

# WEEK 2 UNIT 3

## AUTOMATIC CONVERSION WITH DATA TYPES

Please perform the exercises below in your app project as shown in the video.

### Table of Contents

1	Usage of Data Types for Formatting .....	2
2	Usage of Types for Form Validation .....	3

### Preview

openSAP - Developing Web Apps with SAPUI5		
Getting Started	Containers and Layout	Data Binding
Products		
HT-1000	Notebook Basic 15	957.00 EUR deliver via parcel
HT-1001	Notebook Basic 17	1,249.00 EUR deliver via parcel
HT-1002	Notebook Basic 18	1,570.00 USD deliver via parcel
HT-1003	Notebook Basic 19	1,650.00 EUR deliver via parcel
HT-1007	ITelO Vault	299.00 USD deliver via mail
HT-1010	Notebook Professional 15	1,999.00 EUR deliver via parcel

Figure 1 - Preview of the app after doing this unit's exercises

# 1 USAGE OF DATA TYPES FOR FORMATTING

In this step, we use a data type to apply standard formatting across different currencies.

## Preview

HT-1010	Notebook Professional 15	€1,999.00 deliver via parcel
HT-1011	Notebook Professional 17	¥229,900 deliver via parcel
HT-1020	ITelO Vault Net	€459.00 deliver via mail
HT-1021	ITelO Vault SAT	MX\$149.00 deliver via mail

Figure 2 - Formatted currencies

## webapp/view/App.view.xml

```
<mvc:View ...>
  ...
  <ObjectListItem
    title="{Name}"
    number="{
      parts: [
        {path: 'Price'},
        {path: 'CurrencyCode'}
      ],
      type: 'sap.ui.model.type.Currency',
      formatOptions: {
        showMeasure: false
      }
    }"
    numberUnit="{CurrencyCode}"
  ...>
  ...
</mvc:View>
```

In this step, we format the price displayed in the `ObjectListItem` control depending on the currency by applying the `Currency` data type on the `number` attribute. This is achieved by setting the `type` attribute of the binding syntax to `sap.ui.model.type.Currency`.

Additionally, we set the formatting option `showMeasure` to `false`. This hides the currency code as it is already displayed in the `numberUnit` attribute.

As you can see above, we use a special binding syntax for the `number` property of the `ObjectListItem`. This binding syntax makes use of so-called "Calculated Fields", which allows to pass multiple properties from different models to a formatter function or data type. The properties bound from different models are called "parts".

We could access these two values in a custom formatter function to specify how they should be processed together. In our case, the `currency` type handles the formatting of the price. Please see the related documentation below for more details.

## 2 USAGE OF TYPES FOR FORM VALIDATION

In this step, we add a data type to the input control from one of the previous sessions and enable automatic form validation. To keep the exercise simple and avoid adding more controls, we use the input field for the name used in the “Hello” message, although it obviously was not originally meant to require a number, but a string.

### Preview

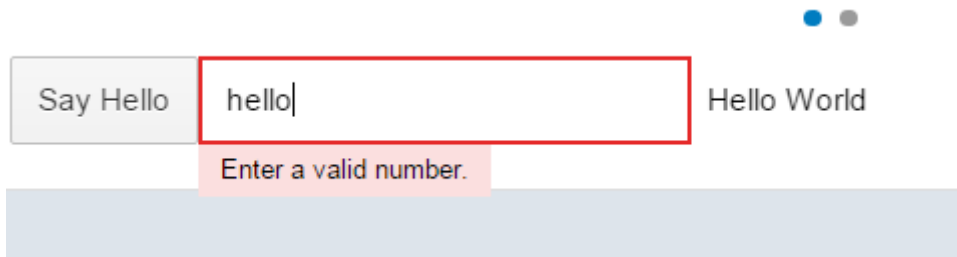


Figure 3 - Form validation based on types

### webapp/view/App.view.xml

```
<mvc:View ...>
  ...
  <IconTabFilter id="start" ...>
    ...
    <Input
      value="{
        path: 'helloPanel>/recipient/amount',
        type: 'sap.ui.model.type.Float',
        formatOptions: {minFractionDigits: 2},
        constraints: {maximum : 3000}
      }"
      description="Hello {helloPanel>/recipient/name}"
      valueLiveUpdate="false"
      width="60%"/>
    </IconTabFilter>
  ...
</mvc:View>
```

We will change the binding information to be able to add the type and additional format options and constraints to it. The way to add it is the same as in the list Item. We use the type `sap.ui.model.Type.Float` to display the entered number in a specific format. Additionally we add a constraint that has to be fulfilled and set the property `valueLiveUpdate` to `false` so that the model is not updated on every keystroke.

### webapp/manifest.json

```
{
  ...
  "sap.ui5": {
    "_version": "1.2.0",
    "rootView": {
      "viewName": "opensap.myapp.view.App",
      "type": "XML",
      "id": "app"
    },
    "handleValidation": true,
    "autoPrefixId": true,
    ...
  }
}
```

We now see automatic type conversion in the input field but not yet input validation in place. In order to achieve this, we will set automatic validation handling to true in the

`manifest.json` file and the SAPUI5 core will handle input validation based on format options and constraints on all input controls automatically.

Now run your app again and check if the type on the input field is working by entering a number higher than 3000 or a string. You should see a red border around the input field and if you focus it you will see a validation error message.

### Related Information

[Calculated Fields for Data Binding](#)

[Custom Formatter Functions](#)

[API Reference: `sap.ui.model.type`](#)

[API Reference: `sap.ui.model.type.Currency`](#)

[API Overview and Samples: `sap.ui.model.type.Currency`](#)

[API Overview and Samples: `sap.ui.model.type.Float`](#)

### Coding Samples

Any software coding or code lines/strings (“Code”) provided in this documentation are only examples and are not intended for use in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules for certain SAP coding. SAP does not warrant the correctness or completeness of the Code provided herein and SAP shall not be liable for errors or damages caused by use of the Code, except where such damages were caused by SAP with intent or with gross negligence.