 4: CRUD Operations

Implement Create, Read, Update, Delete operations with an OData service.

### Exercise 5: Custom Formatter

Create a formatter file with various formatting functions.

## Tips for Interview Success

1. **Understand the Basics**: Master MVC, data binding, and routing

2. **Practice Coding**: Build small applications

3. **Know OData**: Understand CRUD operations and URL parameters

4. **Study manifest.json**: Know its structure and importance

5. **Prepare Examples**: Have real project examples ready to discuss

6. **Stay Updated**: Know about latest UI5 features and versions

7. **Fiori Knowledge**: Understand Fiori design principles

8. **Problem-Solving**: Be ready to solve UI5-specific problems on the spot

---

## Resources for Further Learning

- Official SAP UI5 Documentation: https://ui5.sap.com/
- OpenUI5 Documentation: https://openui5.org/
- SAP Community: https://community.sap.com/
- UI5 Demo Kit: https://ui5.sap.com/test-resources/sap/m/demokit/
- GitHub OpenUI5 Repository: https://github.com/SAP/openui5

Good Luck with Your Interview!