



Vue 3 Binding styles

Binding styles

Color is single property

Font size is two words hence written in double codes or else in camel case, styles can also be applied directly static way. Here color, font size are style properties.

Heading

Underlined text

Status

Promoted Movie

Soldout? movie

Newly promoted movie

Array conditional movie

Object conditional movie



Inline Style

```
<h2 v-bind:style="[]<br/><span>  color: highlightColor,<br/>  fontSize: headerSize + 'px',<br/>  padding: '20px'<br/>]>Inline Style</h2>
```

```
24 <script>
25 export default {
26   name: 'App',
27   data() {
28     return {
29       headingId: 'heading',
30       isDisabled: false,
31       status: 'success',
32       isPromoted: true,
33       isSoldout: true,
34       highlightColor: 'orange',
35       headerSize: 50
36     }
37   },
38 }
39 </script>
```

Object type style binding

```
<h2 v-bind:style="headerStyleObject">Style Object</h2>
</template>
```

Array type style binding in array binding last object styling overides first one

```
<h2 v-bind:style="headerStyleObject">Style Object</h2>
<div v-bind:style="[baseStyleObject, successStyleObject]">Success Style</div>
<div v-bind:style="[baseStyleObject, dangerStyleObject]">Danger Style</div>
</template>
```

```
padding: 20px,
},
dangerStyleObject: {
  color: 'darkred',
  backgroundColor: 'red',
  border: '1px solid darkred',
},
[]
```

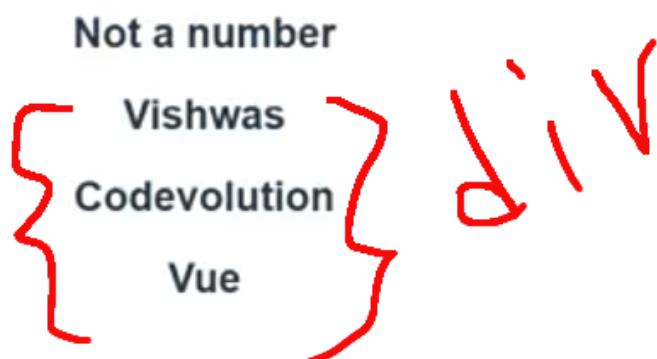


Vue condition rendering 1

V-else always should be followed by v-if /v if-else

If we want to render two parameters then add v-if inside div tag (considered as template not as div)

Instead of div we should bind inside template.



```

<div v-if="display">
  <h2>Vishwas</h2>
  <h2>Codevolution</h2>
  <h2>Vue</h2>
</div>
</template>

```

```

<script>
display: true
</script>
Right way: <template v-if="display">

```

List Rendering 1

Key value and printing value should be same

Keyvalue = iterator value

```

export default {
  name: 'App',
  data() {
    return {
      names: ['Bruce', 'Clark', 'Diana'],
      fullNames: [
        { first: 'Bruce', last: 'Wayne' },
        { first: 'Clark', last: 'Kent' },
        { first: 'Princess', last: 'Diana' },
      ],
    }
  }
}

```

If we want to use a separator between elements then

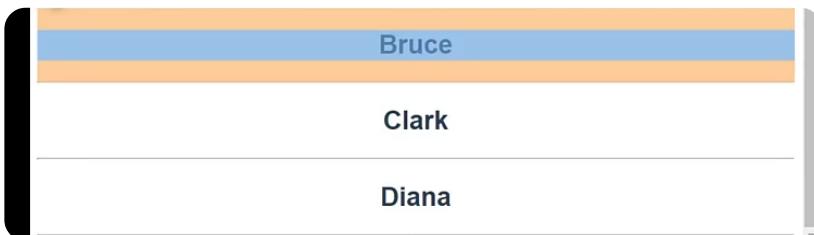
```
2 <template v-for="name in names" :key="name">
3   <h2>{{name}}</h2>
4   <hr />
5 </template>
```



Vue condition rendering 2

If we want to use a separator between elements then

The loop works in such away that he is rendered 3 times even we are declaring it only 1 time



```
2   <template v-for="name in names" :key="name">
3     <h2>{{name}}</h2>
4     <hr />
5   </template>
```

```
export default {
  name: 'App',
  data() {
    return {
      names: ['Bruce', 'Clark', 'Diana'],
      fullNames: [
        { first: 'Bruce', last: 'Wayne' },
        { first: 'Clark', last: 'Kent' },
        { first: 'Princess', last: 'Diana' },
      ],
      actors: [
        {
          name: 'Christian Bale',
          movies: ['Batman', 'The Prestige'],
        },
        {
          name: 'Di Caprio',
          movies: ['Titanic', 'Inception'],
        },
      ],
      myInfo: []
    }
  }
}
```

If we want to display particular element in a loop

Then the ideal way is as follows only bruce is displayed

```
App.vue > t / App.vue > template > template > h2
<template>
  <template v-for="name in names" :key="name">
    <h2 v-if="name === 'Bruce'">{{ name }}</h2>
  </template>
</template>

<script>
export default {
  name: 'App',
  data() {
    return {
      names: ['Bruce', 'Clark', 'Diana'],
    }
  },
}</script>
```

Use of key attribute in Vue

```
File Edit Selection View Go Help Anonymous / damp-de
▼ App.vue x
1  <template>
2    <template v-for="name in names" :key="name">
3      <h2>{{ name }}</h2>
4      <input placeholder="Last name" />
5      <hr />
6    </template>
7    <button @click="shuffle">Shuffle!</button>
8  </template>
9
```

Bruce

Clark

Diana

Barry

If we use key then ok even after shuffling particular input sticked with particular element



Vue event handling

```
<template>
  <h2>{{ 2 + 3 + 5 }}</h2>
  <h2>{{ 5 + 10 + 15 }}</h2>
  <h2>Add method - {{ add() }}</h2>
</template>

<script>
export default {
  name: 'App',
  data() {
    return {}
  },
  methods: {
    add() {
      return 2 + 3 + 5
    },
  },
}
</script>
```

10

30

Add method - 10**Add method - 45**

```
<template>
  <h2>{{ 2 + 3 + 5 }}</h2>
  <h2>{{ 5 + 10 + 15 }}</h2>
  <h2>Add method - {{ add(2,3,5) }}</h2>
  <h2>Add method - {{ add(10,15,20) }}</h2>
</template>

<script>
export default {
  name: 'App',
  data() {
    return {}
  },
  methods: {
    add(a, b, c) {
      return a + b + c
    },
  },
}
</script>
```

```
1 <template>
2   <h2>{{name}}</h2>
3   <div>
4     <button v-on:click="name = 'Batman'">Change name</button>
5   </div>
6 </template>
7
8 <script>
9 export default {
10   name: 'App',
11   data() {
12     return {
13       name: 'Vishwas'
14     }
15   },
16   methods: {
17   }
18 }
19 </script>
20
```

Batman

Change name



On click name changes from vishwas to batman



Event handling 2

Batman

Change name



```
<div>
  <button @click="increment(1, $event)">Increment 1</button>
```

```
  ,
  methods: {
    changeName(event) {
      this.name = 'Batman'
      console.log('Event', event)
    },
    increment(num, event) {
      this.count += num
      console.log(['Event', event])
    },
  },
```

```
<template>
  <h2>{{name}}</h2>
  <div>
    <button v-on:click="changeName">Change name</button>
  </div>
```



Form handling 1

Created formvalues and gave name, defined that in template and binded with V-Model formvalues.name

```
1 <template>
2   <div>
3     <pre>
4       {{ JSON.stringify(formValues, null, 2)}}
5     </pre>
6   </div>
7   <form>
8     <div>
9       <label for="name">Name</label>
10      <input type="text" id="name" v-model="formValues.name" />
11    </div>
12  </form>
13 </template>
14
15 <script>
16 export default {
17   name: 'App',
18   data() {
19     return {
20       formValues: {
21         name: ''
22       }
23     }
24   }
25 }
```

```
{
  "name": "Vishwas"
}
```

Name

To send the data from template to script (2 way binding) we use V-Model

```

<script>
export default {
  name: 'App',
  data() {
    return {
      formValues: {
        name: '',
        profileSummary: '',
        country: '',
        jobLocation: []
      },
    }
  },
  methods: {},
}
</script>

```

`{
 "name": "",
 "profileSummary": "",
 "country": "",
 "joblocation": []
}`

Name

Profile Summary

Country

Select a country

Job Location

- India
- Vietnam
- Singapore

```

<div>
  <label for="name">Name</label>
  <input type="text" id="name" v-model="formValues.name" />
</div>

<div>
  <label for="profile">Profile Summary</label>
  <textarea id="profile" v-model="formValues.profileSummary" />
</div>

<div>
  <label for="country">Country</label>
  <select id="country" v-model="formValues.country">
    <option value="">Select a country</option>
    <option value="india">India</option>
    <option value="vietnam">Vietnam</option>
    <option value="singapore">Singapore</option>
  </select>
</div>

</form>
</template>

```

```

<div>
  <label for="job-location">Job Location</label>
  <select id="job-location" multiple v-model="formValues.jobLocation">
    <option value="">Select a country</option>
    <option value="india">India</option>
    <option value="vietnam">Vietnam</option>
    <option value="singapore">Singapore</option>
  </select>
</div>
</form>

```




Form handling 2

Skill set html in template

```
<div>
  <label>Skill set</label>
  <input type="checkbox" id="html" value="html" v-model="formValues.skillSet" />
  <label for="html">HTML</label>
  <input type="checkbox" id="css" value="css" v-model="formValues.skillSet" />
  <label for="css">CSS</label>
  <input type="checkbox" id="javascript" value="javascript" v-model="formValues.skillSet" />
  <label for="javascript">JavaScript</label>
</div>
```

Submit form event passing , by default page gets reloaded to avoid this we give prevent default.

```
    </pre>
  </div>
  <form @submit="submitForm">
    <div>
```

```
23   yearsOfExperience: '',
24 },
25 }
26 },
27 methods: {
28   submitForm(event) {
29     event.preventDefault()
30     console.log(['Form values', this.formValues])
31   }
32 },
33 }
34 </script>
```

This is for radio button and binding using vmodel

```

<div>
  <label>Years of Experience</label>
  <input
    type="radio"
    id="0-2"
    value="0-2"
    v-model="formValues.yearsOfExperience">
  />
  <label for="0-2">0-2</label>
  <input
    type="radio"
    id="3-5"
    value="3-5"
    v-model="formValues.yearsOfExperience">
  />
  <label for="3-5">3-5</label>
  <input
    type="radio"
    id="6-10"
    value="6-10"
    v-model="formValues.yearsOfExperience">
  />
  <label for="6-10">5-10</label>
  <input

```

```

    {
      "name": "",
      "profileSummary": "",
      "country": "",
      "jobLocation": [],
      "remoteWork": "no",
      "skillSet": [],
      "yearsOfExperience": "0-2"
    }

Name
  

Profile Summary
  

Country
<select>
  <option>Select a country</option>
  <option>India</option>
  <option>Vietnam</option>
  <option>Singapore</option>
</select>  

Job Location
  

 Open to remote work?  

Skill set
 HTML    CSS    JavaScript  

Years of Experience
 0-2    3-5    5-10    10+

```

form values attribute inside script

```

<script>
export default {
  name: 'App',
  data() {
    return {
      formValues: {
        name: '',
        profileSummary: '',
        country: '',
        jobLocation: []
      },
    }
  },
  methods: {},
}
</script>

```



Computed properties

On click of button keyboard price also added to computed total (total is computed property mentioned) and it returns curr.price has each price of item and goes on adds up and calls.

```

        price: 500,
      },
      ],
    },
  },
  methods: {},
  computed: {
    fullName() {
      return `${this.firstName} ${this.lastName}`
    },
    total() {
      return this.items.reduce([(total, curr) => (total = total + curr.price), 0])
    }
  }
}
</script>
<style>
```

Fullname - Bruce Wayne

Computed fullname - Bruce Wayne

Add item

Computed Total - 650



```

<template>
  <h2>Fullname - {{ firstName }} {{ lastName }}</h2>
  <h2>Computed fullname - {{ fullName }}</h2>
  <button @click="items.push({id: 4, title: 'Keyboard', price: 50})">Add item</button>
  <h2>Computed Total - {{ total }}</h2>
</template>

<script>
export default {
  name: 'App',
  data() {
    return {
      firstName: 'Bruce',
      lastName: 'Wayne',
      items: [
        {
          id: 1,
          title: 'TV',
          price: 100,
        },
        {
          id: 2,
          title: 'Phone',
          price: 200,
        }
      ]
    }
  }
}
</script>
```




Props

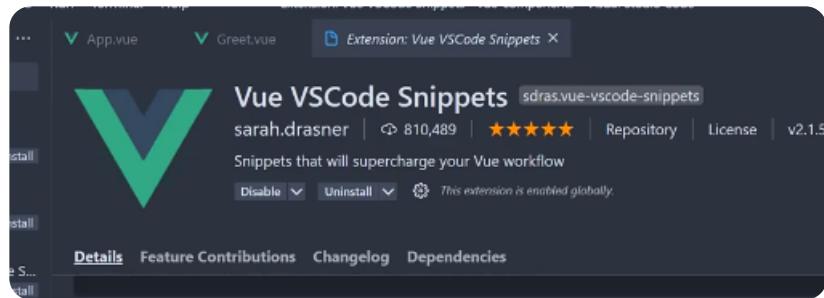
Greet.vue

```
c > components > Greet.vue > Greet.vue > Greet.vue
1   <template>
2 |   <h2>Hello {{ name }}</h2>
3   </template>
4
5   <script>
6   export default {
7     name: "Greet",
8     props: ['name']
9   };
10  </script>
11
12  <style scoped>
13  </style>
```

```
<template>
  <Greet name="Bruce" />
  <Greet name="Clark" />
  <Greet name="Diana" />
</template>

<script>
import Greet from './components/Greet.vue'

export default {
  name: 'App',
  components: {
    Greet,
  }
}
</script>
```





Provide and inject

Case 1

Step 1

```
App.vue ● ComponentC.vue ComponentE.vue ComponentF.vue
src > App.vue > {} "App.vue" > script > default > provide > username
1 <template>
2   <ComponentC />
3 </template>
4
5 <script>
6   import ComponentC from './components/ComponentC.vue'
7
8   export default {
9     name: 'App',
10    components: {
11      ComponentC,
12    },
13    data() {
14      return {
15        }
16    },
17    provide: [
18      username: 'Vishwas'
19    ]
20  }
21 </script>
22
23 <style>
24 #app {
```

Step 2

```
> components > ComponentF.vue > {} "ComponentF.vue" > style
1 <template>
2   <h2>Component F</h2>
3   <h3>Component F username {{ username }}</h3>
4 </template>
5
6 <script>
7   export default {
8     name: 'ComponentF',
9     inject: ['username'],
10   }
11 </script>
12
13 <style scoped>
14 </style>
```

Output

Component C

Component E

Component F

Component F username Vishwas

Case 2

If we want to display username into template but username is not displaying in this way

```
src > App.vue > {} "App.vue" > template > h3
1 | <template>
2 |   | <h3>AppComponent username {{ username }}</h3>
3 |   | <ComponentC />
4 |   </template>
5 |
6 <script>
7 import ComponentC from './components/ComponentC.vue'
8
9 export default {
10   name: 'App',
11   components: {
12     ComponentC,
13   },
14   data() {
15     return {}
16   },
17   provide: {
18     username: 'Vishwas',
19   },
20 }
21 </script>
```

AppComponent username

Component C

Component E

Component F

Component F username Vishwas

