1. Vue uses an HTML-based template syntax that allows you to declaratively bind the rendered DOM to the underlying component instance's data.
2. If you are familiar with Virtual DOM concepts and prefer the raw power of JavaScript, you can also directly write render functions instead of templates, with optional JSX support. However, do note that they do not enjoy the same level of compile-time optimizations as templates.

Text Interpolation

template

```
<p>Using text interpolation: {{ rawHtml }}</p>
<p>Using v-html directive: <span v-html="rawHtml"></span></p>
```

Using text interpolation: <span style="color: red">This should be red.</span>

Using v-html directive: This should be red.

Here we're encountering something new. The #v-html attribute you're seeing is called a **Directive**.

> ⚠ Security Warning
>
> Dynamically rendering arbitrary HTML on your website can be very dangerous because it can easily lead to **XSS vulnerabilities**. Only use `v-html` on trusted content and **never** on user-provided content.

Attribute Bindings

```template
<div v-bind:id="dynamicId"></div>
```

The `v-bind` directive instructs Vue to keep the element's `id` attribute in sync with the component's `dynamicId` property. If the bound value is `null` or `undefined`, then the attribute will be removed from the rendered element.

#v-bind -> can be just " : "
`<div :id="dynamicId"></div>`

Dynamically Binding Multiple Attributes

#Dynamically_Binding_Multiple_Attributes
If you have a JavaScript object representing multiple attributes that looks like this:

```js
data() {
  return {
    objectOfAttrs: {
      id: 'container',
      class: 'wrapper'
    }
  }
}
```

You can bind them to a single element by using `v-bind` without an argument:

```template
<div v-bind="objectOfAttrs"></div>
```