Form container | WeWeb Documentation

|  |  |
| --- | --- |
| URL source | <https://docs.weweb.io/elements/form-container.html> |
| Date de scraping | 2025-04-08T15:53:55.974864+00:00 |

## Form container ​

Forms are the backbone of user interaction in web applications. Form containers in WeWeb provide powerful tools to collect and validate user information. Whether you're building checkout flows, job applications, or product configurators, WeWeb's forms help you create effective user experiences while ensuring data integrity.

## Add a form container ​

To add a form container in WeWeb, go to Add > Basics and search for "form":

### Images

<https://docs.weweb.io/assets/form-container1.D0Hvh-KA.png>

## Understanding form variables ​

When you add a form container to your page, you'll have access to these variables:  
  
isSubmitting  
This variable tells you if the form is currently in the process of being submitted. It will be:  
false when the form is idle true during the brief moment when a user has clicked submit and the form is processing  
This is useful for showing loading states or preventing double submissions.  
isSubmitted  
This variable indicates whether the form has been submitted at least once. It will be:  
false when the form hasn't been submitted yet true after the form has been submitted, regardless of whether the submission was successful  
You can use this to show different content or messages after a user has attempted to submit the form.  
isValid  
This variable tells you about the current validation state of your form. It will be:  
null when the form hasn't been interacted with yet true when all required fields are filled and all validation rules are met false when any required field is empty or any validation rule fails  
This helps you control submit button states or show validation messages.  
formData  
This is an object that contains all the current values from your form fields. For example, if you have fields named "email" and "password", you might see:  
fields  
This is an object that contains information about each form field, including their validation states:

### Code

Langage: unknown

{  
 "email": "user@example.com",  
 "password": "example1"  
}

Langage: unknown

{  
 "email": {  
 "value": "user@example.com",  
 "isValid": undefined  
 },  
 "password": {  
 "value": "example1",  
 "isValid": undefined  
 }  
}

### Images

<https://docs.weweb.io/assets/form2.B0_o0pGi.png>

## Form validation ​

Form validation is the process of checking if information entered into a form meets certain rules or requirements before it can be submitted.

## Common types of validation methods: ​

Required field: ensures essential fields aren't left blank Data format: checks inputs match patterns (emails need @, phone numbers need correct digits) Length: verifies text meets character limits (minimum password length, maximum descriptions) Range: confirms numbers fall within acceptable ranges (age limits, reasonable quantities) Comparison: checks relationships between fields (password confirmation matches, dates are sequential) Regular expression: uses pattern matching for complex formats (postal codes, credit cards) Custom logic: implements business rules (username availability, promotion code validity)

## Handling validation with variables ​

Form validation in WeWeb works on two levels - individual fields and the entire form.

## Field-level validation ​

Each input field in your form has its own isValid property that reflects whether that specific field meets all validation requirements:  
You can use the field-level isValid property to:  
Show an error message directly below a specific field Highlight invalid fields with a red border or background Display a checkmark icon next to correctly filled fields Enable/disable specific related fields based on validation status

### Code

Langage: unknown

{  
 "email": {  
 "value": "user@example.com",  
 "isValid": true // This field passes validation  
 },  
 "password": {  
 "value": "ex",  
 "isValid": false // This field fails validation (too short)  
 }  
}

## Form-level validation ​

The entire form has a global isValid property:  
true - all fields pass validation false - at least one field fails validation null - validation hasn't been evaluated yet  
You can use the form-level isValid property to:  
Enable/disable the submit button until all fields are valid Show an overall form error message at the top Display a progress indicator showing form completion status Trigger conditional visibility of subsequent form sections

## Validation timing settings ​

The Validation property in the Settings tab controls when validation happens:  
  
On input change : both field isValid values and the form's isValid update as you type. This will enable the Validation delay property On form submit : the isValid values only update when the submit button is clicked  
For example, with "On input change," a text field requiring at least 10 characters immediately sets its isValid to false if the input is too short, which in turn makes the form's overall isValid false . With "On form submit," validation happens only when the user attempts to submit.  
This system helps ensure data is correct before submission while giving you control over when users see validation messages.

### Images

<https://docs.weweb.io/assets/form-validation.B35mqZMm.png>

## Password validation example ​

In this example, we show an error message when a password is too long:  
  
Set form validation to "On input change" so validation happens while typing Add an error message text element below the password field Configure the error message's visibility to show only when the password field's isValid is false Set up the password field's validation rule: "is password smaller than 6 characters"  
This creates immediate feedback where the error message automatically appears when the password is too short and disappears when it's valid, all without requiring the user to submit the form.

### Images

<https://docs.weweb.io/assets/form-validation.BPewvMQo.gif>

## On Submit Validation Error Trigger ​

The On submit validation error trigger activates when a form is submitted but fails validation.  
  
While the standard On submit trigger only runs when all validation passes, this specialized trigger provides a way to handle validation failures.  
This allows you to create custom error handling workflows that display helpful messages, log specific validation issues, or guide users on how to correct their input.

### Images

<https://docs.weweb.io/assets/onsubmitvalidation.CuOPbrSH.png>

## Autocomplete ​

Form autocomplete: enables browser suggestions for all fields based on previously entered information. Input-level autocomplete: allows field-specific control over suggestions, letting you enable or disable autocomplete for individual fields regardless of the form-level setting.

## Submit a form ​

To allow users to submit the form, you'll need to add a workflow .  
The form submission workflow should be triggered On submit .

### Images

<https://docs.weweb.io/assets/form-container4.j3MGkmL6.png>

<https://docs.weweb.io/assets/form-container5.95UyWPAh.png>

## Workflow actions ​

When creating workflows within the Form container, two specialized actions become available:  
Set form state Submit form  
These actions can be used not only in workflows attached to the form itself, but also in workflows attached to any element inside the form container (like buttons, inputs, or custom components).

## Set form state ​

Controls form submission status variables:  
  
isSubmitting : toggle loading state (On/Off) isSubmitted : toggle submission status (On/Off)  
Use to manage loading indicators, create multi-step forms, or reset form state.

### Images

<https://docs.weweb.io/assets/form-workflow1.CVPmMBCN.png>

## Submit form ​

Programmatically submits the form, which:  
Runs all form validation Triggers the form's submission process Sets both submission variables appropriately  
  
Use for custom submission triggers or conditional submission logic.

### Images

<https://docs.weweb.io/assets/form-workflow2.CVkFBDdS.png>

## A practical workflow sequence for form submission: ​

⚡ On submit: user clicks the submit button Set isSubmitting to On : show loading spinner API call: send form data to your database (or simply add the Time delay action to simulate processing) Check API response:  
If successful: set isSubmitted to On , show success message If error: keep isSubmitted Off , show error message  
Set isSubmitting to Off : hide loading spinner  
This simple flow provides feedback throughout the submission process while handling both success and error cases in a user-friendly way:

### Images

<https://docs.weweb.io/assets/form-validation1.D6kfKf5q.gif>

## Form input states ​

To improve your web app's accessibility and user experience, it's best practice to add states to form inputs. States help provide visual feedback to users based on form conditions.  
A common example is disabling the submit button until all form fields are valid:  
  
In the example above, we're using the form's isValid property to control the submit button's disabled state. When the form validation fails, the button remains disabled, preventing submission attempts with invalid data.  
You can apply similar state controls to highlight invalid fields, show success indicators, or create other interactive elements that respond to the form's validation state.  
Learn more about adding states in WeWeb .

### Images

<https://docs.weweb.io/assets/form-state.BzLL8SF7.gif>