Everything you need to know about data binding in Vue

Link data in a web app creates a link between the app's version & UI. Here's how to use data binding in Vue with interpolation, v-bind, and v-model.

Binding data in a web app creates a link between the app's UI & version . Here's **how to use data binding in Vue with interpolation, v-bind and v-model** .

The app instance manages data, behavior, and elements, and the UI represents the visual aspect that the user interacts with. Linked data allows you to create dynamic web apps.

Here, you'll explore the different associations in Vue, including one- and two-way binding. You will also learn how to implement these associations using **mustache templates** and directives like **v-bind** and **v-model** .

Interpolation in Vue

Interpolation is one of Vue's most popular data binding types. Interpolation allows you to display data values ?? using HTML tags with the mustache template, usually denoted by double curly braces (**{{ }}**).

With the mustache template, you can integrate dynamic app content such as data and modal attributes into HTML. You can achieve this by enclosing data property or method names from the Vue selection object in curly braces.

Here is a version of the Vue app that uses the mustache template for interpolation in Vue:

???? ??????

{{ title }}

??????

{{ text.toUpperCase() }}

???? ???? ?

The above code block uses the mustache template to display the value of the name property in the Vue app's data object. You can also represent JavaScript expression results by interpolation. For example, the expression **{{text.toUpperCase()}}** in the **p** tag displays the uppercase version of the text value, as shown below:

When mounting a Vue app, Vue evaluates the expression in templates and renders the value in the HTML body. Any data property changes update the corresponding value in the UI.

For example:

The above code block changes the title attribute to 'Hello'. This change will automatically be reflected in the UI because the Vue app binds the title attribute to the UI element, as shown below:

One-way data binding using v-bind . command

Like interpolation, one-way data binding links the app's version to the UI. The difference here is that it binds attributes such as data and methods to the HTML attribute.

One-way data binding supports one-way data flow, preventing users from making changes that affect data properties on an instance of the application. This is useful when you want to show data to the user of the app but prevent the user from modifying that data.

You can achieve one-way data binding in Vue with the **v-bind** command or the shorthand (**:**):

The code block shows how to use the v-bind command and its shorthand to bind the value of an input HTML tag to a text data attribute. Here's an example of a Vue app that uses the **v-bind** command to achieve one-way data binding:

```
?? ???? ????

{{ text }}

?? ??
```

An input field and paragraph element showing the value of the **text** attribute . **The value** attribute of the input field is bound to the text attribute using the **v-bind** command .

This block of code creates a one-way binding, where changes in the text property will update the value of the input field, but changes made in the input field will not update the text property in the Vue instance of the **app** .

To show this, we can start with the initial value of the **text** property , ' **Vue is awesome!** '.

After changing the value of the input field to 'Hello World!', note that the Vue app has not been updated. The text in the paragraph tag remains the same:

However, when changing the value of the text property to **Hello World!** in the Vue app version instead of the input field, the input field is updated to reflect the new value:

This way of data binding is useful in cases where you want to show the data to the user but prevent the user from editing it directly. Using v-bind in Vue.js, you can establish one-way bindings, easily connecting your app's data to UI elements.

Two-way data binding using the v-model . command

Two-way data binding allows changes to a UI element's value to be automatically represented in the underlying data model and vice versa. Most front-end JavaScript frameworks share data and handle real-time events in this way.

Vue.js creates a two-way association using the v-model command. The v-model directive creates a two-way association between a form input element and a data property. When entering an input element, the value is automatically updated in the data attribute. Any data attribute changes update the input element.

Here's an example of a Vue app that uses the v-model directive to achieve two-way data binding:

```
??Two-way data binding in Vue ?? ?? ???? ????
```

{{ text }}

```
?? ??
```

The above code block has an input element with the v-model directive that binds it to the text data attribute. The text property is initially set to lf 'Vue is awesome!'.

The input field updates the text property as you type it and reflects the change on the text attribute in the **p** tag . Vue.js uses the **v-model** directive and the input element for two-way data binding.

While **v-bind** allows one-way communication from Vue app to UI, v-model provides two-way communication between Vue app and UI. Because of its ability to allow two-way communication, v-model is often used when working with form elements in Vue.

Here's what you need to know about data binding in Vue. Hope the article is useful to you.