Intro to components | WeWeb Documentation

|  |  |
| --- | --- |
| URL source | <https://docs.weweb.io/components/intro-to-components.html> |
| Date de scraping | 2025-04-08T15:53:03.971777+00:00 |

## Intro to components ​

Web components allow you to create reusable custom elements — with their functionality encapsulated away from the rest of your code — and utilize them in your web apps.  
They are especially helpful to:  
build faster, design an interface that has the same look-and-feel across pages, and develop complex projects without repeating yourself.  
In summary:  
The core idea of components is simple but powerful: create something once and reuse it in many different ways while keeping its functionality and data self-contained. This approach lets you build modular, maintainable applications without duplicating work or losing control of how your components behave and manage their own data.

## Working with components ​

You can find components in the Libraries section of the Add panel:  
  
Sections and elements without an icon refer to the section and element Templates available in your Libraries .

### Images

<https://docs.weweb.io/assets/components-add.Dat5GQ5u.png>

## Templates vs components ​

When you drag-and-drop a template on a page, it will create a copy of the template. If you later make a change to the template, those changes will not be reflected in the copies you created before.  
When you drag-and-drop a component on a page, it will create a new instance of the component. If you make a change to the component, those changes will be reflected in all the instances of the component you created before.

### Images

<https://docs.weweb.io/assets/components-none.uxt8fE5i.png>

## Create a new component ​

In WeWeb, there are two ways you can create a new component.

## Option 1 ​

You can select an element in the HTML tree on the left of the editor and click on New component in the right panel:

### Images

<https://docs.weweb.io/assets/components-create1.Ce8JD6dI.png>

## Option 2 ​

You can select an element on the canvas and click on the component icon to transform it into a new component:

### Images

<https://docs.weweb.io/assets/components-create2.D7PuVWHP.png>

## Rename or delete components ​

When you create a new component in a project, it will be added to that project's library.  
To view, rename, or delete a project component, you can navigate to Libraries :

### Images

<https://docs.weweb.io/assets/components-library.Fr0-B8eb.png>

<https://docs.weweb.io/assets/components-shared-library.BXy4gs1E.gif>

## Edit a component ​

Once you have created a component, you will be able to:  
change the style of that instance of the component, or edit the component itself so that changes are reflected on all the instances of that component

## Updating an instance vs updating a component ​

To customize one instance of a component, simply make the changes you want in the right panel as you would with any other element.  
To edit a component (and have those changes reflected on all instances of that component), select the component and click on the Edit button at the top of the right panel:

### Images

<https://docs.weweb.io/assets/components-edit-instance.BSPo5V7D.png>

## Updating a component ​

Once you have selected a component and clicked on Edit , you will be able to update the component itself:  
  
In the screenshot above, you can see a few things:  
the Back to instance button in the top right confirms that you are currently editing the component itself, not an instance of the component the Component tab allows you to create properties, variables, workflows, formulas, and triggers associated with the component the Styles , Settings , and Workflow tabs allow you to customize the component as you would any other element. The changes you make here will be applied to all the instances of that component.

### Images

<https://docs.weweb.io/assets/components-edit.DHQLK4o7.png>

## Component properties ​

When you create a component in WeWeb, you can also associate that component with properties (often referred to as "props").  
These properties can be texts, numbers, booleans, selects, collections, and more:

### Images

<https://docs.weweb.io/assets/components-properties-types.lBYRGUvz.png>

## Benefit of using props ​

When you use a component in multiple places, you can use its props to differentiate the behavior of one instance compared to another.  
In the example below, we have a button components with 2 properties ( style and usage ):  
  
One the page, we have 4 instances of the same component, but each instance is styled differently and display a different text.  
This is because each instance of the component has different usage and style property values.  
For example, the text of instance 3 is Login because its usage property value is For app :  
  
But we could change the text and colors of the button by changing the usage and style values of the instance.  
In the example below, we change the property values of instance 3 to the same values as instance 2:

### Images

<https://docs.weweb.io/assets/components-properties-example.DoHbMdx7.png>

<https://docs.weweb.io/assets/components-instance-properties.BT67oQOa.png>

<https://docs.weweb.io/assets/components-instance-property-values.Bg83EwyZ.gif>

## Create component property ​

In the Components tab, click on New to create a new component property:  
  
You will be invited to:  
name it, select a property type, provide a default value.

### Images

<https://docs.weweb.io/assets/components-properties-new.DyM7CJn8.png>

<https://docs.weweb.io/assets/components-properties-select-type.DOStkaZr.png>

## Advanced options ​

In this section, you can decide if the component property:  
should be displayed in the Styles or Settings tab of the instance, should be bindable, should include states, should include classes.

### Images

<https://docs.weweb.io/assets/components-properties-advanced-options.ClgYYk1H.gif>

## Component variables ​

Component variables are variables that are scoped to the component, i.e. they can only be seen and referred to inside the component as opposed to global variables – in the Data tab of the left panel – which can be seen and accessed throughout the project:  
  
In the example above, there are no global variables used on the page. However, there is a component variable ( selectedItem ) which is used to display data about the selected item on the page.

### Images

<https://docs.weweb.io/assets/components-variable.D8iDpDV5.png>

## Component workflows ​

Component workflows are workflows that are scoped to the component, i.e. they can only be executed inside the component as opposed to global workflows – in the Action tab of the left panel – which can be executed throughout the project:  
  
In the example above, there are no global workflows used on the page. However, there is a component workflow ( Count coins ) which can be executed anywhere inside the component.  
In the example below, you can see:  
our component workflow expects a parameter, we select the down arrow icon, and execute the global workflow on click of that icon, with the parameter down to reflect the fact the user just clicked on the down arrow

### Images

<https://docs.weweb.io/assets/components-workflow.C0GHcU4p.png>

<https://docs.weweb.io/assets/components-workflow-execute.CWxCeizP.gif>

## Component formulas ​

When building a component, you can create formulas that are local to that component. These formula will not exist outside of the component:  
  
In the example above, you can see:  
inside our component, we created a CTA text formula, with two parameters that refer to the two properties of the component: style and usage , the combination of these two parameters will decide what text is displayed in our component, the Current value is "Login" because the property values of the current instance of the component are primary and product

### Images

<https://docs.weweb.io/assets/components-formulas.zowYekff.png>

<https://docs.weweb.io/assets/components-formulas-local.B-sYKu1e.png>

## Component events ​

When working with components, it can be helpful to emit a component event that you can then react to outside the component.  
For example, you could have a login form component that authenticates the user and emits a login success event.  
Then, depending on where you use the login form component, you could react to the login success event to decide if you need to close a modal or change page.  
Learn how to emit and react to component events .