



**BUILDING BLOCKS / UI Designer / UI component types**

# Contents

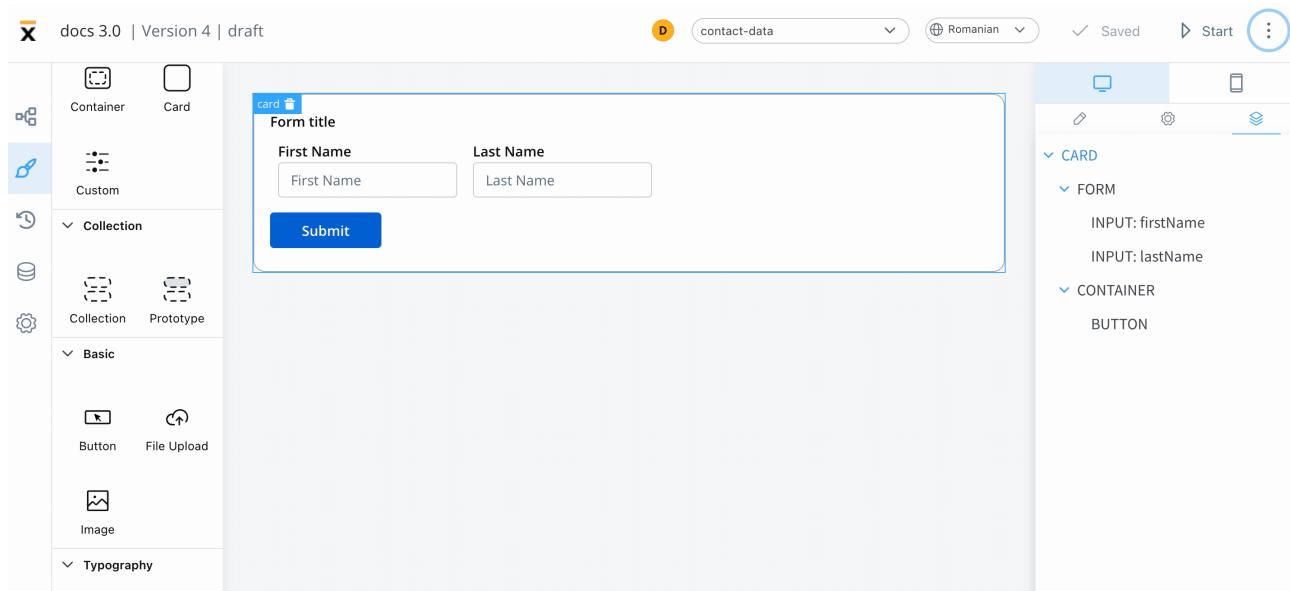
- BUILDING BLOCKS / UI Designer / UI component types / Root components / Container
- BUILDING BLOCKS / UI Designer / UI component types / Root components / Card
- BUILDING BLOCKS / UI Designer / UI component types / Root components / Custom
- BUILDING BLOCKS / UI Designer / UI component types / Collection / Collection Prototype
  - Description
  - Configurable properties:
  - Example
  - Adding elements with UI Actions
    - Step 1 - Defining the Node Action
    - Step 2 - Adding the Button & UI Action
    - Result
- BUILDING BLOCKS / UI Designer / UI component types / Buttons
  - Basic button
    - Configuring a basic button
    - Button styling
  - File upload
    - Configuring a file upload button
    - Button styling
- BUILDING BLOCKS / UI Designer / UI component types / File Preview
  - Configuring a File Preview element
    - File Preview properties (web)
    - File Preview properties (mobile)
    - File preview styling

- File Preview example
- BUILDING BLOCKS / UI Designer / UI component types / Form elements / Input
  - Configuring the input element
    - Input settings
    - Input styling
- BUILDING BLOCKS / UI Designer / UI component types / Form elements / Text area
  - Configuring the text area element
    - Text area settings
    - Text area styling
- BUILDING BLOCKS / UI Designer / UI component types / Form elements / Select
  - Configuring the Select element
    - Select Settings
    - Select styling
  - Example - Dynamic dropdowns
    - Creating the process
    - Configuring the nodes
    - Configuring the UI
- BUILDING BLOCKS / UI Designer / UI component types / Form elements / Checkbox
  - Configuring the checkbox element
    - Checkbox settings
    - Checkbox styling
- BUILDING BLOCKS / UI Designer / UI component types / Form elements / Radio
  - Configuring the radio field element
    - Radio settings

- Radio styling
- BUILDING BLOCKS / UI Designer / UI component types / Form elements / Switch
  - Configuring the switch element
    - Switch settings
- BUILDING BLOCKS / UI Designer / UI component types / Form elements / Datepicker
  - Configuring the datepicker element
    - Datepicker settings
    - Datepicker styling
- BUILDING BLOCKS / UI Designer / UI component types / Form elements / Slider
  - Configuring the slider element
    - Slider settings
    - Multiple sliders
    - Slider styling
- BUILDING BLOCKS / UI Designer / UI component types / Form elements / Segmented button
  - Configuring the segmented button
    - Segmented button settings
    - Segmented button styling
- BUILDING BLOCKS / UI Designer / UI component types / Image
  - Configuring an image
    - Image settings
    - Media library
    - Process Data
    - External
  - UI actions
  - Image styling

# BUILDING BLOCKS / UI Designer / UI component types / Root components / Container

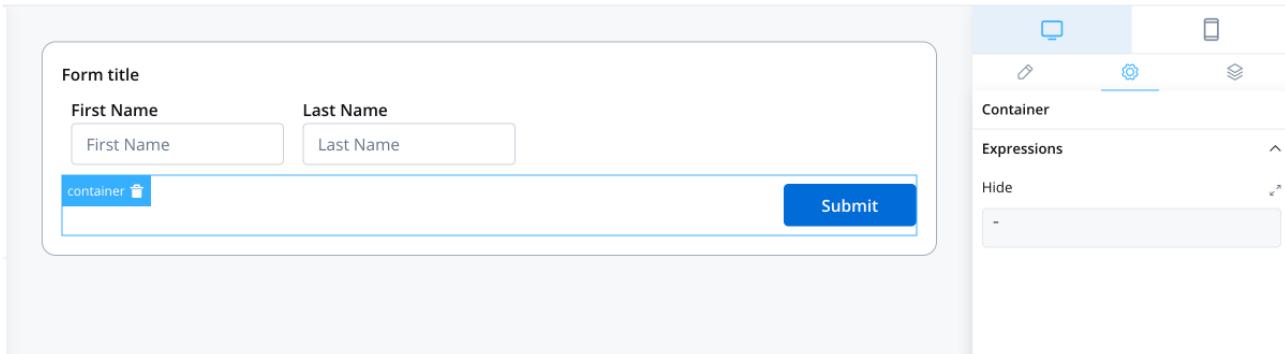
A container is a versatile element that allows you to group components and align them as desired.



The following properties can be configured in the container:

## Settings

- **Expressions (Hide)** - JavaScript expressions used to hide components when they evaluate to true



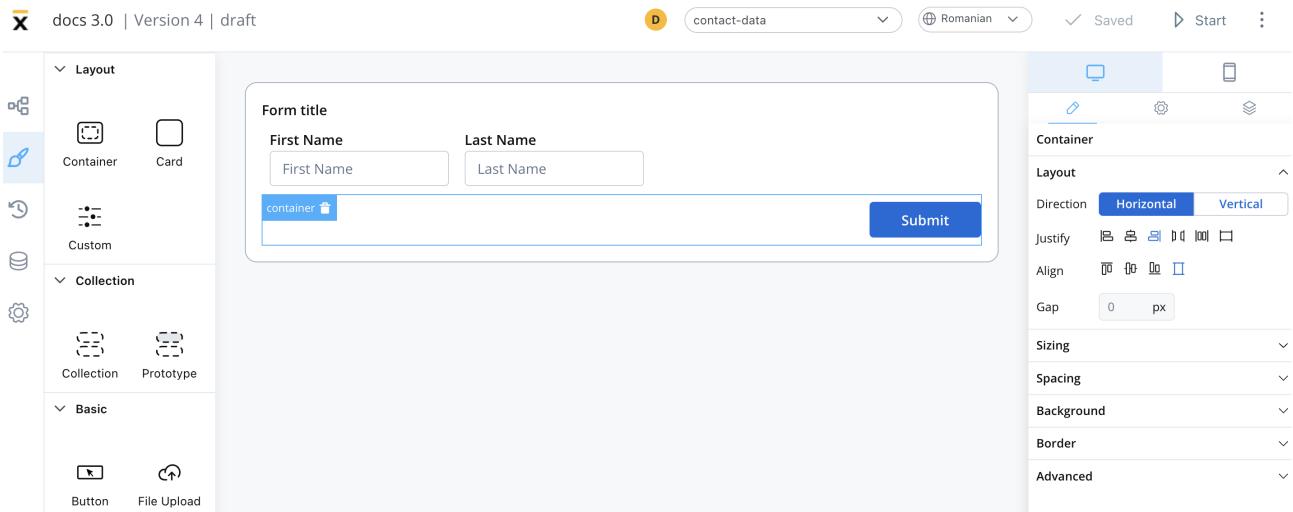
## Styling

- **Layout** - this property is available for components that group children and includes the following options:
  - Direction - Horizontal / Vertical (for example, select *Horizontal*)
  - Justify (H) - (for example, select *end*)
  - Align (V) - this option allows you to align components vertically
  - Gap - you can set the gap between components

More layout demos available below:

» [Layout Demos](#)

When you apply the above properties, you can generate the following output, with the button appearing on the right side of the container, underneath the form with three form elements:



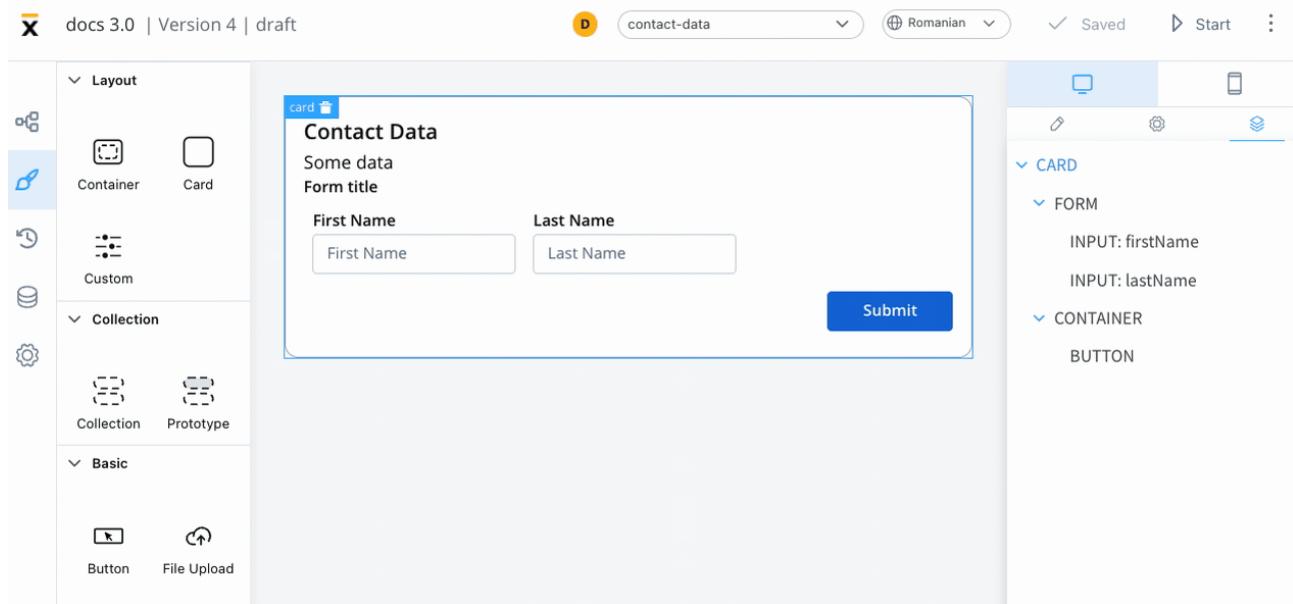
For more information about styling and layout configuration, check the following section:

» [UI Designer](#)

Was this page helpful?

# BUILDING BLOCKS / UI Designer / UI component types / Root components / Card

A card is a graphical component that allows grouping and alignment of other components. It can also include an accordion element for expanding and collapsing content.



The following properties that can be configured:

## Settings

- **Message** - a valid JSON that describes the data pushed to the frontend application when the process reaches a specific user task
- **Title** - the title of the card
- **Subtitle** - the subtitle of the card
- **Card style** - you can choose between a border or raised style
- **Has accordion?** - this feature allows you to add a Bootstrap accordion, which organizes content within collapsible items and displays only one collapsed item at a time

### ⚠ CAUTION

Accordion element is not available for mobile.

The screenshot shows the FLOWX.AI UI Designer interface with a Card component selected. At the top, there are icons for desktop and mobile device preview, a pencil for edit, a gear for settings, and a three-dot menu. Below these are sections for 'Card' and 'Message'. The 'Message' section contains a text input field with a placeholder '-' and a clear icon. Under the 'Properties' section, there are fields for 'Title' (placeholder 'eg. title') and 'Subtitle' (placeholder 'eg. subtitle'). A 'Card style' dropdown is set to 'border', with 'raised' as an alternative. A checkbox labeled 'Has Accordion' is also present.

## Styling

- **Layout** - This property is available for components that group children and includes the following options:
  - Direction - Horizontal / Vertical (for example, select *Vertical*)
  - Justify (H) - (for example, select *center*)
  - Align (V) - this option allows you to align components vertically
  - Gap - you can set the gap between components

More layout demos available below:

» [Layout Demos](#)

This example will generate a card with the following layout configuration:

The screenshot shows the FLOWX.AI UI Designer interface. On the left, there's a sidebar with categories like Layout, Collection, and Basic, each containing icons for Container, Card, Custom, Collection, Prototype, Button, File Upload, and Image. In the center, a card component is selected, displaying a form titled "Contact Data" with fields for "First Name" and "Last Name". A "Submit" button is at the bottom. To the right of the card, the "Layout" panel is open, showing the following configuration:

- Direction:** Horizontal (selected)
- Justify:** Center
- Align:** Center
- Gap:** 0 px
- Sizing:** Fit W: auto
- Spacing:** All values are 0

For more information about styling and layout configuration, check the following section:

### » UI Designer

## Validating elements

To validate all form elements under a card, you need to set the key of the form/element on the property of the button: *Forms To Validate*.

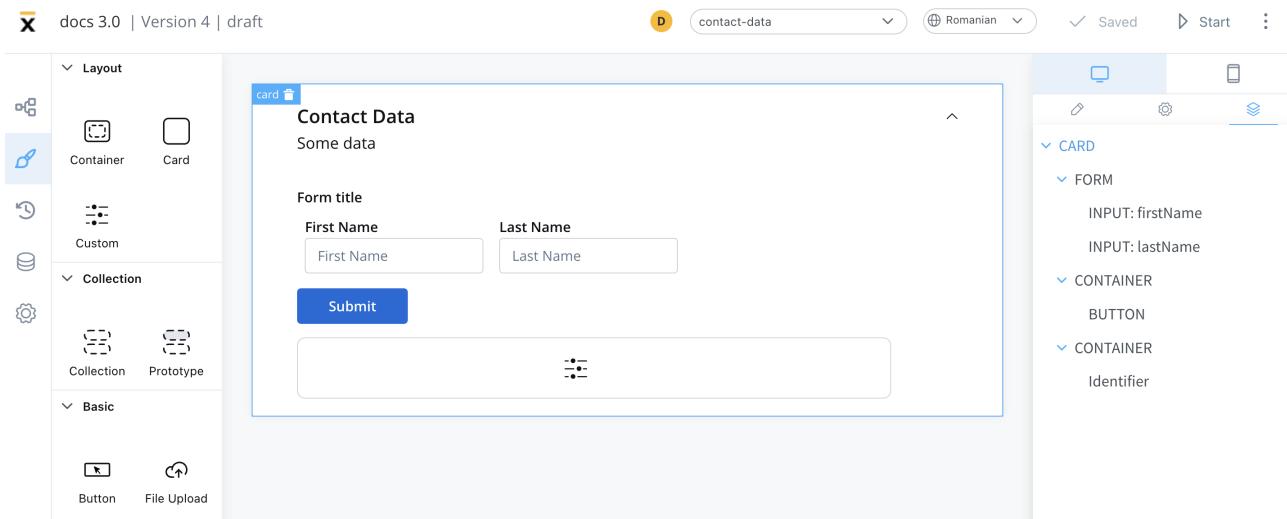
The screenshot shows the FLOWX.AI UI Designer interface. On the left, there's a sidebar with categories like Layout, Collection, Basic, and Typography, each containing icons for various components. The main workspace contains a card titled "Contact Data" with the sub-label "Some data". Inside the card, there are two input fields labeled "First Name" and "Last Name", and a blue button labeled "mit". The right side of the interface has a toolbar with device icons and a "Saved" status. Below the toolbar, there's a "Add action" panel with fields for "Event" (set to "CLICK"), "Action Type" (set to "ACTION"), and "Node Action Name" (set to "upload\_file"). There are also checkboxes for "Use a different name for UI action" and "Add custom body". At the bottom of this panel is a "Remove form to validate" link and a "Forms to validate" dropdown menu. The dropdown menu shows a hierarchical selection path: "Select forms to validate" > "CARD" > "FORM" > "CONTAINER". At the bottom right of the validation panel are "Cancel" and "Save" buttons.

Was this page helpful?

# BUILDING BLOCKS / UI Designer / UI component types / Root components / Custom

Custom components are developed in the web application and referenced here by component identifier. This will dictate where the component is displayed in the component hierarchy and what actions are available for the component.

To add a custom component in the template config tree, we need to know its unique identifier and the data it should receive from the process model.



The properties that can be configured are as follows:

- **Identifier** - this will enable the custom component to be displayed in the component hierarchy and what actions are available for the component
- **Input keys** - used to define the process model paths from which the components will receive their data

- **UI Actions** - actions defined here will be made available to the custom component

The screenshot displays a configuration interface for a custom UI component. At the top, there are icons for a desktop computer and a smartphone. Below these are two small icons: a gear and a stack of three squares. A blue horizontal line is positioned under the gear icon.

**Custom**

**Identifier** ↶ ↾

Identifier

**Input Keys** ^

CustomComponent -pencil icon trash icon

Add an option +

**UI Action** ^

Add UI action +

## Display of User Interface Elements

When a process instance is initiated, the web application receives all the UI elements that can be displayed in the process under the `templateConfig` key.

When a user task is reached in the process instance, the **events-gateway** receive requests, triggering it to display the associated UI element.

Example:

### 1. Starting a process:

- The following is an example of starting a process instance via a **POST** request to

```
{{processUrl}}/api/internal/process/DemoProcess/start:
```

```
{
  "processDefinitionName" : "DemoProcess",
  "tokens" : [ {
    "id" : 662631,
    "startNodeId" : null,
    "currentNodeId" : 662807,
    "currentnodeName" : null,
    "state" : "ACTIVE",
    "statusCurrentNode" : "ARRIVED",
    "dateUpdated" : "2023-02-09T12:23:19.464155Z",
    "uuid" : "ae626fda-8166-49e8-823b-fe24f36524a7"
  } ],
  "state" : "CREATED",
  "templateConfig" : [ {
    "id" : 630831,
    "flowxUuid" : "80ea0a85-2b0b-442a-a123-2480c7aa2dce",
    "nodeDefinitionId" : 662856,
    "uiElements" : [
      {
        "id" : 662856,
        "name" : "User Task 1"
      }
    ]
  } ]
}
```

```
"componentIdentifier" : "CONTAINER",
"type" : "FLOWX",
"order" : 1,
"canGoBack" : true,
"displayOptions" : {
  "flowxProps" : { },
  "style" : null,
  "flexLayout" : {
    "fxLayoutGap" : 0,
    "fxLayoutAlign" : "start stretch",
    "fxLayout" : "column"
  },
  "className" : null,
  "platform" : "DEFAULT"
},
"templateConfig" : [ {
  "id" : 630832,
  "flowxUuid" : "38e2c164-f8cd-4f6e-93c8-39b7cdd734cf",
  "nodeDefinitionId" : 662856,
  "uiTemplateParentId" : 630831,
  "componentIdentifier" : "TEXT",
  "type" : "FLOWX",
  "order" : 0,
  "key" : "",
  "canGoBack" : true,
  "displayOptions" : {
    "flowxProps" : {
      "text" : "Demo text"
    },
    "style" : null,
    "flexLayout" : null,
    "className" : null,
    "platform" : "DEFAULT"
  },
  "expressions" : {
}
```

```
        "hide" : """",
    },
    "templateConfig" : [ ],
    "dataSource" : {
        "processData" : {
            "parentFlowxUuid" : null
        },
        "nomenclator" : {
            "parentFlowxUuid" : null
        }
    }
}
],
{
    "uuid" : "d985d128-ae45-4408-a643-1dd026a644d3",
    "generalData" : null,
    "backCounter" : 0,
    "startedByActionId" : null,
    "subProcesses" : null,
    "subprocessesUuids" : null,
    "baseUrl" : null
}
```

2. The following is an example of a progress message:

```
{
    "progressUpdateDTO": {
        "processInstanceUuid": "5f24c66f-04a7-433a-b64a-
a765d3b8121a",
        "tokenUuid": "11c32ba6-b3e7-4267-9383-25d69b26492c",
        "currentNodeId": 662856
    }
}
```

3. **ProgressUpdateDto** will trigger the **SDK** to search for the UI element having the same **nodeId** as the one from the web socket progress event
4. Additionally, it will ask for data and actions that are required for this component via a GET request `{{processUrl}}/api/process/5f24c66f-04a7-433a-b64a-a765d3b8121aa/data/662856`

Was this page helpful?

# BUILDING BLOCKS / UI Designer / UI component types / Collection / Collection Prototype

## Description

This additional container type is needed to allow multiple prototypes to be defined for a single **Collection**. This allows elements from the same collection to be displayed differently.

For example, suppose you are creating a piece of UI in which the user is presented a list of possible products from which to choose, but you want one of the products to be highlighted as the recommended one. This example requires a collection with two **collection prototypes** (one for the normal product and one for the recommended one).

## Configurable properties:

1. **Prototype Identifier Key** - the key where to look in the iterated object to determine the prototype to be shown - in the below example the key is "type"
2. Prototype Identifier Value - the value that should be present at the **Prototype Identifier Key** when this **COLLECTION\_PROTOTYPE** should be displayed - in the below example the value is "normal" or "recommended"

## Example

The screenshot shows the FLOWX.AI UI Designer interface. The top navigation bar includes the logo, project name "test-collections" (version 1, draft), language "English", and various icons for saving, starting, and more. On the left, a sidebar lists categories: Layout, Collection, and Basic. Under Layout, there are icons for Container (selected), Card, and Custom. Under Collection, there are icons for Collection and Prototype. The main workspace displays a "Collection prototype example" with two items. The first item is a "collection" card containing placeholder text "\${name}" and "\${description}". The second item is a "collection" card with a yellow border, also containing placeholder text "\${name}" and "\${description}". To the right, a sidebar shows the component tree: CONTAINER > COLLECTION > COLLECTION\_PROTOYPE. The first COLLECTION\_PROTOYPE node has "IMAGE" and "TEXT" listed under it. The second COLLECTION\_PROTOYPE node also has "IMAGE" listed under it.

The screenshot shows the FLOWX.AI UI Designer interface. On the left, there's a sidebar with icons for Container, Card, Custom, Collection, and Basic components. The main area displays a "Collection prototype example" with two items. Each item has a placeholder for "name" and "description". The first item has a blue border and is labeled "collection\_prototype". The second item has a yellow border. On the right, there's a panel titled "Prototype Settings" with sections for "Flowx props" (Prototype Identifier Key: type, Prototype Identifier Value: normal) and "ui Actions" (Add ui action). There are also "Preview" and "Edit" buttons at the top right.

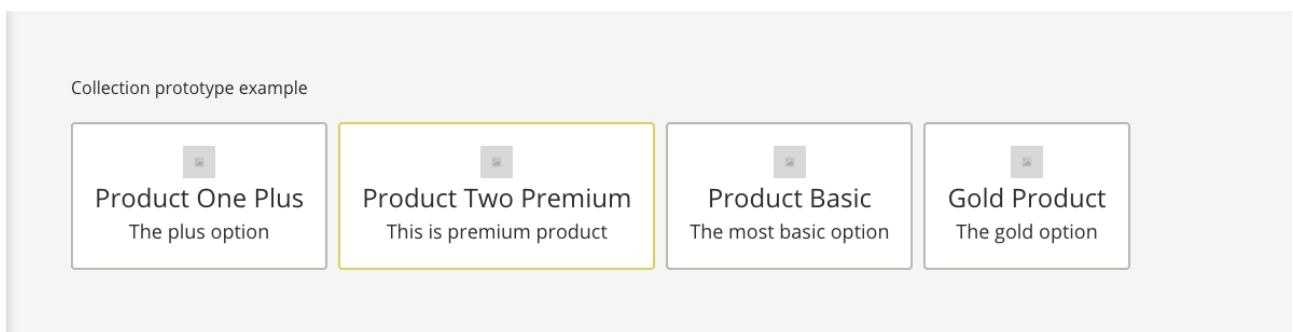
This screenshot is similar to the one above, but the "Prototype Identifier Key" in the "Flowx props" section is set to "recommended" instead of "normal". The rest of the interface, including the sidebar, preview, and settings panel, remains the same.

Source collection data example for products:

```
products: [
  {
    name: 'Product One Plus',
    description: 'The plus option'
    type: 'normal'
  },
  {
    name: 'Product Two Premium',
    description: 'The premium option'
    type: 'normal'
  }
]
```

```
        description: 'This is premium product'
        type: 'recommended',
    },
{
    name: 'Product Basic',
    description: 'The most basic option'
    type: 'normal'
},
{
    name: 'Gold Product',
    description: 'The gold option'
    type: 'normal',
}
]
```

The above configuration will render:



## Adding elements with UI Actions

There are a few differences you need to take into consideration when configuring elements that make use of **UI Actions** inside a **Collection Prototype**.

To showcase these differences, we'll use the next example:



We have a **Collection** with two employees and we want to provide the user with the option of selecting one of the employees (eg. to allow for further processing in the next steps of the process).

## Step 1 - Defining the Node Action

To select one employee from the list, we first must add an **Action** to the **User Task Node** this UI is attached to:

Node: collection-1 (ID: 431461)

Actions

Action Edit  
ID: 431905  
Name: save-item  
Order: 1  
Timer Expression  
Save Data  
Automatic:   
Manual:   
Mandatory:   
Optional:   
Repeatable:   
Autorun Children?   
Allow BACK on this action?   
Data to send  
selectedEmployee  
Add Key  
Save

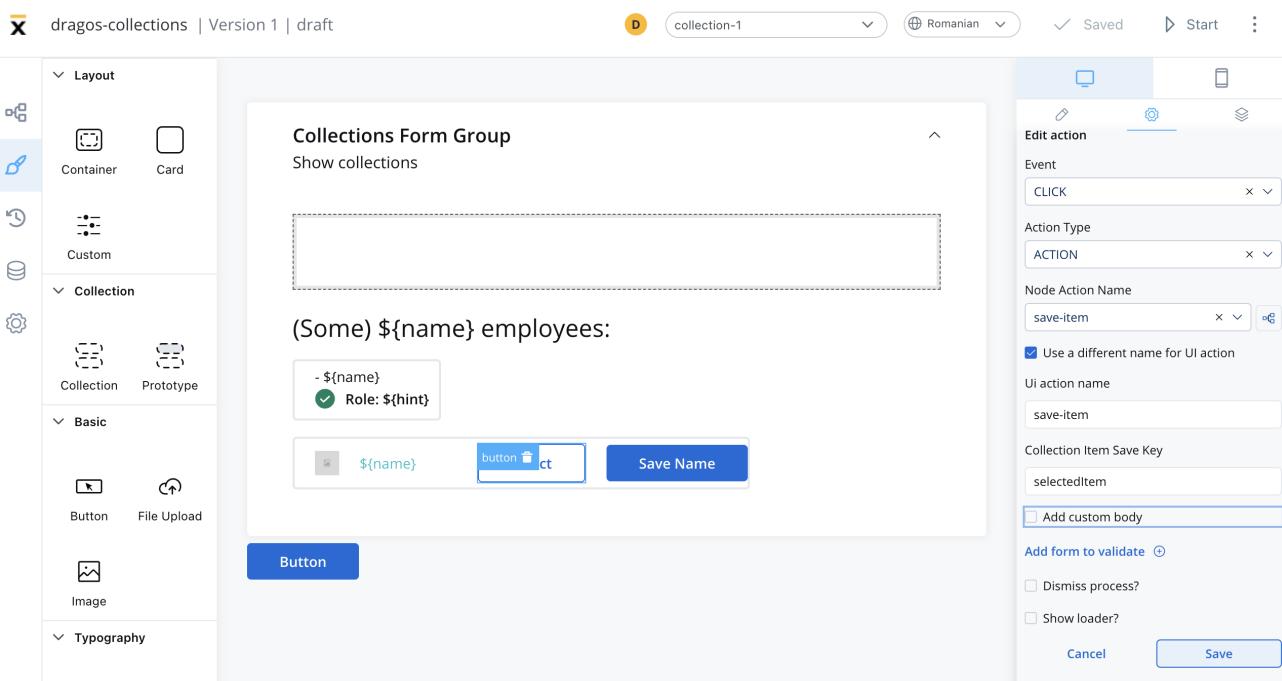
This **save-item** action is **manual** (since it will be triggered by the user) and **optionally** (since selecting an employee is not a requirement to go to the next **Node** in the process).

To allow the user to change his mind about the selected employee, this action is also marked as **Repeatable**.

Keep in mind to check the **Data to send** section. Here we are telling the platform where we want the selected employee (for which the user pressed the **Select** button) to be saved in the **process data**. In this example, we want it to be saved under the `selectedEmployee` key.

## Step 2 - Adding the Button & UI Action

Now that we have a **Node Action** defined, we can go ahead and add the **Select** button in the UI of the **User Task** which contains the Employees Collection.



**Collection Item Save Key** field has an important role in the UI Action configuration of the **Select** button. This field represents how we pass the value of the **Employee** that the user has selected to the **Node Action** that we created in **Step 1**, named `save-item`.

In our example, we set **Collection Item Save Key** to be `selectedEmployee`.

### 🔥 DANGER

**IMPORTANT:** `selectedEmployee` key is how we expose the data from the **Collection** to the Node Action. It is **imperative** that the name in the **Collection Item Save Key** is the same as the one used in the **Data to send** input in the Node Action.

The button and UI action are mostly configured as any other Button and UI Action would be configured.

## Result

This is how the process data looked before we pressed the **Select** button for an employee:

Process status

Data:

```
processInstanceId: 483001
processData: Object {"companies": [{"employees": [{"name": "Mihai Saru", "imageSrc": "https://upload.wikimedia.org/wikipedia/commons/thumb/1/1b/Svelte_Logo.svg/170px-Svelte_Logo.svg.png", "type": "color"}, {"name": "John Doe", "imageSrc": "https://upload.wikimedia.org/wikipedia/commons/thumb/1/1b/Svelte_Logo.svg/170px-Svelte_Logo.svg.png", "type": "color"}], "id": 1}, {"id": 1, "name": "Company A", "type": "color"}]}
tokenId: 483051
tokenUuid: "ae109887-e5aa-46f9-ba18-668ccce33ce8"
webSocketPath: "/ws/updates/process"
processInstanceId: "01d5b64d-7c61-4d98-a590-d23336f89f95"
webSocketAddress: "ws://public.qa.flowxai.dev/01d5b64d-7c61-4d98-a590-d23336f89f95"
```

Tokens

Token uuid	Token Status	Status Current Node	Date updated
ae109887-e5aa-46f9-ba18-668ccce33ce8	ACTIVE	EXECUTED_PARTIAL	04 May 2022, 12:09 PM

This is how the process data looks after we selected an employee from the list (notice the new field `selectedEmployee`):

### Process status

```
Data:  
  processInstanceId: 483001  
  processData: Object {"companies": [{"employees": [{"name": "Mihai Saru", "imageSrc": "https://upload.wikimedia.org/wikipedia/commons/thumb/1/1b/Svelte_Logo.svg/1702px-Svelte_Logo.svg.png", "type": "colored"}, {"name": "John Doe", "imageSrc": "https://upload.wikimedia.org/wikipedia/commons/thumb/1/1b/Svelte_Logo.svg/1702px-Svelte_Logo.svg.png", "type": "colored"}], "name": "Svelte", "imageSrc": "https://upload.wikimedia.org/wikipedia/commons/thumb/1/1b/Svelte_Logo.svg/1702px-Svelte_Logo.svg.png"}},  
  tokenId: 483051  
  selectedEmployee:  
    name: "Mihai Saru"  
    imageSrc: "https://upload.wikimedia.org/wikipedia/commons/thumb/1/1b/Svelte_Logo.svg/1702px-Svelte_Logo.svg.png"  
    type: "colored"  
  tokenUuid: "ae109887-e5aa-46f9-ba18-668ccce33ce8"  
  webSocketPath: "/ws/updates/process"  
  processInstanceUuid: "01d5b64d-7c61-4d98-a590-d23336f89f95"  
  webSocketAddress: "wss://public.qa.flownxai.dev/01d5b64d-7c61-4d98-a590-d23336f89f95"  
  
Tokens
```

Token uid	Token Status	Status Current Node	Date updated
ae109887-e5aa-46f9-ba18-668ccce33ce8	ACTIVE	EXECUTED_PARTIAL	04 May 2022, 12:09 PM

Was this page helpful?

# BUILDING BLOCKS / UI Designer / UI component types / Buttons

There are two types of buttons available, each with a different purpose. These types are:

- Basic button
- File upload button

## ▼ Basic



Button



File Upload

## Basic button

Basic buttons are used to perform an action such as unblocking a token to move forward in the process, sending an OTP, and opening a new tab.

### Configuring a basic button

When configuring a basic button, you can customize the button's settings by using the following options:

- **Properties**
- **UI action**
- **Button styling**

Sections that can be configured regarding general settings:

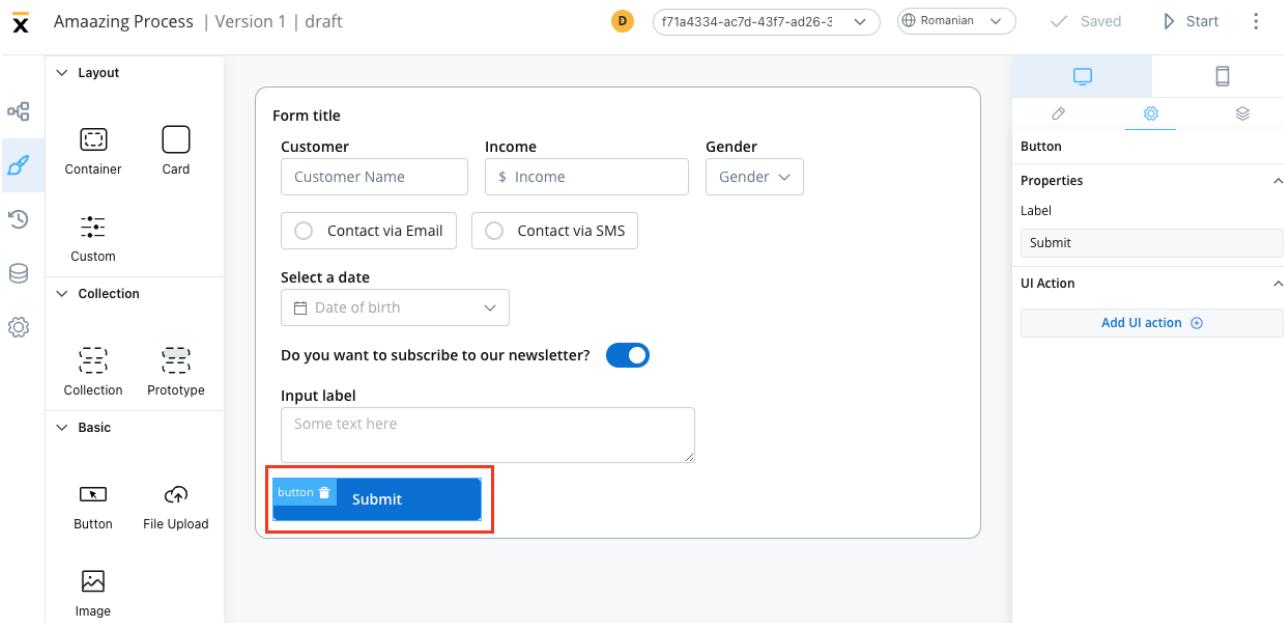
### Properties

- **Label** - it allows you to set the label that appears on the button

## UI action

Here, you can define the UI action that the button will trigger.

- **Event** - possible value: CLICK
- **Action Type** - select the action type



More details on how to configure UI actions can be found [here](#).

## Button styling

### Properties

This section enables you to select the type of button using the styling tab in the UI Designer. There are four types available:

- Primary
- Secondary
- Ghost
- Text

The screenshot shows a user interface for a form builder. On the left, there's a preview of a form with fields for Customer Name, Income, Gender, and contact preferences (Email or SMS). It also includes a date selector, a newsletter subscription toggle, and an input field with placeholder text. At the bottom is a blue 'Submit' button. On the right, a properties panel is open for the 'Submit' button. The panel includes tabs for 'Button' (selected), 'Properties', 'Sizing', and 'Spacing'. Under 'Properties', the 'Type' is set to 'primary'. Under 'Sizing', the width is 200px and height is 40px. Under 'Spacing', the padding is 0 for all sides.

### !(INFO)

For more information on valid CSS properties, click [here](#)

## Icons

To further enhance the Button UI element with icons, you can include the following properties:

- **Icon Key** - the key associated in the Media library, select the icon from the **Media Library**
- **Icon Color** - select the color of the icon using the color picker

### !(**INFO**)

When setting the color, the entire icon is filled with that color, the SVG's fill. Avoid changing colors for multicolor icons.

- **Icon Position** - define the position of the icon within the button:
  - Left
  - Right
  - Center

### !(**INFO**)

When selecting the center position for an icon, the button will display the icon only.

The form displays two sliders for financial inputs:

- Loan amount:** Set to 255000 \$. The slider bar ranges from 10000 \$ to 500000 \$.
- Down payment:** Set to 38250 \$. The slider bar ranges from 38250 \$ to 89250 \$.

Below the sliders are dropdown menus labeled "Form title" and "Loan type". A button at the bottom is highlighted with a red border.

By utilizing these properties, you can create visually appealing Button UI elements with customizable icons, colors, and positions to effectively communicate their purpose and enhance the user experience.

## File upload

This button will be used to select a file and do custom validation on it. Only the Flowx props will be different.

### Configuring a file upload button

When configuring a file upload button, you can customize the button's settings by using the following options:

- **Properties**
- **UI action**
- **Button styling**

Sections that can be configured regarding general settings:

#### Properties

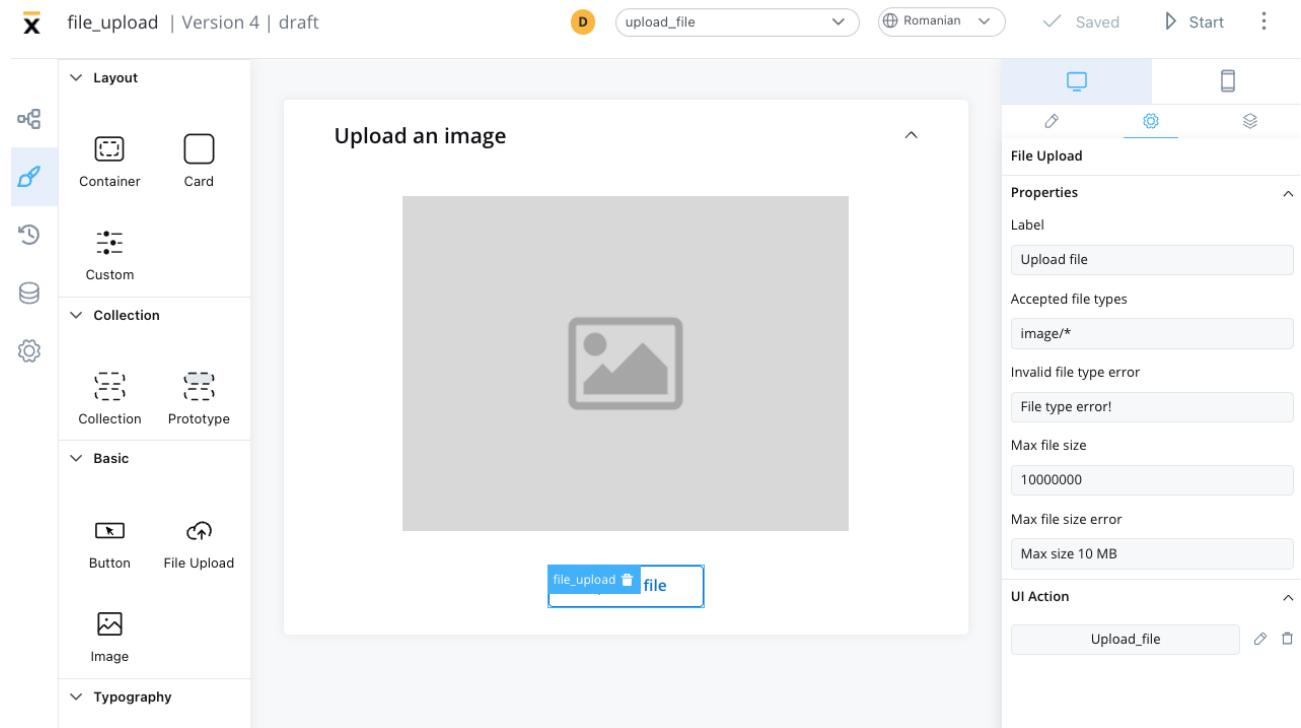
- **Label** - it allows you to set the label that appears on the button
- **Accepted file types** - the accept attribute takes as its value a string containing one or more of these unique file type specifiers, **separated by commas**, may take the following forms:

Value	Definition
audio/*	Indicates that sound files are accepted

Value	Definition
image/*	Indicates that image files are accepted
video/*	Indicates that video files are accepted
<b>MIME type</b> with no params	Indicates that files of the specified type are accepted
string starting with U+002E FULL STOP character (.) (for example, .doc, .docx, .xml)	Indicates that files with the specified file extension are accepted

- **Invalid file type error**
- **Max file size**
- **Max file size error**

Example of an upload file button that accepts image files:



## UI action

Here, you can define the UI action that the button will trigger.

- **Event** - possible value: `CLICK`
- **Action Type** - select the action type

The screenshot shows the FLOWX.AI UI Designer interface. On the left, there is a preview window titled "Upload an image" showing a placeholder for an uploaded image. Below the preview is a "file\_upload" button with a "file" icon. On the right, the configuration panel is open for the "File Upload" component. The panel includes tabs for "File Upload" (selected), "Properties", "Label", "Accepted file types", "Invalid file type error", "Max file size", and "Max file size error". A red box highlights the "UI Action" section, which contains fields for "Upload\_file", "Event" (set to "CLICK"), "Action Type" (set to "ACTION"), and a list of options including "DISMISS", "ACTION" (which is selected), "START\_PROCESS\_INHERIT", "UPLOAD", and "EXTERNAL". There are also checkboxes for "Add form to validate" and "Dismiss process?", and buttons for "Cancel" and "Save".

### INFO

More details on how to configure UI actions can be found [here](#).

## Button styling

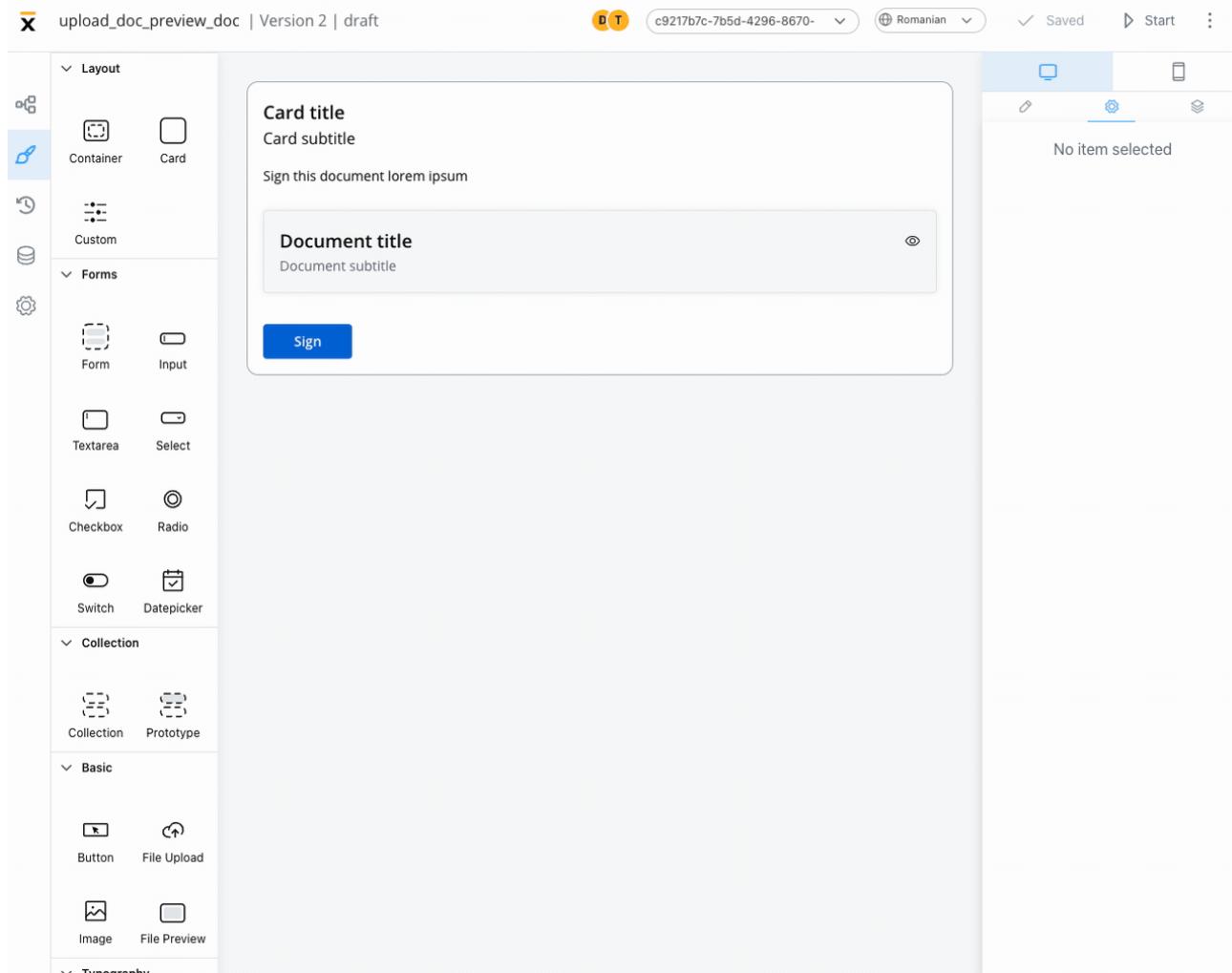
The file upload button can be styled using valid CSS properties (more details [here](#))

[Was this page helpful?](#)

# BUILDING BLOCKS / UI Designer / UI component types / File Preview

## What is a File Preview UI element?

The File Preview UI element is a user interface component that enables users to preview the contents of files quickly and easily without fully opening them. It can save time and enhance productivity, providing a glimpse of what's inside a file without having to launch it entirely.

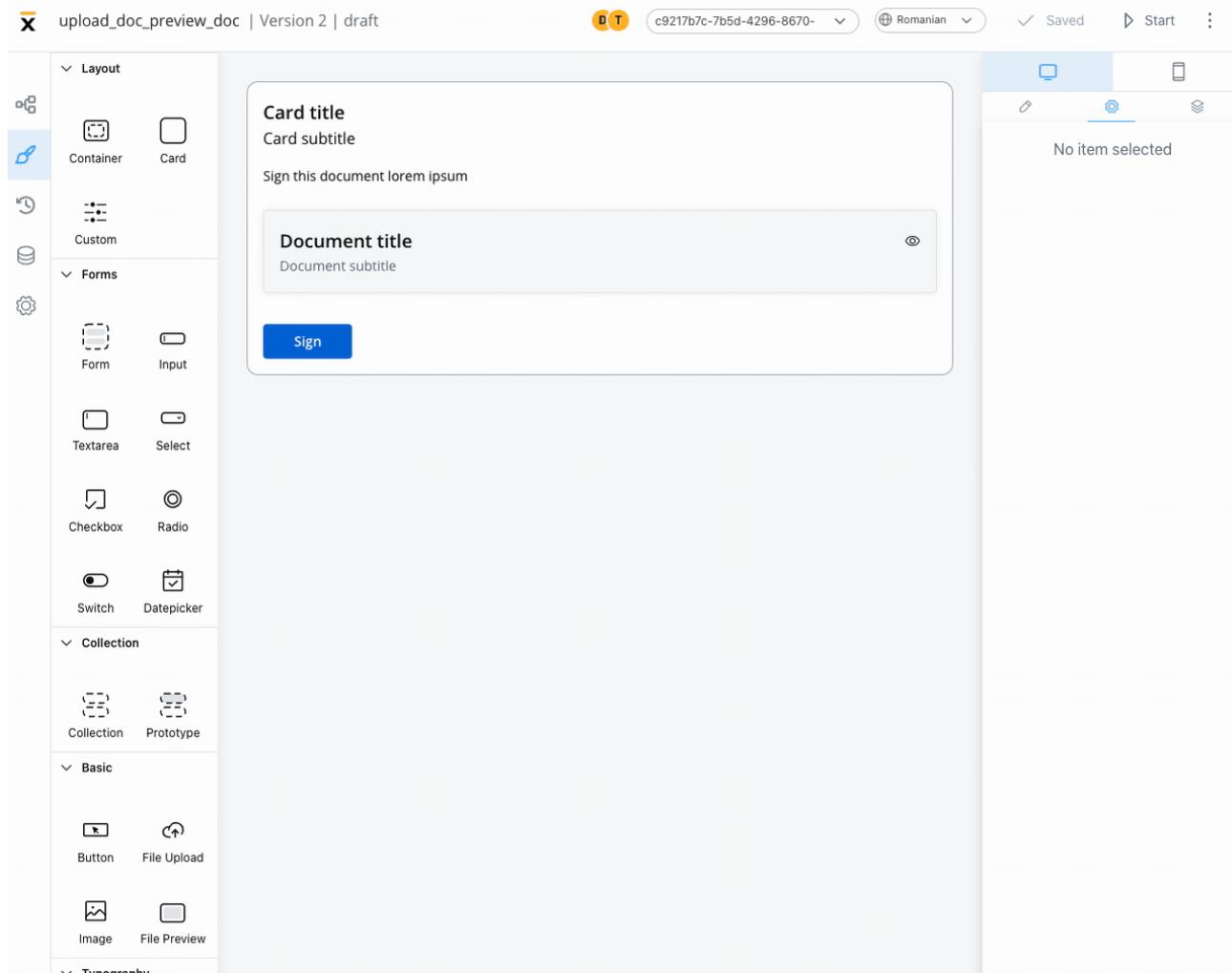


File Preview UI elements offer various benefits such as conveying information, improving the aesthetic appeal of an interface, providing visual cues and feedback or presenting complex data or concepts in a more accessible way.

## Configuring a File Preview element

A File Preview element can be configured for both mobile and web applications.

### File Preview properties (web)



The File Preview element settings consist of the following properties:

- **Title** - the title of the element (if it is downloaded or shared - the file name should be the title used in preview component)
- **Has subtitle** - the subtitle of the element
- **Display mode** - depending on the selected display method the following properties are available:
  - **Inline → Has accordion:**

- `false` - display preview inline, without expand/collapse option
  - `true` - Default View: Collapsed - display preview inline, with expand/collapse option, by default collapsed
  - `true` - Default View: Expanded - display preview inline, with expand/collapse option, by default expanded
- **Modal** → view icon is enabled
- **Source Type** -
    - **Process Data** - process key where the document is found (creates the binding between the element and process data)
    - **Static** - URL of the document

### CAUTION

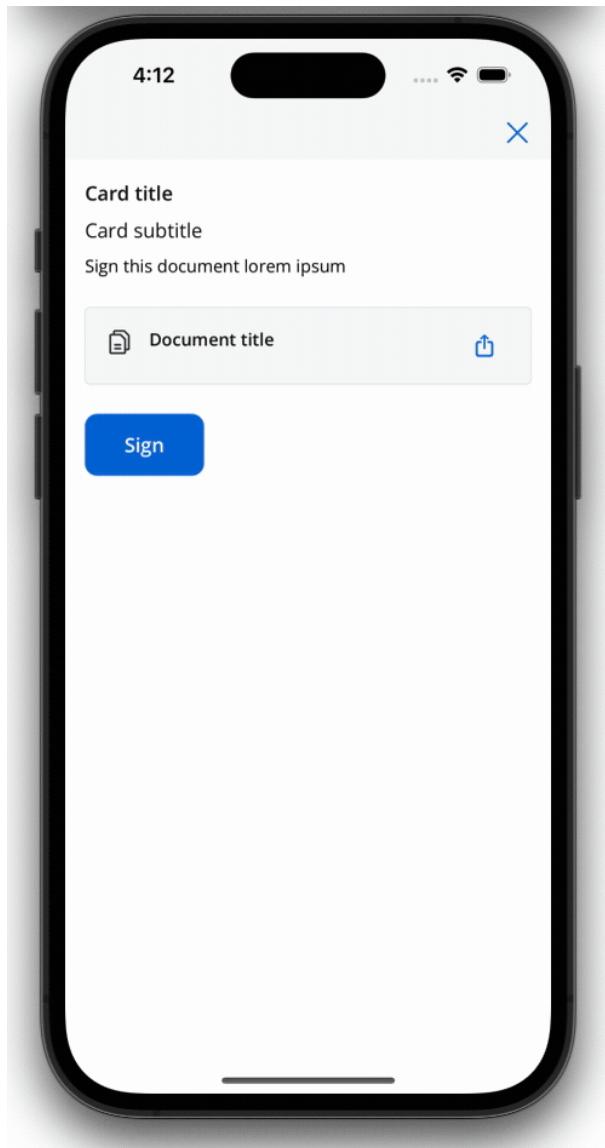
It's worth noting that the inline modal view can raise accessibility issues if the file preview's height exceeds the screen height.

## File Preview properties (mobile)

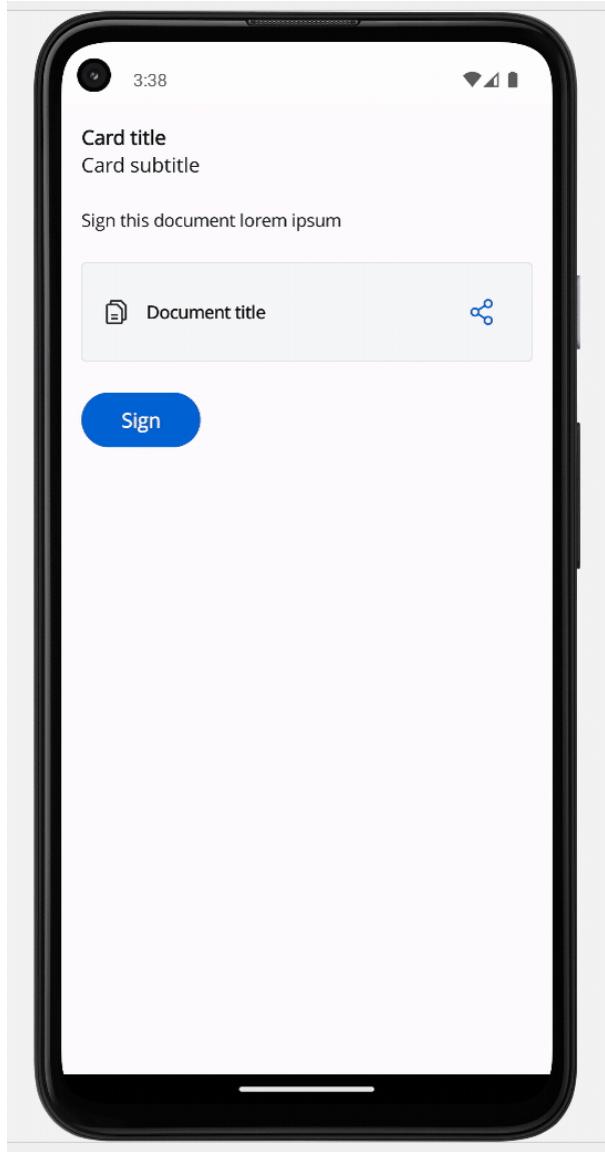
### INFO

Both iOS and Android devices support the share button.

## iOS



## Android



## File preview styling

The File Preview styling property enables you to customize the appearance of the element by adding valid CSS properties, for more details, click [here](#).

When drag and drop a File Preview element in UI Designer, it comes with the following default styling properties:

## Sizing

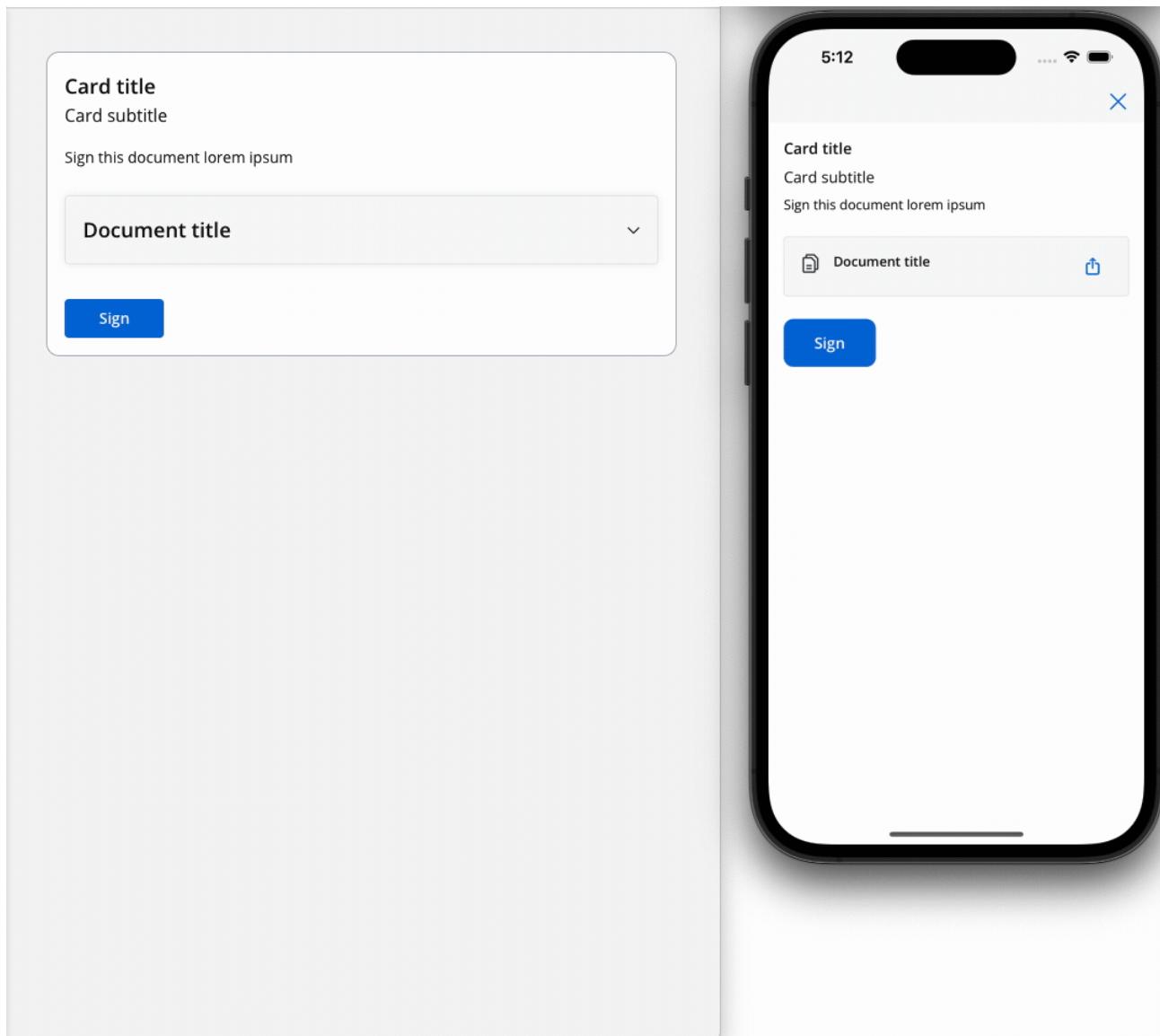
- **Fit W - auto**
- **Fit H - fixed / Height - 400 px**

The screenshot illustrates the FLOWX.AI UI Designer interface. On the left, a sidebar lists various component categories: Layout (Container, Card, Custom), Forms (Form, Input, Textarea, Select, Checkbox, Radio, Switch, Datepicker), Collection (Collection, Prototype), and Basic (Button, File Upload). In the center, a card titled "Card title" contains "Card subtitle" and the placeholder text "Sign this document lorem ipsum". Below this is a "File Preview" component showing a PDF titled "Document title" with the subtitle "Document subtitle". The PDF content is a sample text about Lorem ipsum. To the right of the preview is a detailed configuration panel for the "File Preview" component, including sections for Sizing (Fit W: auto, Fit H: fixed, Height: 400 px), Spacing, Typography (Title color #1E1E1C, Subtitle color #1E1E1C), Background (color #1E1E1C), Border (Radius: 0 px, Width: 0 px, Color: #1E1E1C), and Advanced settings (Add class: Comma separated class names).

## File Preview example

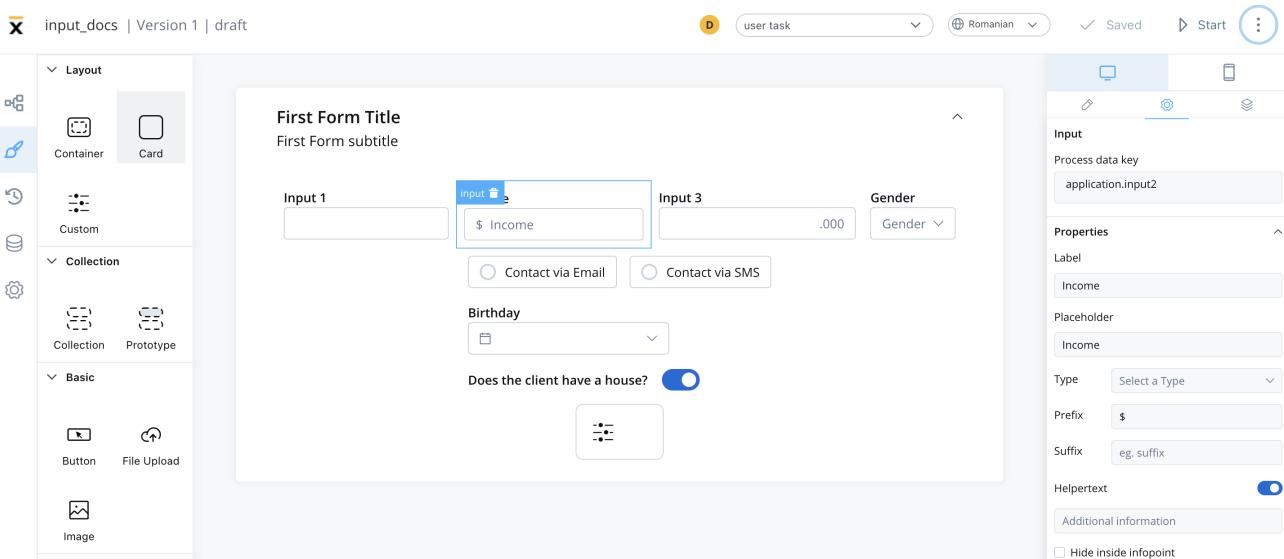
Below is an example of a File Preview UI element with the following properties:

- **Display mode - Inline**
- **Has accordion - True**
- **Default view - Expanded**
- **Source Type - Static**



**Was this page helpful?**

# BUILDING BLOCKS / UI Designer / UI component types / Form elements / Input



An input field is a form element that enables users to input data with validations and can be hidden or disabled.

## Configuring the input element

### Input settings

The Input Field offers the following configuration options:

- General
- Properties
- Datasource

- **Validators**
- **Expressions**
- **UI actions**
- **Input styling**

## General

- **Process data key** - creates the binding between form element and process data, so it can be later used in [decisions](#), [business rules](#) or [integrations](#)

## Properties

- **Label** - the label that appears on the input field
- **Placeholder** - the placeholder text that appears in the input field when it is empty
- **Type** - the type of data that the input field can accept, such as text, number, email, or password
- **Prefix** - a label that appears as a prefix to the input field
- **Suffix** - a label that appears as a suffix to the input field
- **Helpertext** - additional information about the input field (can be hidden inside an infopoint)

The screenshot shows the FLOWX.AI UI Designer interface. On the left is a sidebar with icons for Image, Typography (Text, Link), Forms (Form, Input, Textarea, Select, Checkbox, Radio, Switch, Datepicker), and a general settings gear icon. The main area displays a form titled "First Form Title" with a subtitle "First Form subtitle". The form contains several input fields: "Input 1" (Text), "Input 2" (Text with placeholder "\$ Income" and helper text "This is a helper text"), "Input 3" (Text with placeholder ".000"), and a "Gender" dropdown. Below these are two radio buttons for "Contact via Email" and "Contact via SMS", and a date picker for "Birthday". A toggle switch labeled "Does the client have a house?" is also present. To the right of the form is a sidebar for "Input" configuration, showing "Process data key: application.input2" and various properties like Label (Income), Placeholder (Income), Type (number), Prefix (\$), Suffix (eg. suffix), and Helpertext (This is a helper text). There is also a checkbox for "Hide inside infopoint".

## Datasource

The default value for the element can be configured here, this will autofill the input field when you will run the process.

This screenshot is similar to the previous one, showing the FLOWX.AI UI Designer interface. The form and sidebar are identical. However, the right sidebar has a different configuration for the "Input" component. It now shows "Datasource" instead of "Properties". Under "Datasource", there is a "Default value" field containing the value "555". Other sections like "Validators", "Expressions", and "UI Action" are also visible in the sidebar.

**First Form Title** ^

First Form subtitle

Income Mortgage Gender

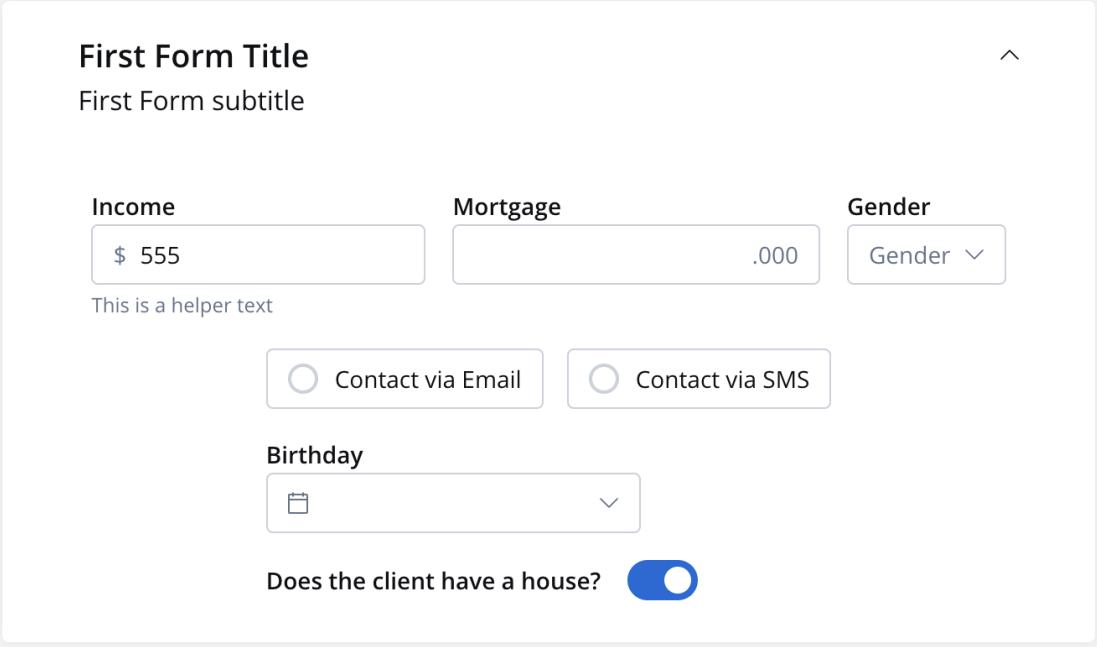
\$ 555 .000 Gender ▾

This is a helper text

Contact via Email  Contact via SMS

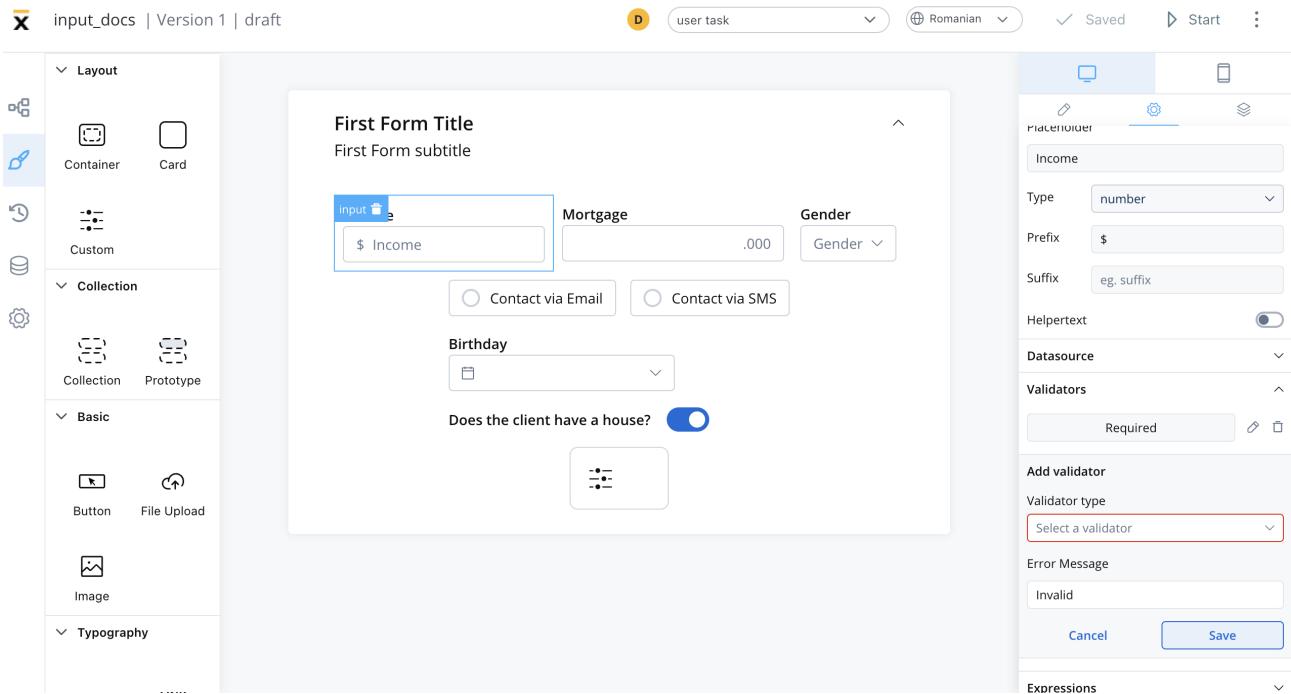
Birthday ▼

Does the client have a house?



## Validators

There are multiple validators can be added to an input (more details [here](#)).



## Expressions

The input field's behavior can be defined using JavaScript expressions for hiding or disabling the element. The following properties can be configured for expressions:

- **Hide** - JavaScript expression used to hide the Input Field when it returns a truthy value
- **Disabled** - JavaScript expression used to disable the Input Field when it returns a truthy value

### !(INFO)

It's important to make sure that disabled fields have the same expression configured under the path expressions → hide.

The screenshot shows the FLOWX.AI UI Designer interface. On the left, there's a sidebar with categories: Layout (Container, Card, Custom), Collection (Collection, Prototype), and Basic (Button, File Upload). The main area displays a form titled "First Form Title" with a subtitle "First Form subtitle". The form contains several input fields: a text input for "Income" with a placeholder "\$ Income", a numeric input for "Mortgage" with a placeholder ".000", a dropdown for "Gender", and two radio buttons for "Contact via Email" and "Contact via SMS". Below these are a date input for "Birthday" and a toggle switch for "Does the client have a house?". To the right of the form is a panel for configuring the "Income" field, which includes settings for Placeholder, Type (number), Prefix (\$), Suffix (eg. suffix), Helpertext, Datasource, Validators, and Expressions. The "Expressions" section is highlighted with a red border and contains the expression \${application.key}=='TEST'.

## UI actions

UI actions can be added to the Input Field to define its behavior and interactions.

- **Event** - possible value: CHANGE
- **Action Type** - select the action type

The screenshot shows the FLOWX.AI UI Designer interface. On the left, there's a sidebar with categories like Layout, Collection, and Basic, each containing icons for Container, Card, Custom, Collection, Prototype, Button, File Upload, and Image. The main area displays a form titled "First Form Title" with a subtitle "First Form subtitle". The form contains several input fields: a text input with a dollar sign prefix and a file icon, a text input for "Mortgage", a dropdown for "Gender", two radio buttons for "Contact via Email" and "Contact via SMS", a date input for "Birthday", and a toggle switch for "Does the client have a house?". To the right of the form is a detailed configuration panel for the first input field. It includes tabs for desktop and mobile views, settings for placeholder text ("Income"), type ("number"), prefix ("\$"), suffix ("eg. suffix"), help text, data source, validators, expressions, hide logic (\${application.key} != 'TEST'), disabled status, and UI actions.

## !(INFO)

For more details on how to configure a UI action, click [here](#).

## Input styling

### Icons

- **Icon Key** - the key associated in the Media library, select the icon from the **Media Library**
- **Icon Color** - select the color of the icon using the color picker

## !(INFO)

When setting the color, the entire icon is filled with that color, the SVG's fill. Avoid changing colors for multicolor icons.

You have the option to enhance the Input element by incorporating two types of icons:

- **Left Icon:** You can include an icon on the left side of the Input element. This icon can serve as a visual cue or symbol associated with the input field's purpose or content.
- **Right Icon:** Same as left icon.

By utilizing these two types of icons, you can provide users with a more intuitive and visually appealing experience when interacting with the Input element.

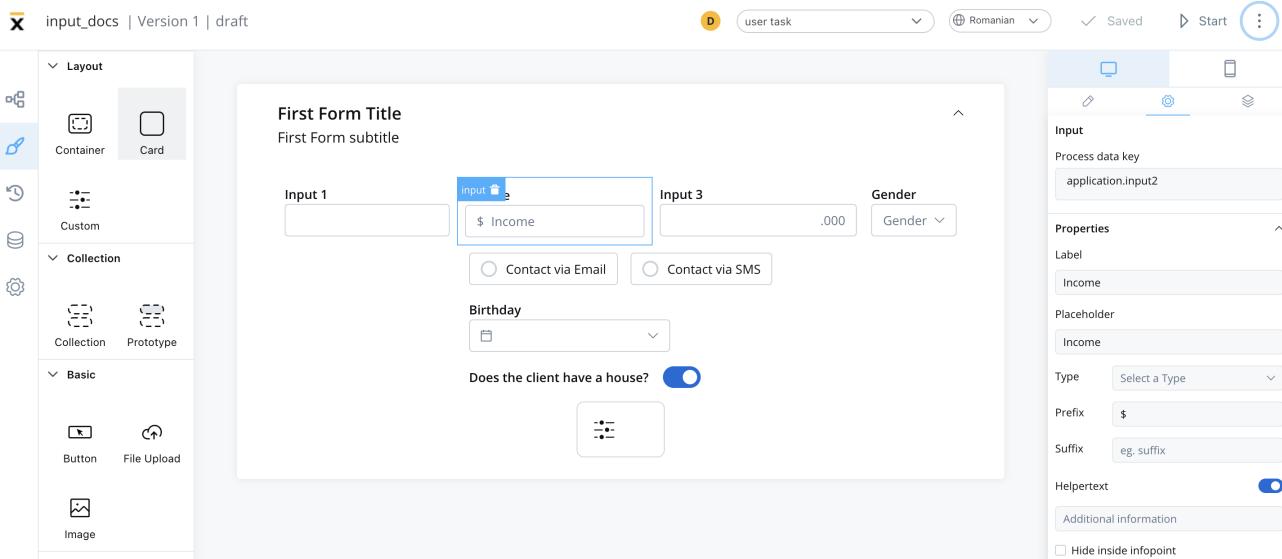
The screenshot shows a user interface for 'Enter Personal Information'. On the left, there's a form with fields for First Name, Last Name, and Date of birth. Below these are sections for Employment type (radio buttons for Employed and Pensioner), a Save personal information toggle switch, and a Loan amount slider set at 255000 \$. At the bottom, there's a Form title field. On the right, a sidebar titled 'Input' provides detailed settings for the selected component, including 'Properties' for Left and Right Icons (both enabled with green icons), 'Sizing' (Fit W set to 'fill'), 'Spacing' (horizontal and vertical padding and margin values), and 'Typography' (font color set to blue).

- The Input Field can be styled using valid CSS properties (more details [here](#))

The screenshot shows the FLOWX.AI UI Designer interface. On the left, there's a sidebar with categories like Layout, Collection, Basic, and more, each containing icons for different components. The main area displays a form titled "First Form Title" with a subtitle "First Form subtitle". The form contains several input fields: a text input with a dollar sign placeholder, a text input for "Mortgage" with a ".000" suffix, a dropdown for "Gender", two radio buttons for "Contact via Email" and "Contact via SMS", a date input for "Birthday", and a toggle switch for "Does the client have a house?". To the right of the form is a "Properties" panel with sections for Input, Properties, Label, Placeholder, Type, Prefix, Suffix, Helpertext, Datasource, and Validators. The "Input" section shows "application.input2" under "Process data key". The "Properties" section shows "Income" under "Label" and "Placeholder". The "Type" section shows "number" under "Type". The "Prefix" section shows "\$" under "Prefix". The "Suffix" section shows "eg. suffix" under "Suffix". The "Helpertext" section has a checked toggle button. The "Datasource" section has a dropdown menu. The "Validators" section has a dropdown menu.

Was this page helpful?

# BUILDING BLOCKS / UI Designer / UI component types / Form elements / Text area



A text area is a form element used to capture multi-line input from users in a conversational interface. The text area component is typically used for longer inputs such as descriptions, comments, or feedback, providing users with more space to type their responses.

It is an important tool for creating intuitive and effective conversational interfaces that can collect and process large amounts of user input.

## Configuring the text area element

### Text area settings

The text area offers the following configuration options:

- **General**
- **Properties**
- **Datasource**
- **Validators**

- **Expressions**
- **UI actions**
- **Text area styling**

## General

- **Process data key** - creates the binding between form element and process data, so it can be later used in [decisions](#), [business rules](#) or [integrations](#)

## Properties

- **Label** - the label of the text area
- **Placeholder** - the placeholder text that appears in the text area
- **Helpertext** - additional information about the text area field (can be hidden inside an infopoint)

## Datasource

The default value for the element can be configured here, this will autofill the text field when you will run the process.

## Validators

There are multiple validators can be added to a text area element (more details [here](#)).

The screenshot shows the FLOWX.AI UI Designer interface. On the left, there's a sidebar with various components like Text, Link, Form, Input, Textarea, Select, Checkbox, Radio, Switch, Datepicker, and Indicators. The main area displays a form titled 'Form title' with sections for 'Customer' (Customer Name, Income, Gender), 'Contact via Email/SMS', 'Select a date' (Date of birth dropdown), and a newsletter subscription toggle. Below these is a large Textarea field with placeholder text 'Some text here'. At the bottom is a 'Submit' button. To the right of the form is a detailed sidebar for the 'Textarea' component, showing configuration options for 'Process data key' (set to 'textKey'), 'Properties' (Label, Placeholder, Helpertext, Datasource, Default value set to 'eg. Name'), and 'Validators' (with a 'Add a validator' button).

## Expressions

The text area's behavior can be defined using JavaScript expressions for hiding or disabling the element. The following properties can be configured for expressions:

- **Hide** - JavaScript expression used to hide the text area when it returns a truthy value
- **Disabled** - JavaScript expression used to disable the text area when it returns a truthy value

### INFO

It's important to make sure that disabled fields have the same expression configured under the path expressions → hide.

The screenshot shows the FLOWX.AI UI Designer interface. On the left, there's a sidebar with categories like Layout, Collection, and Basic, each containing icons for Container, Card, Custom, Collection, Prototype, Button, File Upload, and Image. The main area displays a form titled "First Form Title" with a subtitle "First Form subtitle". The form contains several input fields: a text input with placeholder "\$ Income", a numeric input with placeholder ".000" and a suffix ".000", a dropdown menu for "Gender", two radio buttons for "Contact via Email" and "Contact via SMS", a date input for "Birthday", and a toggle switch for "Does the client have a house?". To the right of the form is a panel for configuring the "Income" field, which includes settings for Placeholder, Type (number), Prefix (\$), Suffix (eg. suffix), Helpertext, Datasource, Validators, and Expressions. The "Expressions" section is highlighted with a red border and contains the expression \${application.key}=='TEST'.

## UI actions

UI actions can be added to the text area field to define its behavior and interactions.

- **Event** - possible value: CHANGE
- **Action Type** - select the action type

The screenshot shows the FLOWX.AI UI Designer interface. On the left, there's a sidebar with categories like TEXT, LINK, FORMS, INDICATORS, and a MESSAGE icon. The FORMS section is expanded, showing options for Form, Input, Textarea, Select, Checkbox, Radio, Switch, and Datepicker. The INDICATORS section is also expanded, showing a Message icon. The main area displays a form titled "Form title". It includes fields for "Customer Name", "Income", and "Gender". There are two radio buttons for "Contact via Email" and "Contact via SMS". A dropdown menu for "Select a date" is set to "Date of birth". A toggle switch is turned on next to the question "Do you want to subscribe to our newsletter?". Below these is a large text area containing "Some text here". At the bottom is a blue "Submit" button. To the right of the form, there are several configuration panels: "Helpertext" (with a toggle switch), "Datasource" (with a dropdown menu), "Default value" (with a text input field containing "eg. Name"), "Validators" (with a "Add a validator" button), "Expressions" (with a "Hide" section containing a condition "\$(textKey) !== 'TEST'"), "Disabled" (with a dropdown menu containing "-"), and "UI Action" (with a "Add UI action" button). The entire interface has a light gray background with blue highlights for selected components.

## INFO

For more details on how to configure a UI action, click [here](#).

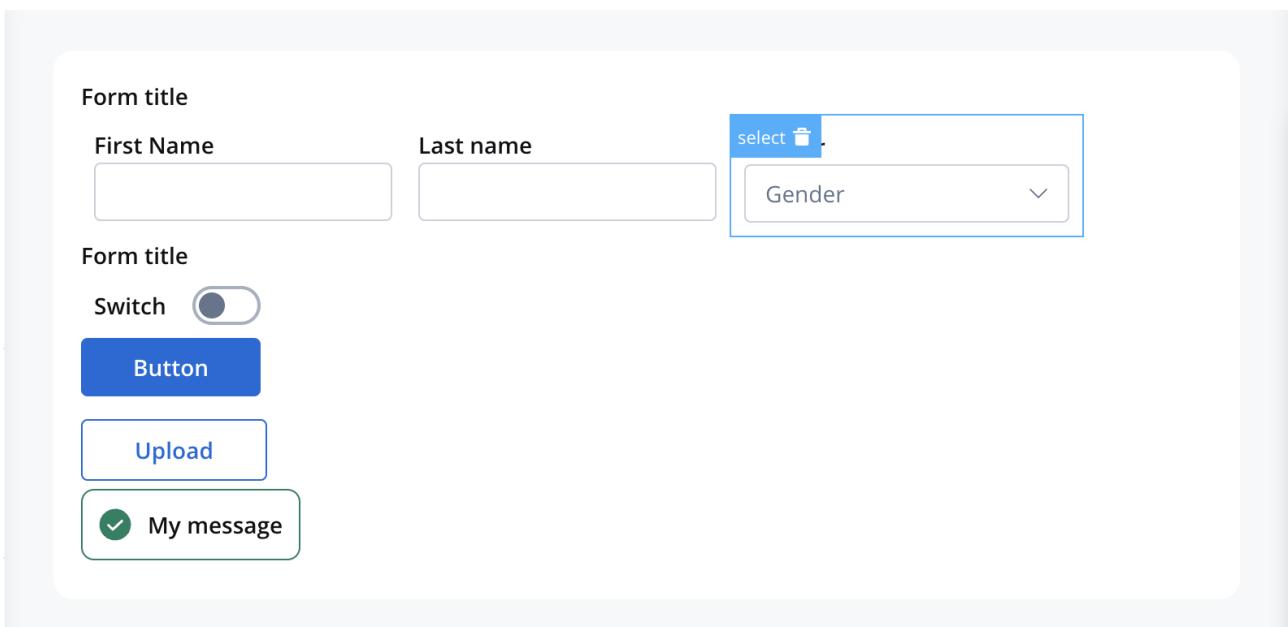
## Text area styling

The ability to style the text area element using CSS properties is relevant because it allows you to customize the appearance of the text area to match the overall design of the website or application.

» [UI Designer styling](#)

Was this page helpful?

# BUILDING BLOCKS / UI Designer / UI component types / Form elements / Select



The Select form field is an element that enables users to make a choice from a list of predefined options. It consists of multiple values, each of which is defined by a label that is displayed in the dropdown menu, and a code that is saved.

## (!) INFO

For instance, you could have a label of "Female" with the value "F" and "Male" with the value "M". This means that when a user selects "Female" in the process instance, the value "F" will be stored for the "Select" key.

## Configuring the Select element

## Select Settings

These allow you to customize the settings for the Select Field:

- **General**
- **Properties**
- **Datasource**
- **Validators**
- **Expressions**
- **UI actions**
- **Select styling**

### General

- **Process data key** - creates the binding between form element and process data so it can be later used in **decisions**, **business rules** or **integrations**

### Properties

- **Label** - the label of the select
- **Placeholder** - placeholder when the field has no value
- **Empty message** - text displayed for custom type when no results are found
- **Search for options** - displays a search to filter options
- **Helpertext** - additional information about the select field (can be hidden inside an infopoint)

### Datasource

- **Default value** - autofill the select with this value. Going back to the example with Woman label with F value and Man with M to have a default value of Woman we need to configure here F
- **Source Type** - it can be Static, Enumeration, or Process Data
- **Add option** - label - value pairs can be defined here

## Validators

There are multiple validators can be added to a select (more details [here](#)).

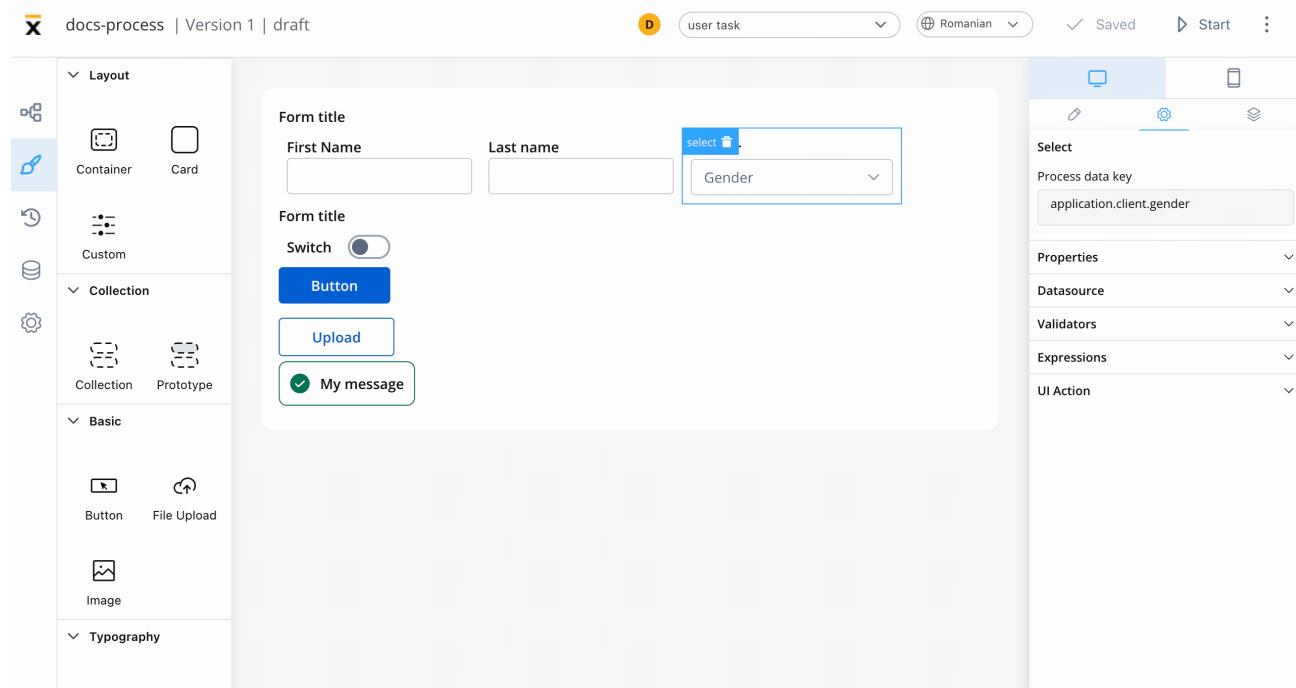
## Expressions

The select field's behavior can be defined using JavaScript expressions for hiding or disabling the element. The following properties can be configured for expressions:

- **Hide** - JavaScript expression used to hide the Select Field when it returns a truthy value
- **Disabled** - JavaScript expression used to disable the Select Field when it returns a truthy value

### (!) INFO

It's important to make sure that disabled fields have the same expression configured under the path expressions → hide.



## UI actions

UI actions can be added to the select element to define its behavior and interactions.

- **Event** - possible value: `CHANGE`
- **Action Type** - select the action type

### !(INFO)

For more details on how to configure a UI action, click [here](#).

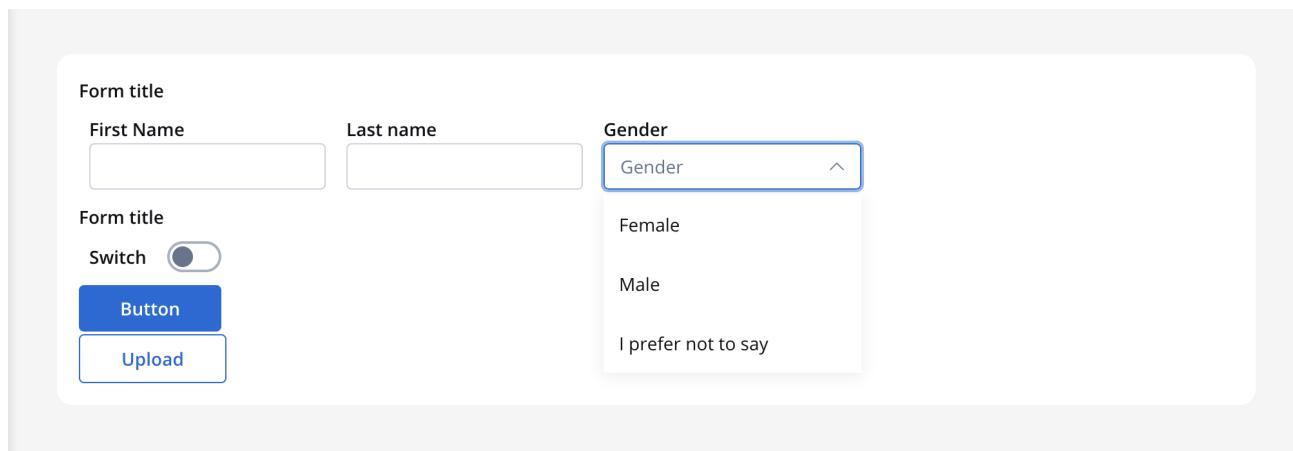
## Select styling

Styling the Select field using CSS properties allows you to customize the appearance of the dropdown list and make it more visually appealing and

consistent with the overall design of the website or application.

### » UI Designer styling

For example, a FORM element with a **layout** configuration including direction of Horizontal and some inputs, and a select element will look like this:



## Icons

When customizing the appearance of a Select UI element that includes an icon, you can utilize the following properties:

- **Icon Key** - the key associated in the Media library, select the icon from the **Media Library**
- **Icon Color** - select the color of the icon using the color picker

### (!) INFO

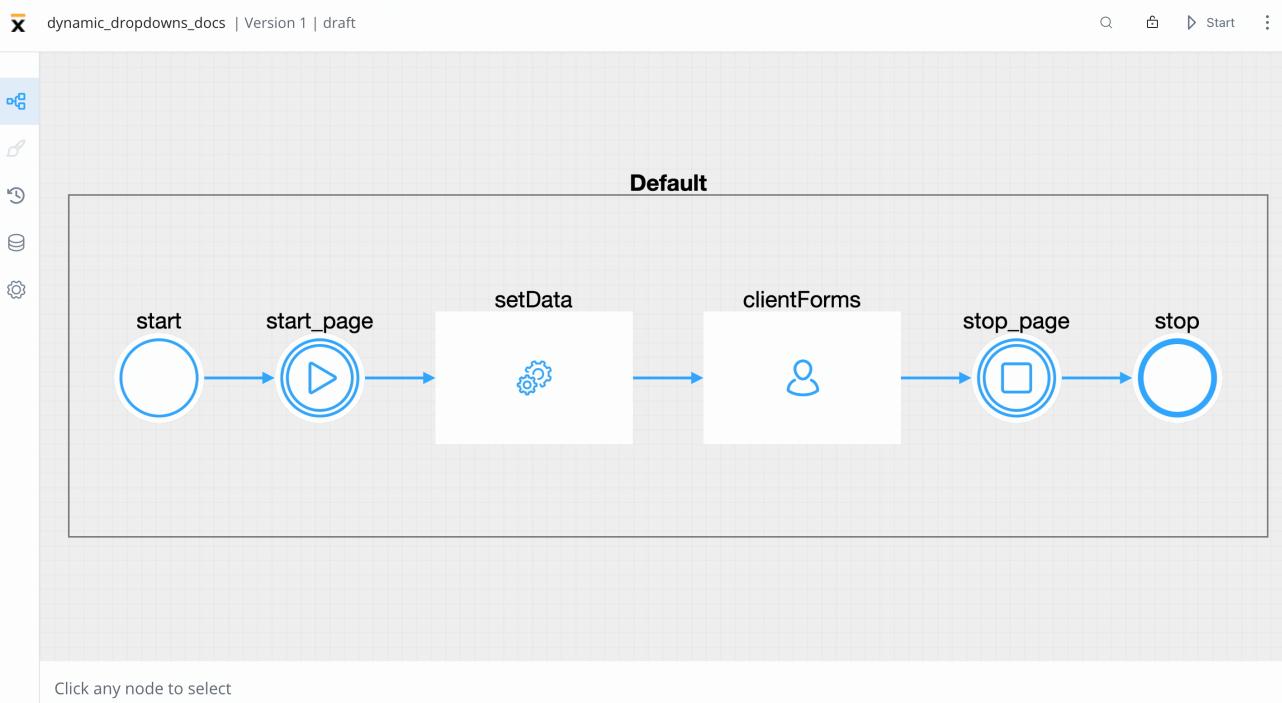
When setting the color, the entire icon is filled with that color, the SVG's fill. Avoid changing colors for multicolor icons.

## Example - Dynamic dropdowns

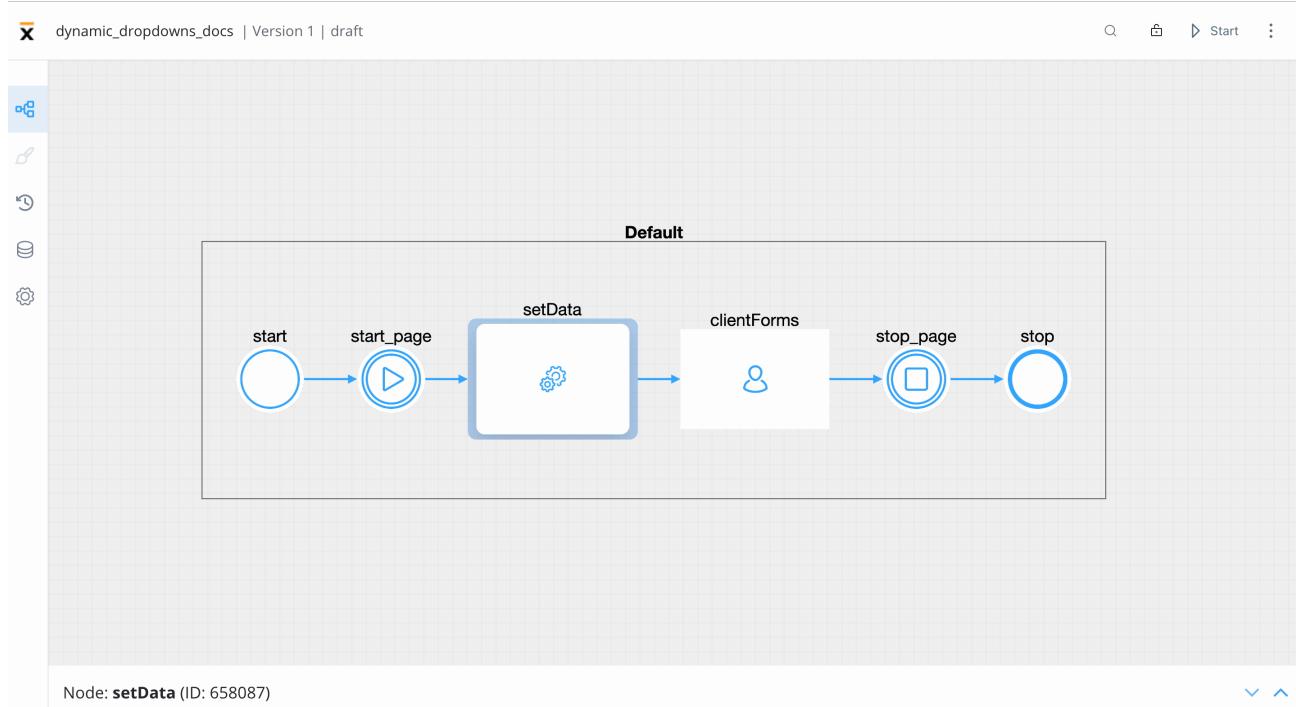
As mentioned previously, you can create dropdowns including static data, enumerations, or **process data**. Let's create an example using **process data** to create a process that contains **dynamic dropdowns**.

To create this kind of process, we need the following elements:

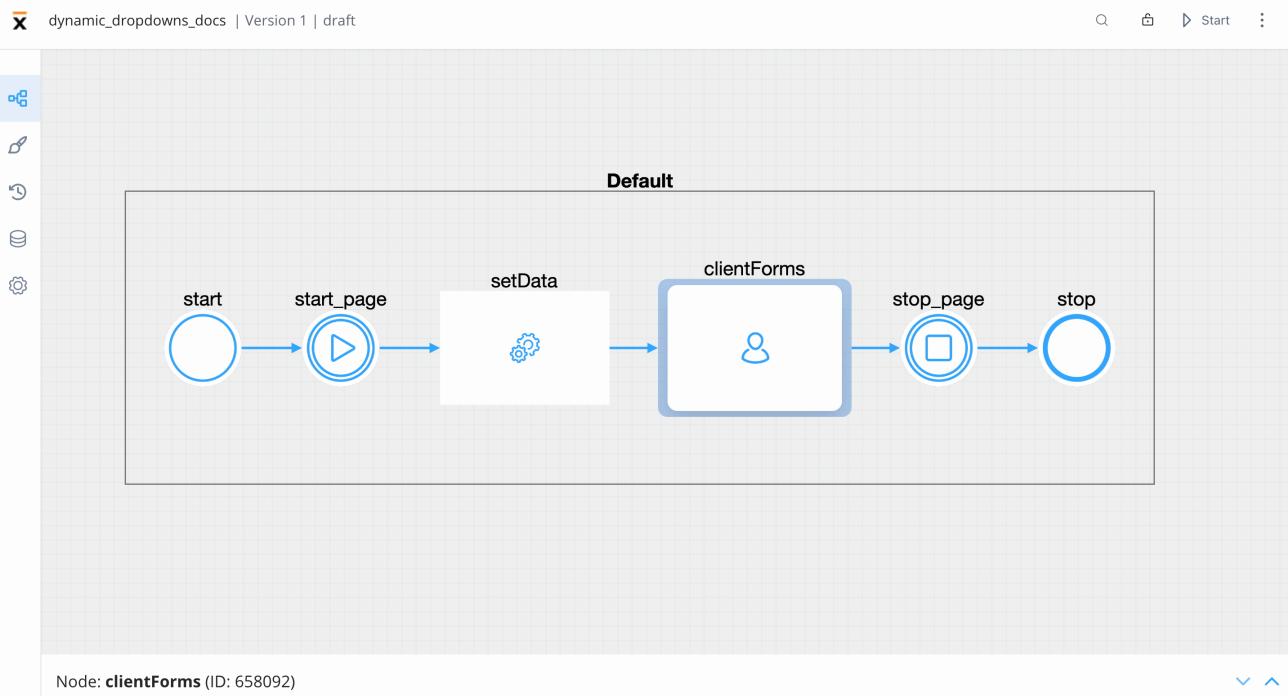
- a **start** node and an **end** node
- a **start milestone** UI element to it and an **end milestone** node



- a **task node** (this will be used to set which data will be displayed on the dropdowns)



- a **user task node** (here we have the client forms and here we add the SELECT elements)



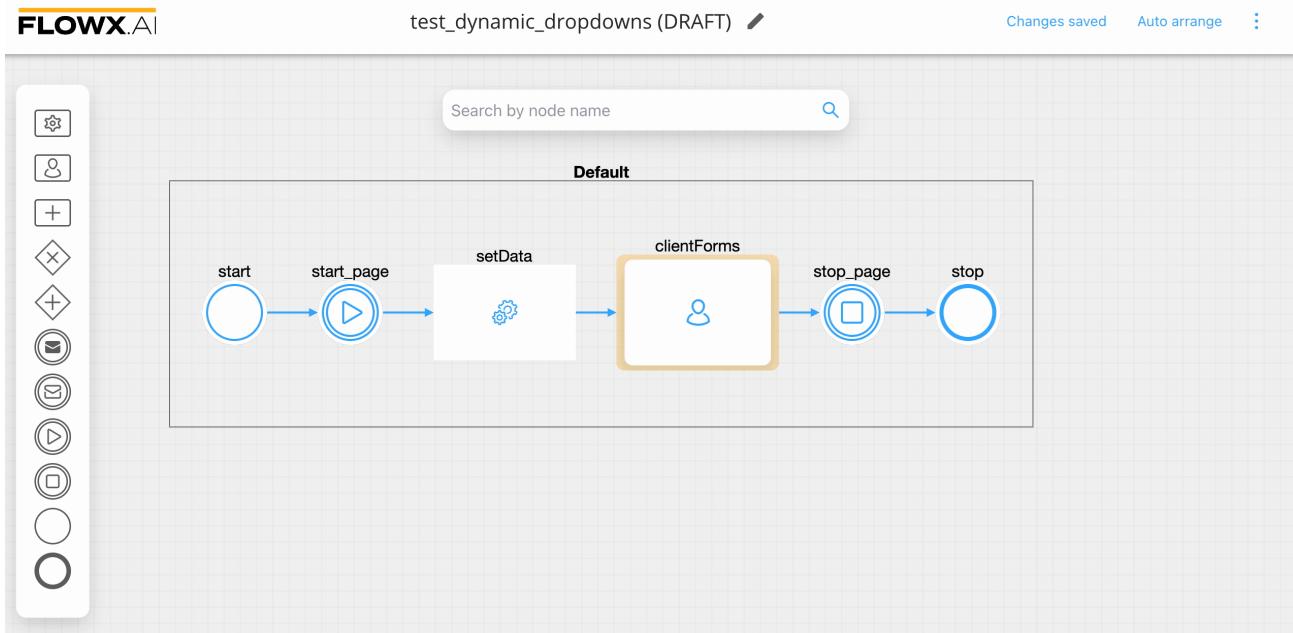
## Creating the process

Follow the next steps to create the process from scratch:

1. Open **FLOWX Designer** and from the **Processes** tab select **Definitions**.
2. Click on the breadcrumbs (top-right corner) then click **New process** (the Process Designer will now open).
3. Now add all the **necessary nodes** (as mentioned above).

## Configuring the nodes

1. On the **start milestone** node, add a **page** UI element.
2. On the **task node**, add a new **Action** (this will set the data for the dropdowns) with the following properties:
  - Action type - **Business Rule**
  - **Automatic**
  - **Mandatory**
  - **Language** (we used an **MVEL** script to create a list of objects)
3. On the **user task node**, add a new **Action** (submit action, this will validate the forms and save the date) with the following properties:
  - **Action type** - Save Data
  - **Manual**
  - **Mandatory**
  - **Data to send** (the key where the data will be sent) - **application**



Below you can find the MVEL script used in the above example:

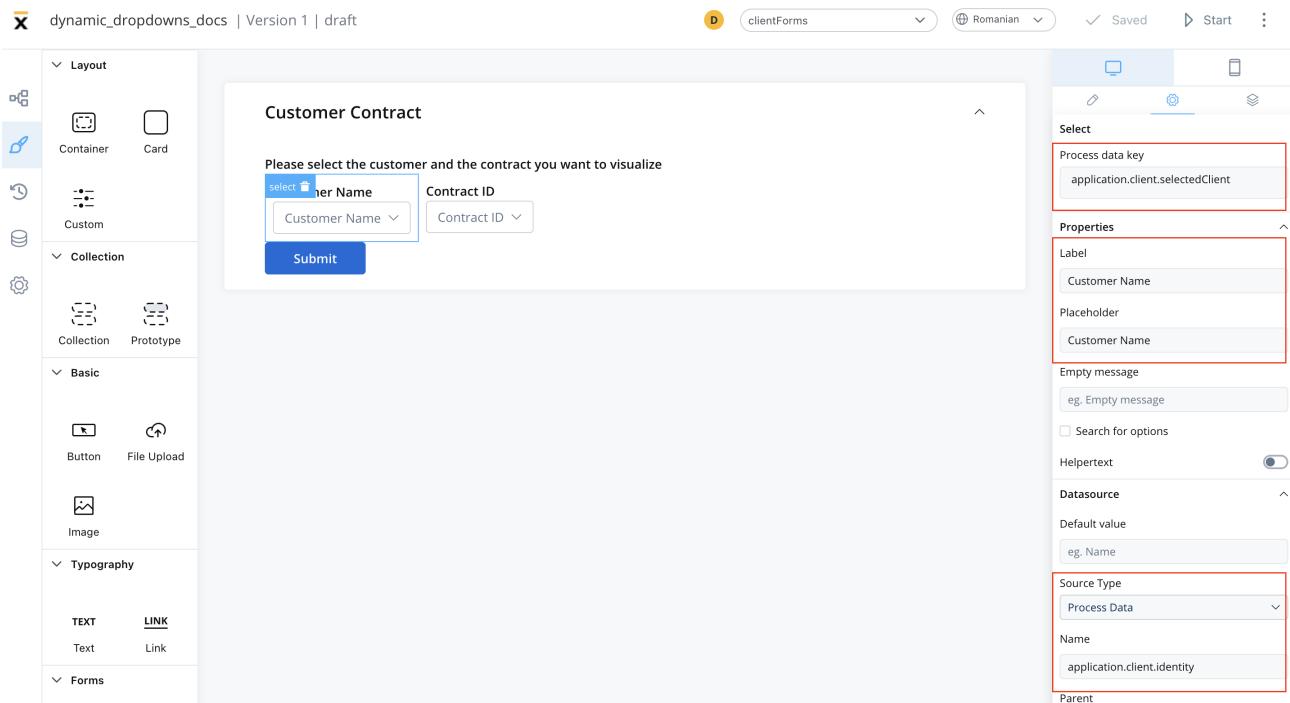
```
output.put("application",
{
    "client": {
        "identity": [
            {
                "value": "001",
                "label": "Eddard Stark"
            },
            {
                "value": "002",
                "label": "Sansa Stark"
            },
            {
                "value": "003",
                "label": "Catelyn Stark"
            }
        ]
    },
}])
```

```
"contracts": {
    "001": [
        {
            "value": "c001",
            "label": "Eddard Contract 1"
        },
        {
            "value": "c007",
            "label": "Eddard Contract 2"
        }
    ],
    "003": [
        {
            "value": "c002",
            "label": "Catelyn Contract 1",
        },
        {
            "value": "c003",
            "label": "Catelyn Contract 2",
        },
        {
            "value": "c004",
            "label": "Catelyn Contract 3"
        }
    ],
    "002": [
        {
            "value": "c005",
            "label": "Sansa Contract 1",
        }
    ]
});
```

## Configuring the UI

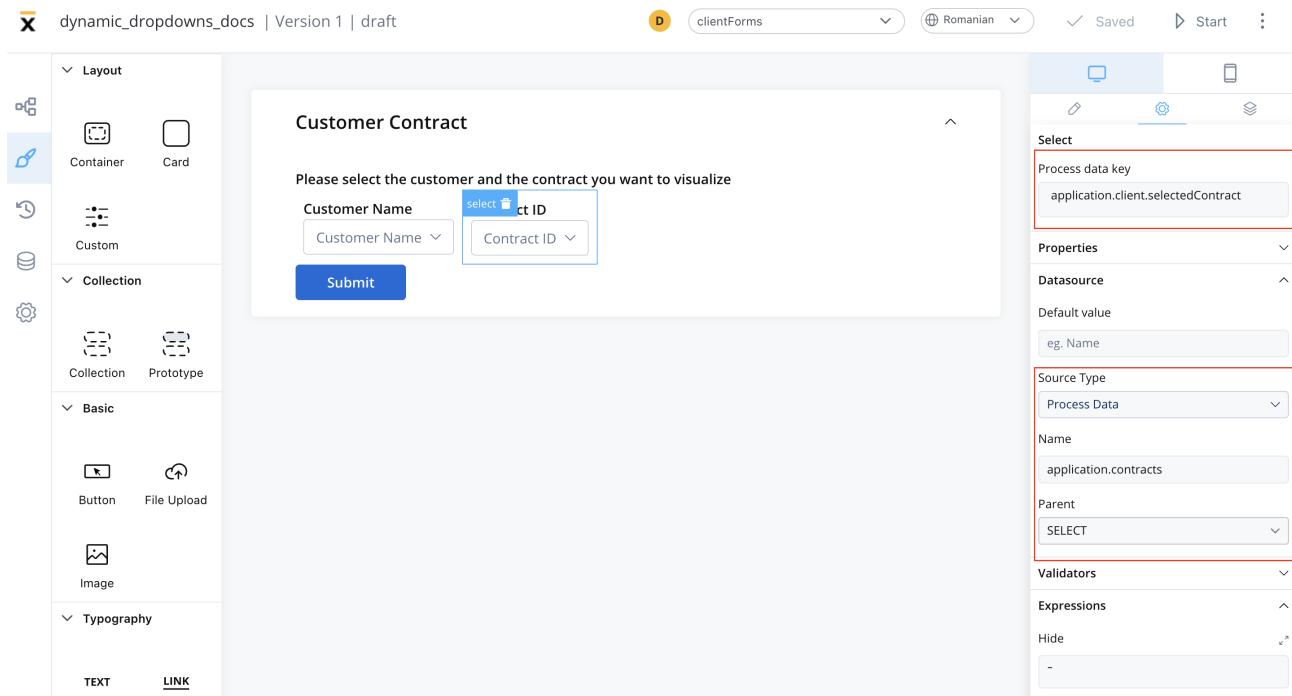
Follow the next steps to configure the UI needed:

1. Select the **user task node** and click the **brush icon** to open **UI Designer**
2. Add a **card** element as a **root component** (this will group the other elements inside it) with the following properties:
  - **Message** - `{"application": ${application}}`
  - **Title** - *Customer Contract*
3. Inside the **card**, add a **form element**.
4. Inside the **form** add two **select elements**, first will represent, for example, the *Customer Name* and the second the *Contract ID*.
5. For first select element (Customer Name) set the following properties:
  - **Process data key** - `application.client.selectedClient`
  - **Label** - Customer Name
  - **Placeholder** - Customer Name
  - **Source type** - Process Data (to extract the data added in the **task node**)
  - **Name** - `application.client.identity`



6. For the second select element (Contract ID) set the following properties:

- o **Process data key** - `application.client.selectedContract`
- o **Label** - Contract ID
- o **Placeholder** - Contract ID
- o **Source Type** - Process Data
- o **Name** - `application.contracts`
- o **Parent** - `SELECT` (choose from the dropdown list)



7. Add a button under the form that contains the select elements with the following properties:

- **Label** - Submit
- **Add UI action** - add the submit action attached earlier to the user task node

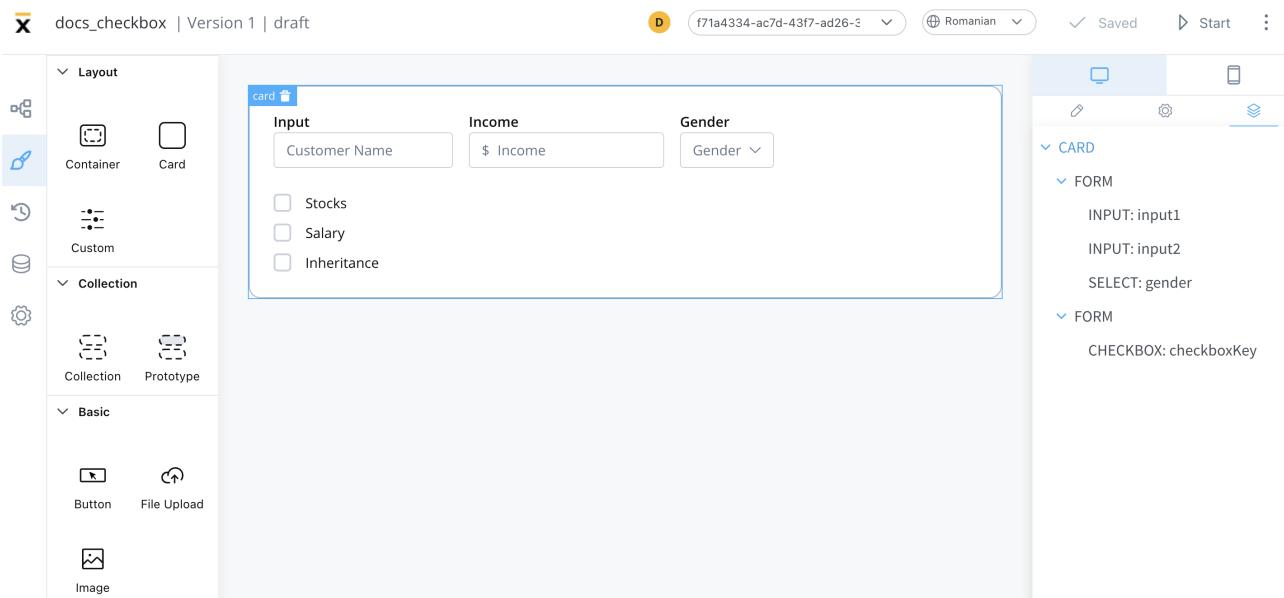
The screenshot shows the FLOWX.AI UI Designer interface. On the left, there's a sidebar with categories like Layout, Collection, Basic, and Typography, each containing various UI components such as Container, Card, Custom, Collection, Prototype, Button, File Upload, and Image. The main workspace displays a form titled "Customer Contract" with instructions: "Please select the customer and the contract you want to visualize". It contains two dropdown menus labeled "Customer Name" and "Contract ID", and a blue button labeled "mit". To the right of the workspace is a properties panel for a "Button" component. The "Properties" section shows "Label" set to "Submit". The "UI Action" section has a red border around it, containing fields for "Event" (set to "CLICK"), "Action Type" (set to "ACTION"), and "Node Action Name" (set to "submit"). Below these are several checkboxes: "Use a different name for UI action", "Add custom body", "Add form to validate", "Dismiss process?", and "Show loader?".

## 8. Test and run the process by clicking **Start process**.

The screenshot shows the FLOWX.AI process editor. On the left, there's a sidebar with icons for search, file, and settings. The main workspace shows a workflow titled "Default". The process starts at a "start" node, transitions to a "start\_page" node (represented by a play button icon), then to a "setData" step (represented by a gear icon). This is followed by a "clientForms" step (represented by a person icon). The process then continues to a "stop\_page" node (represented by a square icon), "stop", and finally ends at a "stop" node. At the bottom of the workspace, there's a note: "Click any node to select".

Was this page helpful?

# BUILDING BLOCKS / UI Designer / UI component types / Form elements / Checkbox



A checkbox form field is an interactive element in a web form that provides users with multiple selectable options. It allows users to choose one or more options from a pre-determined set by simply checking the corresponding checkboxes.

This type of form field can be used to gather information such as interests, preferences, or approvals, and it provides a simple and intuitive way for users to interact with a form.

## Configuring the checkbox element

## Checkbox settings

The available configuration options for this form element are:

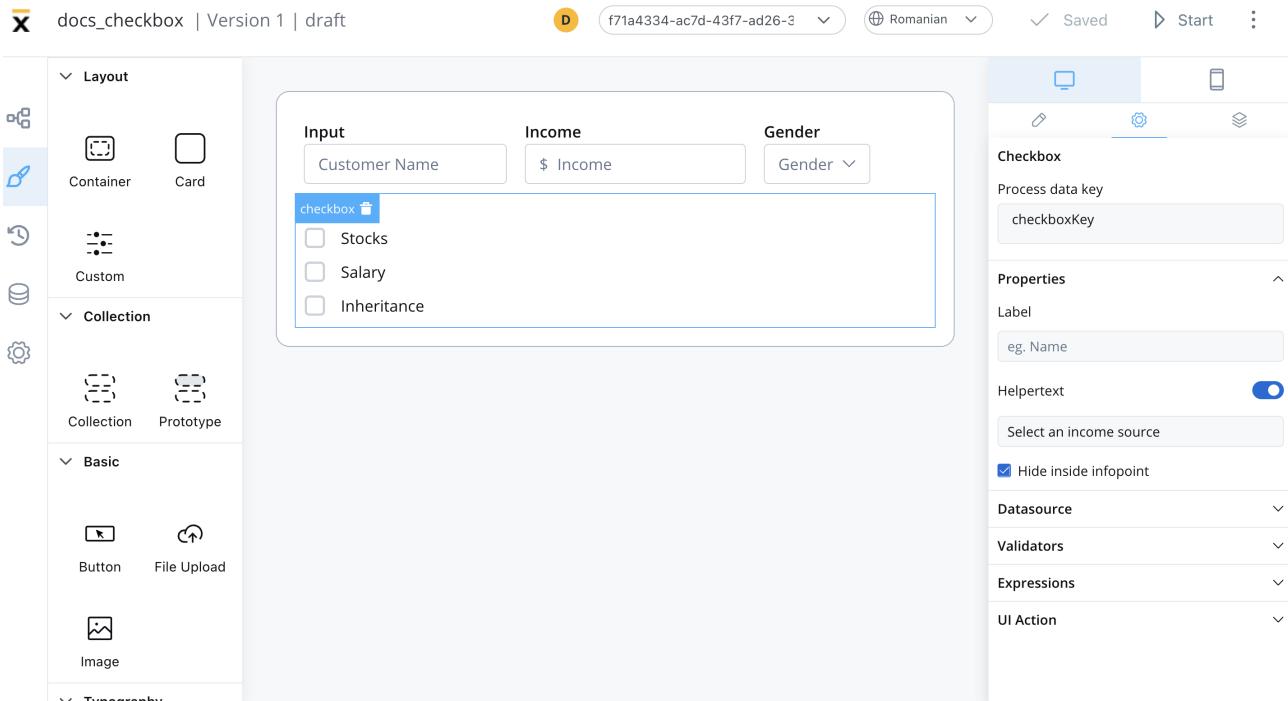
- [General](#)
- [Properties](#)
- [Datasource](#)
- [Validators](#)
- [Expressions](#)
- [UI actions](#)
- [Checkbox styling](#)

### General

- **Process data key** - creates the binding between form element and process data, so it can be later used in [decisions](#), [business rules](#) or [integrations](#)

### Properties

- **Label** - the label that appears on the checkbox
- **Helpertext** - additional information about the checkbox (can be hidden inside an infopoint)



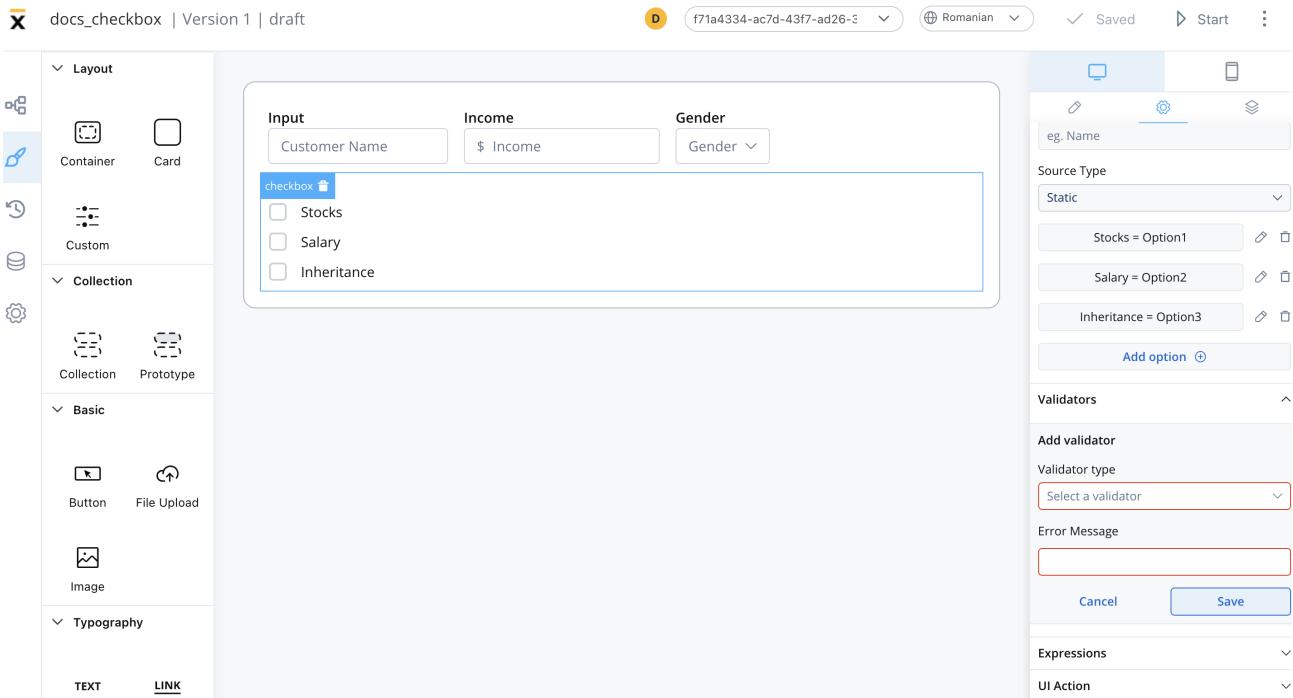
## Datasource

- **Default Value** - the default value of the checkbox
- **Source Type** - it can be Static, Enumeration, or Process Data
- **Add option** - label - value pairs can be defined here

The screenshot shows the FLOWX.AI UI Designer interface. On the left is a sidebar with categories like Layout, Collection, Basic, and others, each containing icons for different UI components. The main workspace displays a form with three input fields: 'Customer Name' (text), '\$ Income' (text), and 'Gender' (dropdown). Below these is a section titled 'checkbox' containing three options: 'Stocks', 'Salary', and 'Inheritance', each with a corresponding checkbox. To the right of the workspace is a preview area showing mobile and desktop versions of the form. The 'Properties' panel on the far right is expanded, showing settings for the 'checkbox' component. A red box highlights the 'Datasource' section, which contains three static options: 'Stocks = Option1', 'Salary = Option2', and 'Inheritance = Option3'. There is also a button labeled 'Add option'.

## Validators

The following validators can be added to a checkbox: `required` and `custom` (more details [here](#)).



## Expressions

The checkbox behavior can be defined using JavaScript expressions for hiding or disabling the element. The following properties can be configured for expressions:

- **Hide** - JavaScript expression used to hide the checkbox when it returns a truthy value
- **Disabled** - JavaScript expression used to disable the checkbox when it returns a truthy value

### INFO

It's important to make sure that disabled fields have the same expression configured under the path expressions → hide.

## UI actions

UI actions can be added to the checkbox element to define its behavior and interactions.

- **Event** - possible value: `CHANGE`
- **Action Type** - select the action type

 **INFO**

For more details on how to configure a UI action, click [here](#).

## Checkbox styling

The type of the checkbox can be selected by using the **styling** tab in **UI Designer**, possible values:

- clear
- bordered

 **INFO**

For more valid CSS properties, click [here](#).

A clear checkbox element with three options added, and a column layout will look like as it follows:

**✓ First Form Title**

First Form subtitle

Input 1

\$ Income

Input 3

.000

Gender

▼

- Stocks
- Salary
- Inheritance

**Was this page helpful?**

# BUILDING BLOCKS / UI Designer / UI component types / Form elements / Radio

The screenshot shows the FLOWX.AI interface. On the left, there's a sidebar with navigation links for Processes (Definitions, Active process, Process Instances, Failed process start) and Content Management (Enumerations, Substitution tags, Content models, Languages, Source systems, Media Library). The main area displays a form titled "Form title". The form contains three input fields: "Customer" (with "Customer Name" placeholder), "Income" (with "\$ Income" placeholder), and "Gender" (with a dropdown menu). Below these are two radio buttons: "Contact via Email" and "Contact via SMS", where "Contact via SMS" is selected. A blue "Submit" button is at the bottom of the form.

Radio buttons are normally presented in radio groups (a collection of radio buttons describing a set of related options). Only one radio button in a group can be selected at the same time.

## Configuring the radio field element

### Radio settings

The available configuration options for this form element are:

- **General**
- **Properties**
- **Datasource**
- **Validators**
- **Expressions**
- **UI actions**
- **Radio styling**

#### General

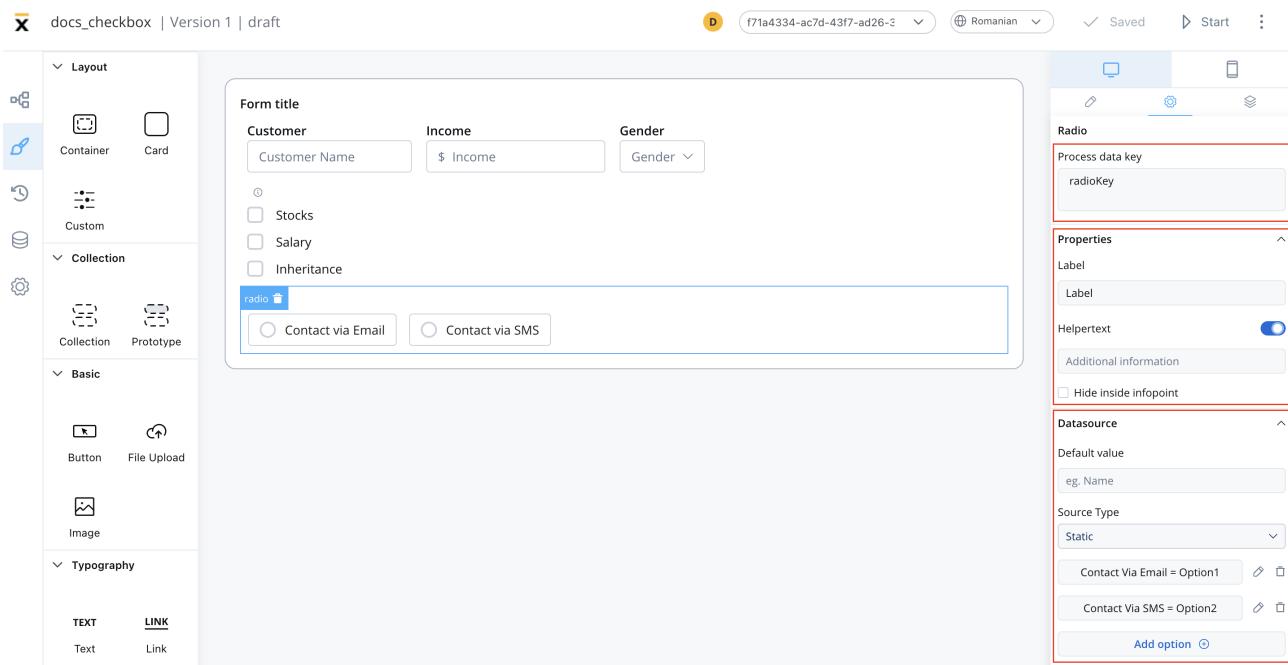
- **Process data key** - creates the binding between form element and process data so it can be later used in **decisions**, **business rules** or **integrations**

#### Properties

- **Label** - the label that appears on the radio
- **Helpertext** - additional information about the radio (can be hidden inside an infopoint)

## Datasource

- **Default Value** - the default values of the radio element
- **Source Type** - it can be Static, Enumeration, or Process Data
- **Add option** - label - value pairs can be defined here



## Validators

The following validators can be added to a radio: `required` and `custom` (more details [here](#))

## Expressions

The radio's element behavior can be defined using JavaScript expressions for hiding or disabling the element. The following properties can be configured for expressions:

- **Hide** - JavaScript expression used to hide the Radio element when it returns a truthy value
- **Disabled** - JavaScript expression used to disable the Radio element when it returns a truthy value

### !(INFO)

It's important to make sure that disabled fields have the same expression configured under the path expressions → hide.

The screenshot shows the FLOWX.AI UI Designer interface. On the left, there is a sidebar with categories: Layout (Container, Card, Custom), Collection (Collection, Prototype), and Basic (Button, File Upload, Image). The main area displays a form titled "Form title". The form contains three input fields: "Customer" (Customer Name), "Income" (\$ Income), and "Gender" (Gender dropdown). Below these is a section with a radio button labeled "radio". Underneath the radio button are two options: "Contact via Email" and "Contact via SMS". To the right of the form, there is a panel for configuring the "Radio" component. It includes fields for "Process data key" (radioKey) and "Validators" (Required). The "Expressions" section is highlighted with a red border and contains entries for "Hide" and "Disabled".

## UI actions

UI actions can be added to the radio element to define its behavior and interactions.

- **Event** - possible value: CHANGE
- **Action Type** - select the action type

**!** INFO

For more details on how to configure a UI action, click [here](#).

## Radio styling

The type of the radio can be selected by using the **styling** tab in **UI Designer**, possible values:

- clear
- bordered

**!** INFO

For more valid CSS properties, click [here](#).

A Radio element with two options added, and with a layout configuration set to horizontal will look like as it follows:

**✓ First Form Title**

First Form subtitle

Input 1      \$ Income      Input 3 .000      Gender

Stocks  
 Salary  
 Inheritance

Contact via Email     Contact via SMS

**Was this page helpful?**

# BUILDING BLOCKS / UI Designer / UI component types / Form elements / Switch

The form interface includes the following fields:

- Form title:** Customer, Income, Gender.
- Customer:** Customer Customer.
- Income:** \$ 999999999.
- Gender:** Gender dropdown.
- Contact methods:** Two radio buttons: "Contact via Email" (selected) and "Contact via SMS".
- Date selection:** A date picker showing 09.02.1972.
- Newsletter subscription:** A question "Do you want to subscribe to our newsletter?" followed by a toggle switch. The switch is in the 'on' position.
- Submit button:** A blue "Submit" button at the bottom left.

A switch, a toggle switch, is another form element that can be utilized to create an intuitive user interface. The switch allows users to select a response by toggling it between two states. Based on the selection made by the user, the corresponding Boolean value of either true or false will be recorded and stored in the process instance values for future reference.

## Configuring the switch element

### Switch settings

The available configuration options for this form element are:

- General
- Properties
- Datasource
- Validators
- Expressions
- UI actions
- Switch styling

## General

- **Process data key** - creates the binding between form element and process data so it can be later used in decisions, business rules or integrations

## Properties

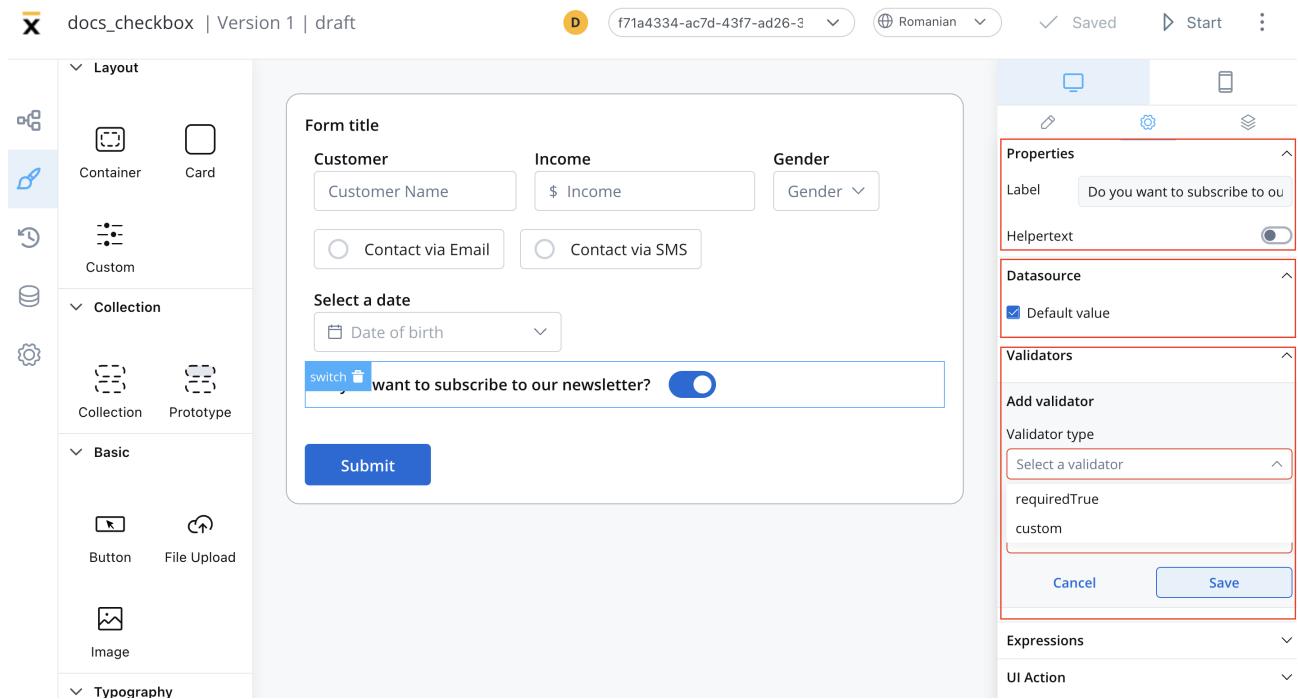
- **Label** - the label of the switch
- **Helpertext** - additional information about the switch element (can be hidden inside an infopoint)

## Datasource

- **Default Value** - the default value of the switch form field (it can be switched on or switched off)

## Validators

The following validators can be added to a switch element: `requiredTrue` and `custom` (more details [here](#)).



## Expressions

- **Hide** - JavaScript expression used to hide the Switch element when it returns a truthy value
- **Disabled** - JavaScript expression used to disable the Switch element when it returns a truthy value

## UI actions

UI actions can be added to the switch element to define its behavior and interactions.

- Event - possible value: CHANGE
- Action Type - select the action type

### INFO

For more details on how to configure a UI action, click [here](#).

## Switch styling

The label of the switch can be positioned either as `start` or `end`.

The screenshot shows the FLOWX.AI UI Designer interface. On the left, there's a sidebar with categories like Layout, Custom, Collection, Basic, and Media. In the center, a form titled "Form title" is displayed. It includes fields for "Customer Name", "Income", and "Gender". Below these are two radio buttons for "Contact via Email" and "Contact via SMS". A date picker for "Date of birth" is also present. At the bottom of the form is a switch component with the label "Do you want to subscribe to our newsletter?". To the right of the form is a panel titled "Switch" with sections for Properties, Sizing, Spacing, Background, Typography, and Advanced. The "Label position" dropdown is set to "end".

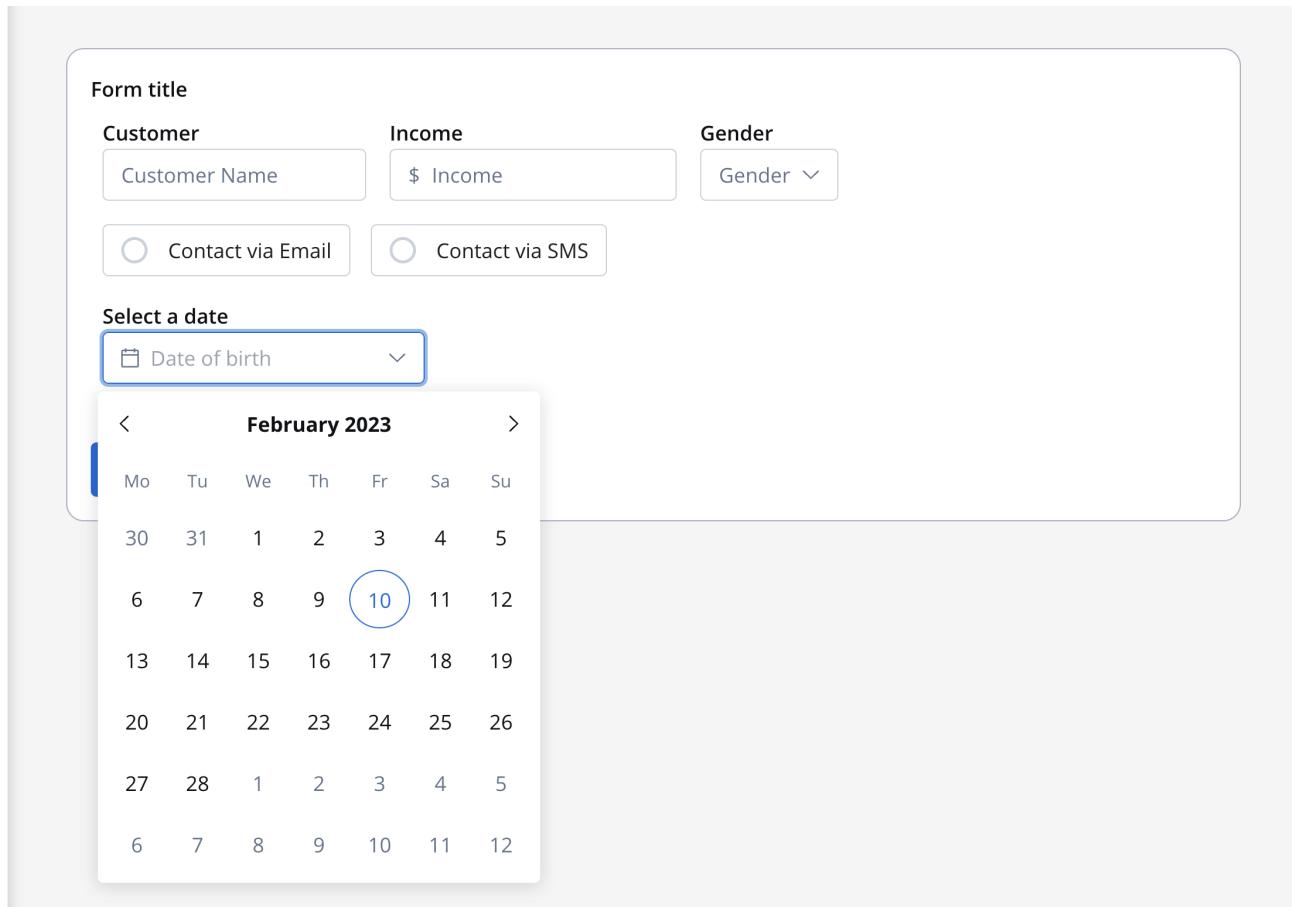
### INFO

For more valid CSS properties, click [here](#).

Was this page helpful?

**BUILDING BLOCKS / UI Designer / UI component types / Form elements /**

# Datepicker



The datepicker (Calendar Picker) is a lightweight component that allows end users to enter or select a date value.

## INFO

The default datepicker value is `DD.MM.YYYY`.

## Configuring the datepicker element

## Datepicker settings

The available configuration options for this form element are:

- [General](#)
- [Properties](#)
- [Datasource](#)
- [Validators](#)
- [Expressions](#)
- [UI actions](#)
- [Datepicker styling](#)

### General

- **Process data key** - creates the binding between form element and process data so it can be later used in [decisions](#), [business rules](#) or [integrations](#)

### Properties

- **Label** - the label of the datepicker
- **Placeholder** - placeholder when the field has no value
- **Min Date** - set the minimum valid date selectable in the datepicker
- **Max Date** - set the maximum valid date selectable in the datepicker
- **Min Date, Max Date error** - when a date is introduced by typing, define the error message to be displayed
- **Helpertext** - additional information about the input field (can be hidden inside an infopoint)

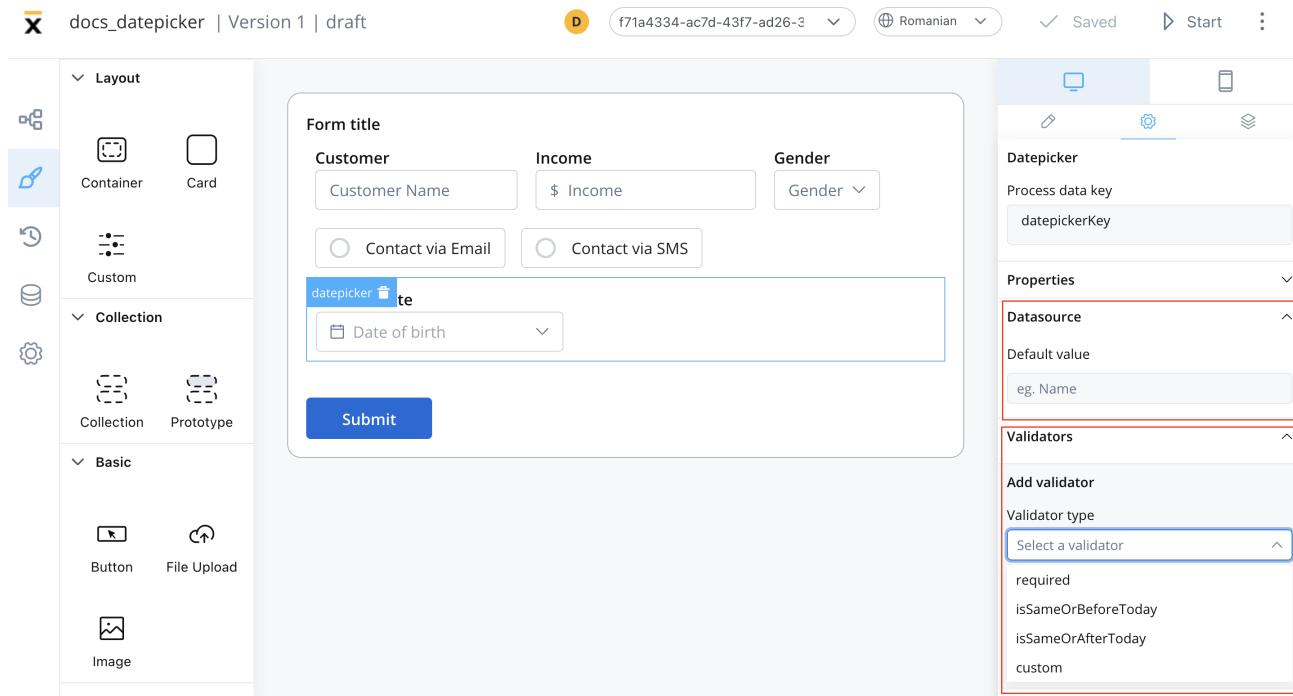
The screenshot shows the FLOWX.AI UI Designer interface. On the left, there's a sidebar with categories like Layout, Collection, and Basic, each containing icons for Container, Card, Custom, Collection, Prototype, Button, File Upload, and Image. The main area displays a form titled "Form title" with fields for Customer Name, Income, Gender, and Contact methods (Email or SMS). A Datepicker element is highlighted with a blue border. To the right, a properties panel is open for the Datepicker component, showing fields for Process data key (datepickerKey), Label (Select a date), Placeholder (Date of birth), Min date and Max date (both set to dd.mm.yyyy), Min date error, Max date error, and Helpertext (with a toggle switch). The entire properties panel is enclosed in a red border.

## Datasource

- **Default Value** - the default values of the datepicker element, this will autofocus the datepicker when you will run the process

## Validators

The following validators can be added to a datepicker: `required`, `custom`, `isSameOrBeforeToday` or `isSameOrAfterToday` (more details [here](#)).



## Expressions

The datepicker behavior can be defined using JavaScript expressions for hiding or disabling the element. The following properties can be configured for expressions:

- **Hide** - JavaScript expression used to hide the datepicker when it returns a truthy value
- **Disabled** - JavaScript expression used to disable the datepicker when it returns a truthy value

### INFO

It's important to make sure that disabled fields have the same expression configured under the path expressions → hide.

## UI actions

UI actions can be added to the datepicker element to define its behavior and interactions.

- **Event** - possible value: CHANGE
- **Action Type** - select the action type

### INFO

For more details on how to configure a UI action, click [here](#).

The screenshot shows the FLOWX.AI UI Designer interface. On the left, there's a sidebar with categories like Layout, Collection, and Basic, each containing icons for Container, Card, Custom, Collection, Prototype, Button, File Upload, and Image. The main area displays a form titled "Form title". The form contains fields for "Customer Name", "Income", "Gender", and two radio buttons for "Contact via Email" and "Contact via SMS". Below these is a datepicker field labeled "Date of birth". At the bottom is a blue "Submit" button. To the right of the form is a properties panel. The "Expressions" section is expanded, showing "Hide" and "Disabled" dropdown menus, both currently empty. The "UI Action" section is also expanded, showing "Event" set to "CHANGE" and "Action Type" set to "Select an action type". There are "Cancel" and "Save" buttons at the bottom of the properties panel. The top of the screen shows the project name "docs\_datepicker", version "Version 1", and status "draft". It also includes a language dropdown set to "Romanian", a save icon, a start icon, and a more options icon.

## Datepicker styling

The styling of a datepicker element can be customized in various ways using CSS properties like typography color, border-radius/width, or advanced CSS params. This allows you to create a datepicker that fits seamlessly with the overall design of the application you are developing.

**!** INFO

For more valid CSS properties, click here.

Was this page helpful?

# BUILDING BLOCKS / UI Designer / UI component types / Form elements / Slider

The screenshot shows a 'Form title' component with the following fields:

- Form title:** Enter Personal Information
- First Name:** John
- Last Name:** Doe
- Date of birth:** 18.12.1995
- Income range:** A segmented button with options: < \$50.000, \$50.000 - \$100.000, \$100.000 - \$150.000, > \$150.000. The first option is selected.
- Rate your experience with us:** A horizontal slider with a scale from 1 to 5. The slider is positioned at 1.
- Save personal information:** A toggle switch that is turned on.
- Submit:** A blue button.

It allows users to pick only one option from a group of options, and you can choose to have between 2 and 5 options in the group. The segmented button is

easy to use, and can help make your application easier for people to use.

## Configuring the slider element

### Slider settings

The available configuration options for this form element are:

- **General**
- **Properties**
- **Datasource**
- **Validators**
- **Expressions**
- **UI actions**
- **Slider styling**

#### General

- **Process data key** - creates the binding between form element and process data so it can be later used in **decisions**, **business rules** or **integrations**

#### Properties

- **Label** - the label of the slider
- **Show value label** - a toggle option that determines whether the current selected value of the slider is displayed as a label alongside the slider handle
- **Helptext** - an optional field that provides additional information or guidance related to the usage or function of the slider, it can be hidden within an

infopoint, which users can expand or access for further detail

- **Min Value** - the minimum value or starting point of the slider's range, it defines the lowest selectable value on the slider
- **Max Value** - the maximum value or end point of the slider's range, it sets the highest selectable value on the slider
- **Suffix** - an optional text or symbol that can be added after the displayed value on the slider handle or value label, it is commonly used to provide context or units of measurement
- **Step size** - the increment or granularity by which the slider value changes when moved, it defines the specific intervals or steps at which the slider can be adjusted, allowing users to make more precise or discrete value selections

## Datasource

- **Default Value** - the default value of the slider (static value - integer) the initial value set on the slider when it is first displayed or loaded, it represents a static value (integer), that serves as the starting point or pre-selected value for the slider, users can choose to keep the default value or adjust it as desired

The screenshot shows the FLOWX.AI UI Designer interface. On the left, there's a sidebar with various UI component icons categorized under 'Layout', 'Forms', and 'Collection'. In the center, there's a form titled 'Enter Personal Information' with fields for First Name, Last Name, and Date of birth. Below these are sections for Employment type (radio buttons for Employed and Pensioner), Income range (a segmented slider with categories: < \$50.000, \$50.000 - \$100.000, \$100.000 - \$150.000, > \$150.000), and Loan amount (a slider with a value of 55000 € and a step size of 5000). At the bottom, there's a 'Save personal information' toggle switch and a 'Submit' button. On the right, there's a detailed configuration panel for the 'Slider' component, showing settings for 'Process data key' (sliderKey), 'General properties' (Slider label, Show value label checked), and 'Datasource' (Default value: 20000).

## Validators

The following validators can be added to a slider: `required` and `custom` (more details [here](#)).

## UI actions

UI actions can be added to the slider element to define its behavior and interactions.

- **Event** - possible value: CHANGE
- **Action Type** - select the action type, ! for more details on how to configure a UI action, click [here](#)

## Multiple sliders

You can also use multiple sliders UI elements that are interdependent, as you can see in the following example:

The screenshot shows a user interface for entering personal information. On the left, there are icons for close, edit, info, and refresh. The main title is "Enter Personal Information".  
Form title: First Name (placeholder "First Name"), Last Name (placeholder "Doe"), Date of birth (placeholder "18.12.1995").  
Employment type: Radio buttons for "Employed" and "Pensioner".  
Save personal information: A toggle switch.  
Loan amount: A slider with values 10000 \$ and 500000 \$. The current value is 255000 \$.  
Down payment: A slider with values 38250 \$ and 89250 \$. The current value is 38250 \$.  
Loan type: A dropdown menu showing "Conventional".  
A large blue "Submit" button is at the bottom.

### INFO

You can improve the configuration of the slider using computed values as in the example above. These values provide a more flexible and powerful approach for handling complex use cases. You can find an example by referring to the following documentation:

## Dynamic & computed values

# Slider styling

To create a slider with specific styling, sizing, typography, and color settings, you can use the following configuration:

- **Sizing**
- **Typography**
- **Background**

## Sizing

- set the width of the button - fill/fixed/auto

## Typography

Choose an appropriate font family, size, and weight for the button label text.

- **Label Color** - set the color of the button label text
- **Min/Max values** - set the color of
- **Result** - set the color of the

## Background

- **Selected BG** - set the background color of the button.
- **ComponentBg** - set the background color of the button.

The screenshot shows the FLOWX.AI UI Designer interface. On the left is a sidebar with categories like Layout, Forms, and Collections, each containing various UI components. The main area displays a form titled "Enter Personal Information". This form includes fields for First Name, Last Name, Date of birth, Employment type (radio buttons for Employed and Pensioner), Income range (a segmented button with options < \$50.000, \$50.000 - \$100.000, \$100.000 - \$150.000, and > \$150.000), and a Slider for Loan amount ranging from 10000 € to 100000 €. A "Save personal information" toggle switch is also present. A "Submit" button is at the bottom. To the right of the form is a detailed component editor for the Slider, which is highlighted with a red box. The editor shows settings for Sizing (Fit W: fill), Spacing (0 to 16), Typography (Label, Min/Max values, Result), Background (Selected BG, ComponentBg), and Advanced (Add class). The component editor has tabs for Desktop and Mobile viewports.

### INFO

For more valid CSS properties, click [here](#).

Was this page helpful?

# BUILDING BLOCKS / UI Designer / UI component types / Form elements / Segmented button

The screenshot shows a user interface for entering personal information. On the left, there is a vertical toolbar with icons for close, edit, info, and eye. The main area has a title 'Enter Personal Information'. It includes fields for 'First Name' (John), 'Last Name' (Doe), and 'Date of birth' (18.12.1995). Below these are four options for 'Income range': '< \$50.000', '\$50.000 - \$100.000', '\$100.000 - \$150.000', and '> \$150.000'. A 'Save personal information' toggle switch is turned on. At the bottom is a blue 'Submit' button.

It allows users to pick only one option from a group of options, and you can choose to have between 2 and 5 options in the group. The segmented button is easy to use, and can help make your application easier for people to use.

## Configuring the segmented button

### Segmented button settings

The available configuration options for this form element are:

- **General**
- **Properties**
- **Datasource**
- **Validators**
- **Expressions**

- **UI actions**
- **Segmented button styling**

## General

- **Process data key** - creates the binding between form element and process data so it can be later used in **decisions**, **business rules** or **integrations**

## Properties

- **Label** - the label of the segmented button
- **Helpertext** - additional information about the segmented button (can be hidden inside an infopoint)

## Datasource

- **Default Value** - the default value of the segmented button (it can be selected from one of the static source values)
- **Source Type** - it is by default Static
- **Add option** - value/label pairs can be defined here

## Validators

The following validators can be added to a segmented button: **required** and **custom** (more details [here](#)).

The screenshot shows the FLOWX.AI UI Designer interface. On the left, there's a sidebar with various UI component categories and their icons. In the center, there's a form titled "Enter Personal Information" with fields for First Name, Last Name, and Date of birth. Below these is a "Income range" section with a segmented button. At the bottom, there's a "Save personal information" toggle and a "Submit" button. On the right, there's a properties panel for the selected "Segmented Button" component. The properties panel has sections for "Properties" (with "Label" set to "Income range" and "Helpertext" checked), "Datasource" (with a default value of "eg. Name" and a static source mapping income ranges to options), and "Validators" (with a placeholder "Add a validator").

## UI actions

UI actions can be added to the segmented button element to define its behavior and interactions.

- **Event** - possible value: CHANGE
- **Action Type** - select the action type

### INFO

For more details on how to configure a UI action, click [here](#).

## Segmented button styling

To create a segmented button with specific styling, sizing, typography, and color settings, you can use the following configuration:

- **Sizing**
- **Typography**
- **Background**

## Sizing

- set the width of the button - fill/fixed/auto

## Typography

Choose an appropriate font family, size, and weight for the button label text.

- **Label Color** - set the color of the button label text
- **Selected State** - set the color of the label text when the button is selected
- **Unselected State** - set the color of the label text when the button is not selected

## Background

- **Selected state** - set the background color of the button
- **Unselected state** - set the background color of the button

The screenshot shows the FLOWX.AI UI Designer interface. On the left is a sidebar with categories like Layout, Forms, and Collections, each containing icons for different UI components. The main area displays a form titled "Enter Personal Information". The form includes fields for First Name, Last Name, Date of birth, and an income range selector. Below the form is a "Save personal information" toggle and a "Submit" button. To the right of the form are several styling panels: "Sizing" (with "Fit W" set to "auto"), "Spacing" (with grid settings), "Typography" (with options for Label color, Selected, and Unselected states), "Background" (with options for Selected and Unselected states), "Border" (with Radius and Width settings), and "Advanced" (with a "Set color" button). Red boxes highlight the "Sizing", "Typography", and "Background" panels.

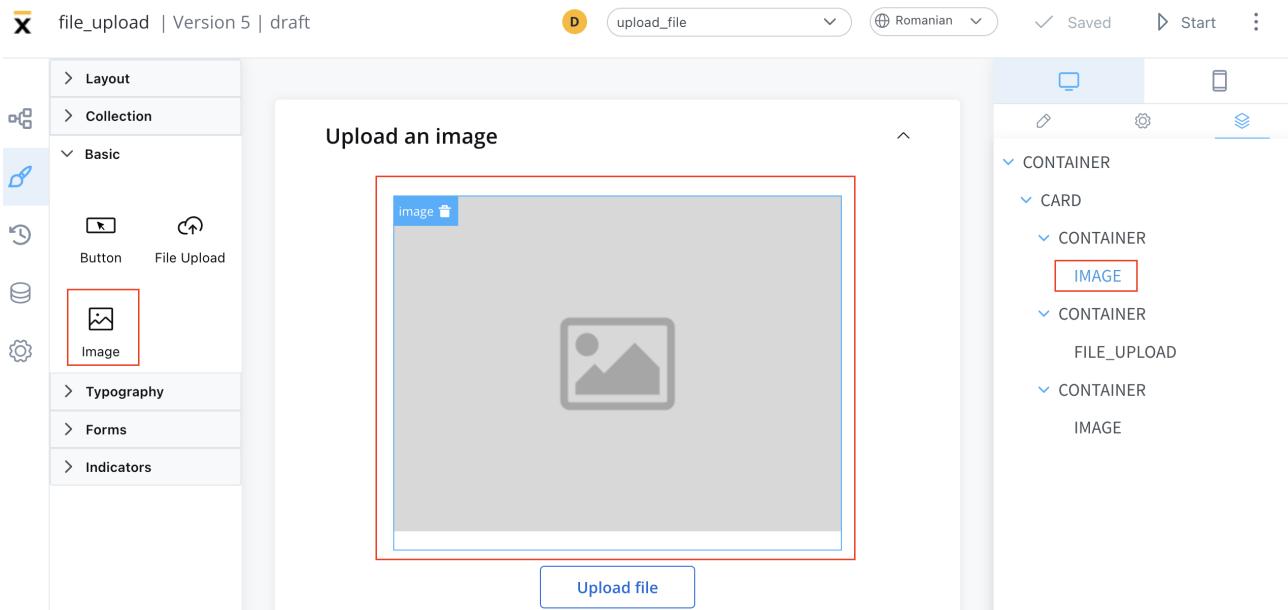
### INFO

For more valid CSS properties, click [here](#).

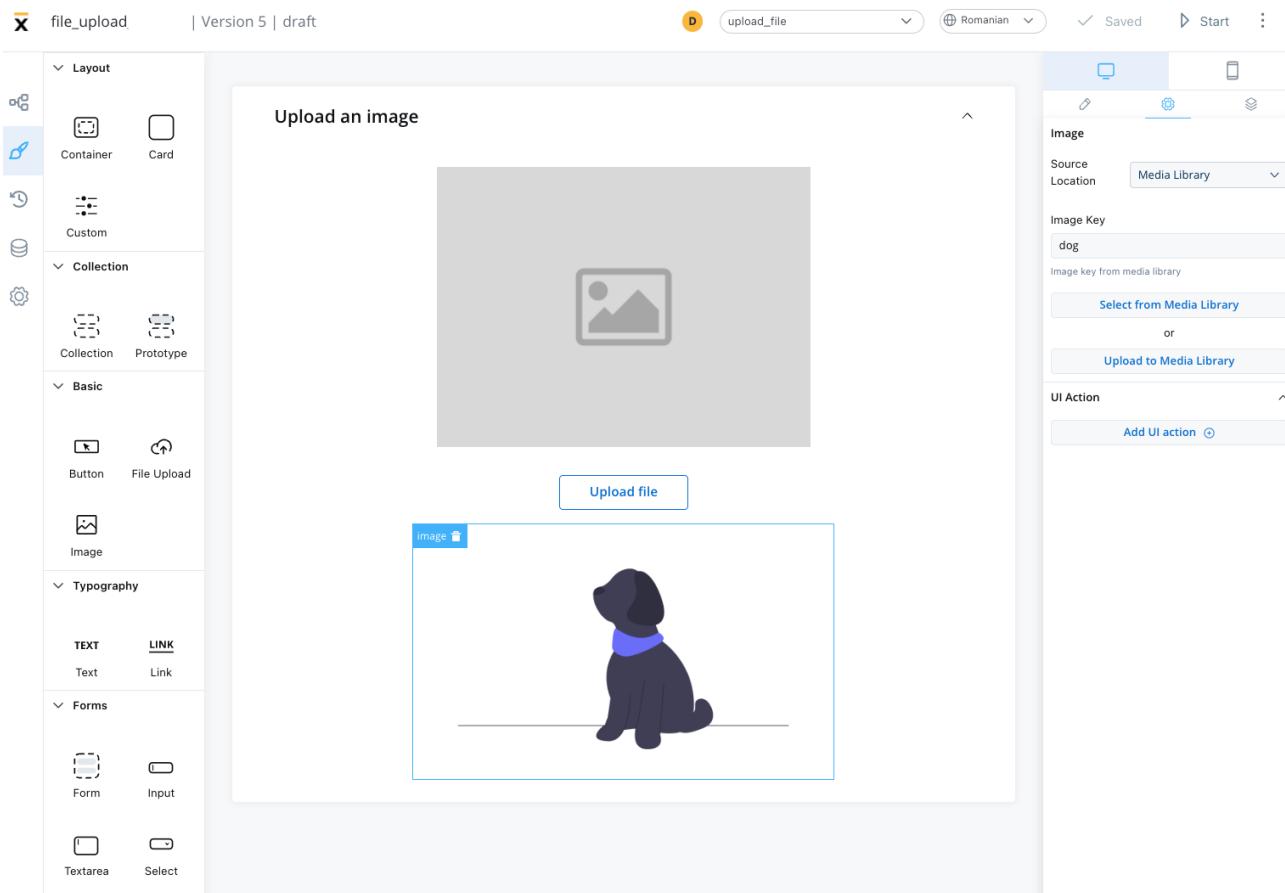
Was this page helpful?

# BUILDING BLOCKS / UI Designer / UI component types / Image

Image UI elements are graphical components of a user interface that display a static or dynamic visual representation of an object, concept, or content.



These elements can be added to your interface using the UI Designer tool, and they are often used to convey information, enhance the aesthetic appeal of an interface, provide visual cues and feedback, support branding and marketing efforts, or present complex data or concepts in a more intuitive and accessible way.



## Configuring an image

Configuring an image in the UI Designer involves specifying various settings and properties. Here are the key aspects of configuring an image:

### Image settings

The image settings consist of the following properties:

- **Source location** - the location from where the image is loaded:
  - **Media Library**

- o **Process Data**
- o **External**

Depending on which **Source location** is selected, different configurations are available:

## Media library

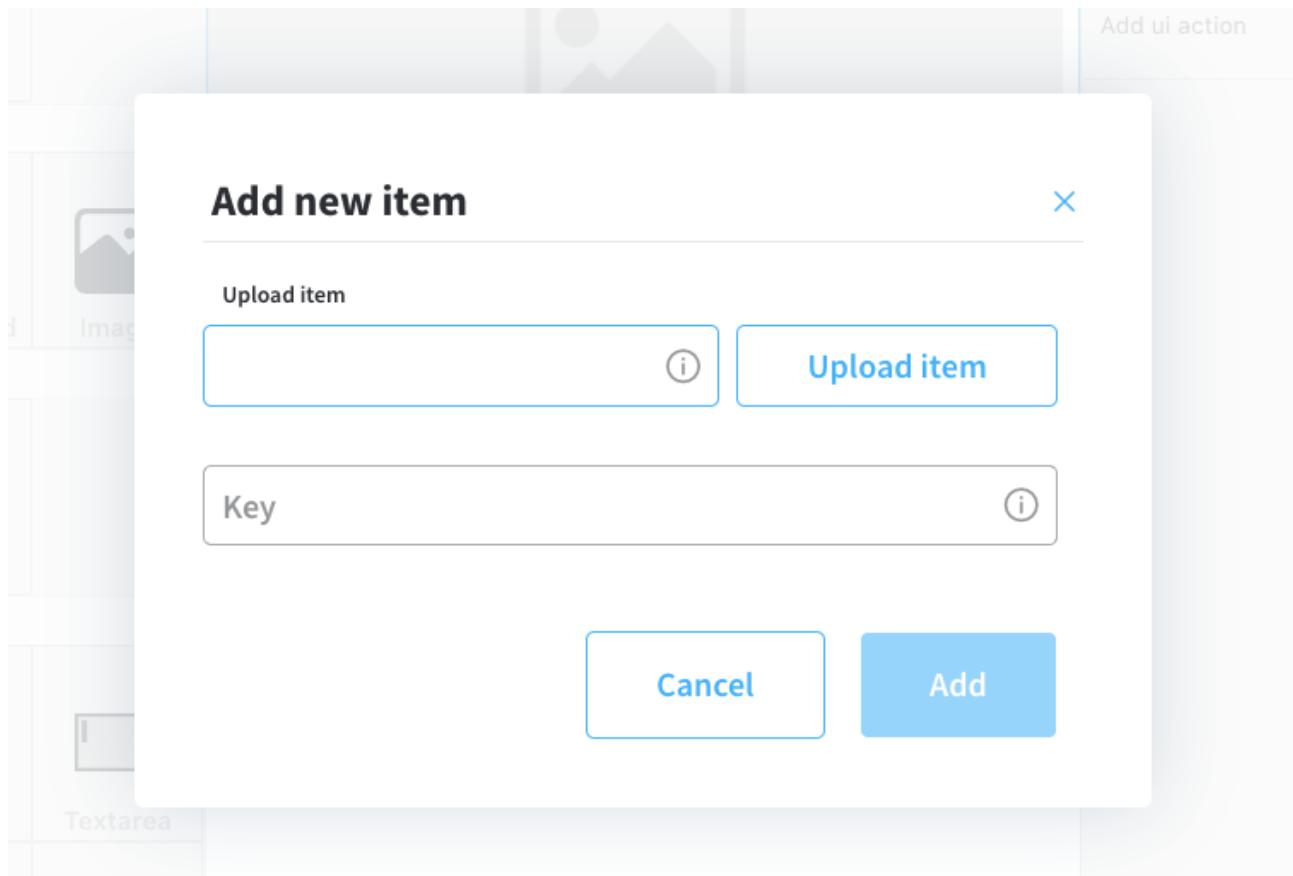
The screenshot shows the FLOWX.AI UI Designer interface. On the left, there is a sidebar with various building block categories: Layout, Collection, Basic, Typography, Forms, and Media. Under the Media category, the 'Image' icon is selected. In the center, there is a component preview area titled 'Upload an image'. Below it is a placeholder image of a dog. At the bottom right of this area is a blue 'Upload file' button. To the right of the component preview is a detailed configuration panel. This panel includes tabs for 'Image', 'Source Location', 'Image Key', and 'UI Action'. The 'Image' tab is active, showing a dropdown menu set to 'Media Library'. The 'Source Location' dropdown also shows 'Media Library'. The 'Image Key' field contains the value 'dog'. Below these fields are buttons for 'Select from Media Library' and 'Upload to Media Library'. The 'UI Action' section has a single option: 'Add UI action'. A red box highlights the 'Source Location' and 'Image Key' sections.

- **Image key** - the key of the image from the media library

- **Select from media library** - search for an item by key and select it from the media library

Preview	Key	Format	Size	Edited at	Edited by
	video_file	png	20.58 KB	23 Feb 2023, 11:29 AM	John Doe
	switches	png	0.03 MB	23 Feb 2023, 11:28 AM	John Doe
	form_example	png	22.29 KB	23 Feb 2023, 11:27 AM	John Doe
	avatar2	png	0.03 MB	23 Feb 2023, 11:26 AM	John Doe
	avatar1	png	27.73 KB	23 Feb 2023, 11:26 AM	John Doe
	cat	png	15.63 KB	22 Feb 2023, 1:18 PM	John Doe
	dog	png	20.65 KB	22 Feb 2023, 1:18 PM	John Doe

- **Upload to media library** - add a new item (upload an image on the spot)
  - **upload item** - supported formats: PNG, JPG, GIF, SVG, WebP;  
! (maximum size - 1 MB)
  - **key** - the key must be unique and cannot be changed afterwards



## Process Data

The screenshot shows the FLOWX.AI UI Designer interface. On the left, there is a sidebar with various component categories: Layout, Collection, Basic, Typography, and Forms. Under 'Basic', the 'Image' component is selected, shown as a placeholder icon with a 'File Upload' button below it. In the center, a modal window titled 'Upload an image' is displayed, featuring a large input field with a placeholder 'image' and a 'File Upload' button at the bottom. To the right of the modal, a detailed configuration panel for the 'Image' component is visible, with tabs for 'Image', 'Text', and 'UI Action'. The 'Image' tab is active, showing settings for 'Source Location' (Process Data), 'Source Type' (URL), 'Process Data Key' (app.image), and 'Placeholder Url' (Public url). A 'UI Action' section at the bottom has an 'Add UI action' button.

- Identify the **Source Type**. It can be either a **URL** or a **Base 64 string**.
- Locate the data using the **Process Data Key**.
- If using a URL, provide a **Placeholder URL** for public access. This is the URL where the image placeholder is available.

file\_upload | Version 5 | draft      upload\_file      Romanian      Saved      Start      :

**Layout**

- Container
- Card

**Custom**

**Collection**

- Collection
- Prototype

**Basic**

- Button
- File Upload

**Image**

Source Location: Process Data

Source Type: URL

Process Data Key: app.image

Placeholder Url: Public url

**UI Action**

Add UI action

Upload an image

Upload file

Image

Source Location: Process Data

Source Type: URL

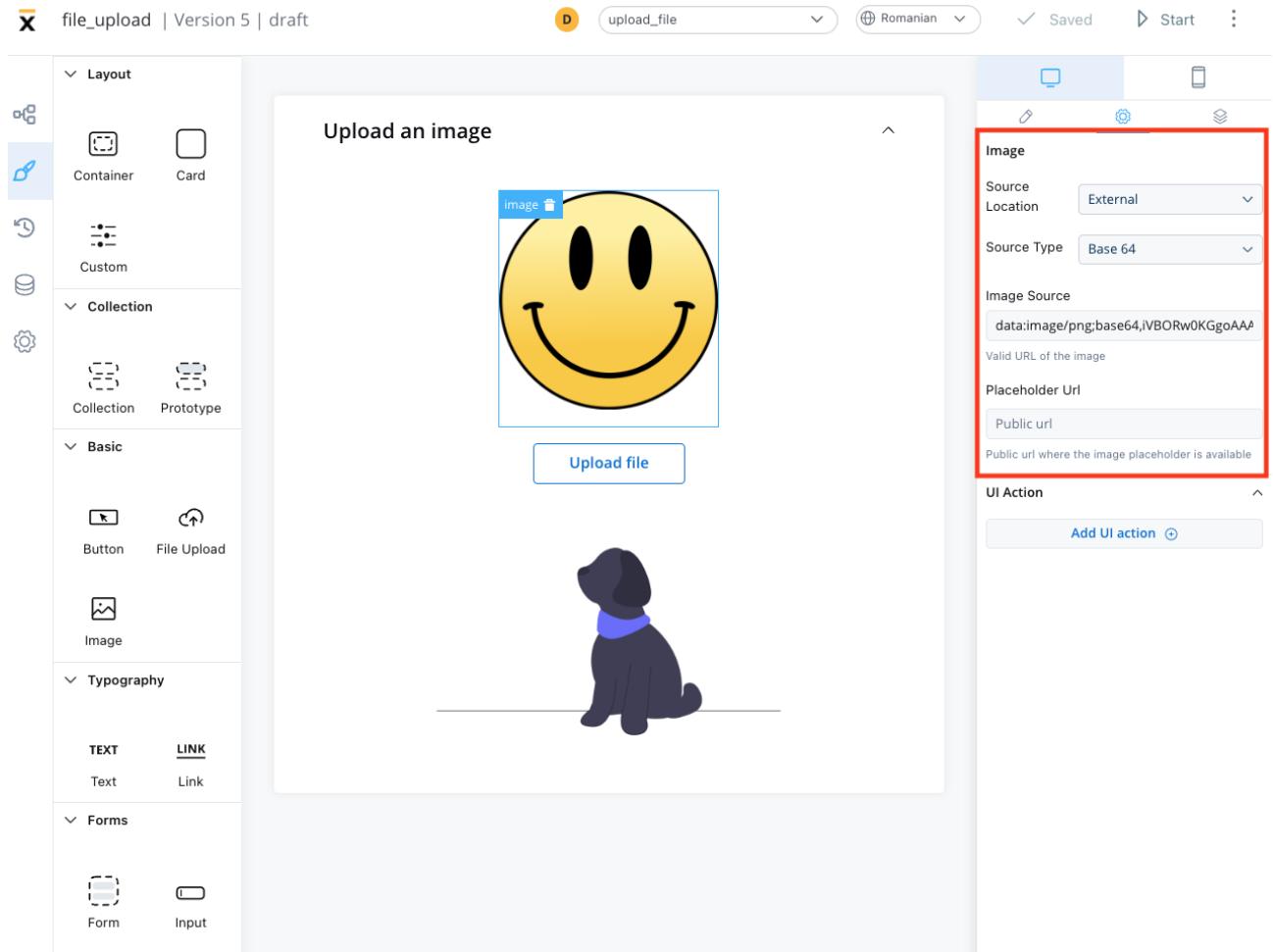
Process Data Key: app.image

Placeholder Url: Public url

UI Action

Add UI action

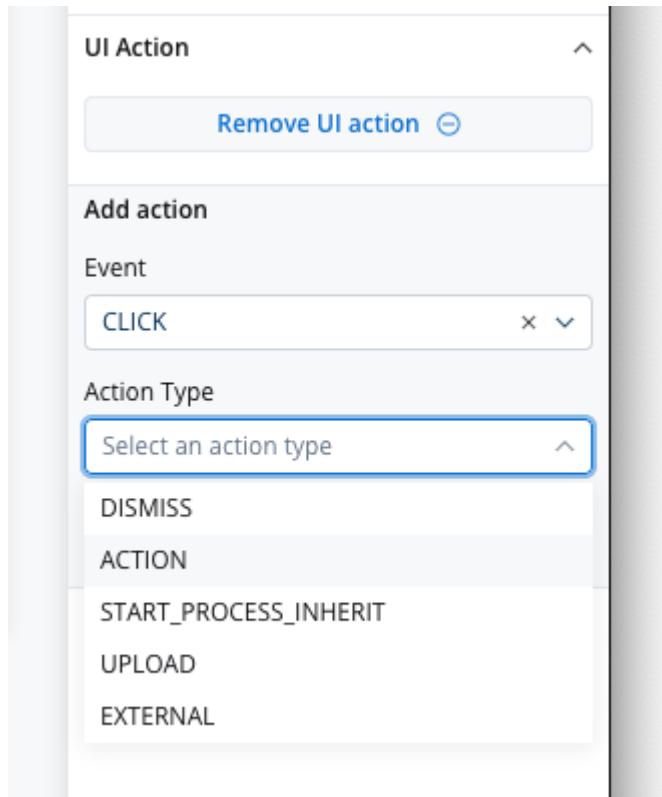
## External



- **Source Type:** it can be either a **URL** or a **Base 64 string**
- **Image source:** the valid URL of the image.
- **Placeholder URL:** the public URL where the image placeholder is available

## UI actions

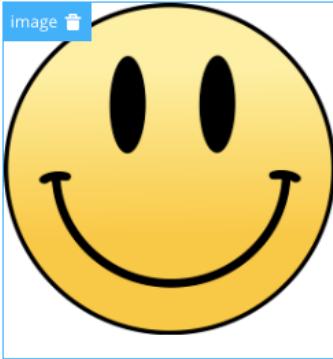
The UI actions property allows you to add a UI Action, which must be configured on the same node. For more details on UI Actions, refer to the documentation [here](#).



## Image styling

The image styling property allows you to add or to specify valid CSS properties for the image. For more details on CSS properties, click [here](#).

Upload an image



Upload file



Image

Sizing

Fit W: fixed 220 px

Fit H: auto

Spacing

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
16			

Border

Radius: 0 px

Width: 0 px

Color: #1E1E1C

Advanced

Add class: Comma separated class names

Was this page helpful?