Guía de Estilo

> Próximamente, también proporcionaremos consejos para su ejecución. A veces simplemente tendrá que ser disciplinado, pero siempre que sea posible, trataremos de mostrarle cómo usar ESLint y otros procesos automatizados para simplificar el cumplimiento.

Categorías [#Categorias]

Prioridad A: Esencial [#Prioridad-A-Esencial]

Prioridad B: Muy recomendable [#Prioridad-B-Muy-recomendable]

Prioridad C: Recomendado [#Prioridad-C-Recomendado]

Prioridad D: Usar con precaución [#Prioridad-D-Usar-con-precaucion]

Reglas de prioridad A: Esencial (prevención de errores) [#Reglas-de-prioridad-A-Esencial-prevencion-de-errores]

Nombres de componentes de varias palabras [#Nombres-de-componentes-de-varias-palabras-esencial]

ESENCIAL

Los nombres de los componentes siempre deben ser de varias palabras, a excepción de los componentes raíz de la “App”

evita conflictos [http://w3c.github.io/webcomponents/spec/custom/#valid-custom-element-name]

[#Incorrecto] Incorrecto

```JS
Vue.component('todo', {
  // ...
})
```

```JS
export default {
  name: 'Todo',
  // ...
}
```

[#Correcto] Correcto

```JS
Vue.component('todo-item', {
  // ...
})
```

```JS
export default {
  name: 'TodoItem',
  // ...
}
```

Componente data [#componente-data-esencial] ESENCIAL

El componente `data` debe ser una función.

`data`

`new Vue`

▶ **Explicación Detallada**

[#Incorrecto-1] Incorrecto

```JS
Vue.component('some-comp', {
  data: {
    foo: 'bar'
  }
})
```

```JS
export default {
  data: {
    foo: 'bar'
  }
}
```

[#Correcto-1] Correcto

```JS
Vue.component('some-comp', {
  data: function () {
    return {
      foo: 'bar'
    }
  }
})
```

```JS
// En un archivo .vue
export default {
  data () {
    return {
      foo: 'bar'
    }
  }
}
```

```js
// Está bien usar un objeto directamente
// en una instancia raíz de Vue,
// ya que solo existirá esa única instancia.
new Vue({
  data: {
    foo: 'bar'
  }
})
```

Definiciones de Props [#Definiciones-de-Props-esencial]

ESENCIAL

Las definiciones de Props deben ser lo mas detalladas posible.

▶ **Explicación Detallada**

[#Incorrecto-2] Incorrecto

```js
// Esto esta bien solo cuando se prototipa
props: ['status']
```

[#Correcto-2] Correcto

```js
props: {
  status: String
}
```

```JS
// Mucho mejor!
props: {
  status: {
    type: String,
    required: true,
    validator: function (value) {
      return [
        'syncing',
        'synced',
        'version-conflict',
        'error'
      ].indexOf(value) !== -1
    }
  }
}
```

Key + `v-for` [#Key-v-for-esencial]

Siempre use `key` con `v-for`.

`key` `v-for` *perenne*

constancia del objeto

[https://bost.ocks.org/mike/constancy/]

▶ **Explicación Detallada**

[#Incorrecto-3] Incorrecto

```HTML
<ul>
  <li v-for="todo in todos">
    {{ todo.text }}
  </li>
</ul>
```

[#Correcto-3] Correcto

```HTML
<ul>
  <li
    v-for="todo in todos"
    :key="todo.id"
  >
    {{ todo.text }}
  </li>
</ul>
```

Evitar `v-if` con `v-for` [#ESENCIAL-Evitar-v-if-con-v-for-esencial]

Nunca use `v-if` en el mismo elemento que `v-for` .

- `v-for="user in users" v-if="user.isActive"`
  `users`
  `activeUsers`
- `v-for="user in users" v-if="shouldShowUsers"`
  `v-if` `ul` `ol`

▶ **Explicación Detallada**

[#Incorrecto-4] Incorrecto

```HTML
<ul>
  <li
    v-for="user in users"
    v-if="user.isActive"
    :key="user.id"
  >
    {{ user.name }}
  <li>
</ul>
```

```HTML
<ul>
  <li
    v-for="user in users"
    v-if="shouldShowUsers"
    :key="user.id"
  >
    {{ user.name }}
  <li>
</ul>
```

[#Correcto-4] Correcto

```HTML
<ul>
  <li
    v-for="user in activeUsers"
    :key="user.id"
  >
    {{ user.name }}
  <li>
</ul>
```

```HTML
<ul v-if="shouldShowUsers">
  <li
    v-for="user in users"
    :key="user.id"
  >
    {{ user.name }}
  <li>
</ul>
```

Scoping de Estilos de Componentes [#Scoping-de-Estilos-de-Componentes-esencial] ESENCIAL

Para las aplicaciones, los estilos, en un componente `App` de nivel superior y en los componentes de "layout" pueden ser globales, pero todos los demás componentes siempre deben ser *scoped*

componentes single-file [../guide/single-file-components.html] *No* `scoped` [https://vue-loader.vuejs.org/en/features/scoped-css.html] CSS modules [https://vue-loader.vuejs.org/en/features/css-modules.html] BEM [http://getbem.com/]

Para las bibliotecas de componentes, sin embargo, se debería implementar una estrategia basada en clases en vez de usar el atributo `scoped` .

▶ Explicación Detallada

[#Incorrecto-5] Incorrecto

```HTML
<template>
  <button class="btn btn-close">X</button>
</template>

<style>
.btn-close {
  background-color: red;
}
</style>
```

[#Correcto-5] Correcto

HTML

```
<template>
  <button class="button button-close">X</button>
</template>

<!-- Usando el atributo `scoped` -->
<style scoped>
.button {
  border: none;
  border-radius: 2px;
}

.button-close {
  background-color: red;
}
</style>
```

HTML

```
<template>
  <button :class="[$style.button, $style.buttonClose]">X</button>
</template>

<!-- Usando CSS modules -->
<style module>
.button {
  border: none;
  border-radius: 2px;
}

.buttonClose {
  background-color: red;
}
</style>
```

HTML

```
<template>
  <button class="c-Button c-Button--close">X</button>
</template>

<!-- Usando la convención BEM -->
<style>
.c-Button {
  border: none;
  border-radius: 2px;
}

.c-Button--close {
  background-color: red;
}
</style>
```

Nombres de propiedades privadas [#Nombres-de-propiedades-privadas-esencial]

ESENCIAL

Siempre use el prefijo `$_` para propiedades privadas en un plugin, mixin, etc. Luego, para evitar conflictos con código de otros desarrolladores, también incluya un *scope* (ej. `$_yourPluginName_`).

▶ **Explicación Detallada**

[#Incorrecto-6] Incorrecto

JS

```
var myGreatMixin = {
  // ...
  methods: {
    update: function () {
      // ...
    }
  }
}
```

```JS
var myGreatMixin = {
  // ...
  methods: {
    _update: function () {
      // ...
    }
  }
}
```

```JS
var myGreatMixin = {
  // ...
  methods: {
    $update: function () {
      // ...
    }
  }
}
```

```JS
var myGreatMixin = {
  // ...
  methods: {
    $_update: function () {
      // ...
    }
  }
}
```

[#Correcto-6] Correcto

```JS
var myGreatMixin = {
  // ...
  methods: {
    $_myGreatMixin_update: function () {
      // ...
    }
  }
}
```

Reglas de prioridad B: Altamente Recomendadas (Mejorar la legibilidad) [#Reglas-de-prioridad-B-Altamente-Recomendadas-Mejorar-la-legibilidad]

Cada componente es un archivo [#Cada-componente-es-un-archivo-altamente-recomendado] ALTAMENTE RECOMENDADO

Siempre que un compilador pueda concatenar archivos, cada componente debería estar en su propio archivo.

[#Incorrecto-7] Incorrecto

```js
Vue.component('TodoList', {
  // ...
})

Vue.component('TodoItem', {
  // ...
})
```

[#Correcto-7] Correcto

```
components/
|- TodoList.js
|- TodoItem.js
```

```
components/
|- TodoList.vue
|- TodoItem.vue
```

Notación de nombres de componentes single-file [#Notación-de-nombres-de-componentes-single-file-altamente-recomendado]

ALTAMENTE RECOMENDADO

Los nombres de los archivos de los componentes single-file [../guide/single-file-components.html] deben ser siempre PascalCase o siempre kebab-case.

[#Incorrecto-8] Incorrecto

```
components/
|- mycomponent.vue
```

```
components/
|- myComponent.vue
```

[#Correcto-8] Correcto

```
components/
|- MyComponent.vue
```

```
components/
|- my-component.vue
```

Nombre de componentes base [#Nombre-de-componentes-base-altamente-recomendado]

ALTAMENTE RECOMENDADO

Los componentes base (también conocidos como componentes de presentación, “tontos”, o puros) que aplican estilos y convenciones específicas de la aplicación,

deben comenzar con un prefijo específico, tal como `Base`, `App` o `V`.

▶ **Explicación Detallada**

[#Incorrecto-9] Incorrecto

```
components/
|- MyButton.vue
|- VueTable.vue
|- Icon.vue
```

[#Correcto-9] Correcto

```
components/
|- BaseButton.vue
|- BaseTable.vue
|- BaseIcon.vue
```

```
components/
|- AppButton.vue
|- AppTable.vue
|- AppIcon.vue
```

```
components/
|- VButton.vue
|- VTable.vue
|- VIcon.vue
```

Nombres de componentes de instancia única [#Nombres-de-componentes-de-instancia-unica-altamente-recomendado] ALTAMENTE RECOMENDADO

Componentes que deben tener solamente una única instancia activa deben comenzar con el prefijo `The`, para denotar que solo puede haber una.

por página

por ahora

[#Incorrecto-10] Incorrecto

```
components/
|- Heading.vue
|- MySidebar.vue
```

[#Correcto-10] Correcto

```
components/
|- TheHeading.vue
|- TheSidebar.vue
```

Nombres de componentes fuertemente acoplados [#Nombres-de-componentes-fuertemente-acoplados-altamente-recomendado]

ALTAMENTE RECOMENDADO

Los componentes hijo que están fuertemente acoplados a su padre deben incluir el nombre del componente padre como prefijo.

▶ **Explicación Detallada**

[#Incorrecto-11] Incorrecto

```
components/
|- TodoList.vue
|- TodoItem.vue
|- TodoButton.vue
```

```
components/
|- SearchSidebar.vue
|- NavigationForSearchSidebar.vue
```

[#Correcto-11] Correcto

```
components/
|- TodoList.vue
|- TodoListItem.vue
|- TodoListItemButton.vue
```

```
components/
|- SearchSidebar.vue
|- SearchSidebarNavigation.vue
```

Orden de las palabras en el nombre de los componentes [#Orden-de-las-palabras-en-el-nombre-de-los-componentes-altamente-recomendado]

ALTAMENTE RECOMENDADO

Los nombres de los componentes deben comenzar con la palabra de más alto nivel (muchas veces la más general) y terminar con palabras descriptivas.

▶ **Explicación Detallada**

[#Incorrecto-12] Incorrecto

```
components/
|- ClearSearchButton.vue
|- ExcludeFromSearchInput.vue
|- LaunchOnStartupCheckbox.vue
|- RunSearchButton.vue
|- SearchInput.vue
|- TermsCheckbox.vue
```

[#Correcto-12] Correcto

```
components/
|- SearchButtonClear.vue
|- SearchButtonRun.vue
|- SearchInputQuery.vue
|- SearchInputExcludeGlob.vue
|- SettingsCheckboxTerms.vue
|- SettingsCheckboxLaunchOnStartup.vue
```

Componentes con cierre automático [#Componentes-con-cierre-automatico-altamente-recomendado] ALTAMENTE RECOMENDADO

Componentes sin contenido deben cerrarse automáticamenete en componentes single-file [../guide/single-file-components.html] , plantillas basadas en *strings*, y JSX [../guide/render-function.html#JSX] - pero nunca en plantillas del DOM.

tag

official “void” elements [https://www.w3.org/TR/html/syntax.html#void-elements]

[#Incorrecto-13] Incorrecto

```HTML
<!-- En componentes de un solo archivo, templates basados en string,
<MyComponent></MyComponent>
```

```HTML
<!-- En plantillas del DOM -->
<my-component/>
```

[#Correcto-13] Correcto

```HTML
<!-- En componentes de un solo archivo, templates basados en string,
<MyComponent/>
```

```HTML
<!-- En plantillas del DOM -->
<my-component></my-component>
```

Notación de nombres de componentes en templates [#Notación-de-nombres-de-componentes-en-templates-altamente-recomendado] ALTAMENTE RECOMENDADO

En la mayoría de los proyectos, los nombres de los componentes deben ser siempre PascalCase en componentes single-file [../guide/single-file-components.html] y plantillas basadas en *string* - pero kebab-case en plantillas del DOM.

-
- `<MyComponent>`

  `<my-component>`
-

utilizar kebab-case en todos lados también es aceptable

[#Incorrecto-14] Incorrecto

```HTML
<!-- En componentes single-file y templates basados en string -->
<mycomponent/>
```

```HTML
<!-- En componentes single-file y templates basados en string -->
<myComponent/>
```

```HTML
<!-- En DOM templates -->
<MyComponent></MyComponent>
```

[#Correcto-14] Correcto

```HTML
<!-- En componentes single-file y templates basados en string -->
<MyComponent/>
```

```HTML
<!-- En DOM templates -->
<my-component></my-component>
```

```HTML
<!-- En todos lados -->
<my-component></my-component>
```

Notación de nombres de componentes en JS/JSX [#ALTAMENTE RECOMENDADO Notación-de-nombres-de-componentes-en-JS-JSX-altamente-recomendado]

Los nombres de los componentes en JS/JSX [../guide/render-function.html#JSX] siempre deben ser PascalCase, aunque pueden ser kebab-case dentro de *strings* en aplicaciones más simples, que solo utilizan registro global de componentes a través de `Vue.component` .

▶ **Explicación Detallada**

[#Incorrecto-15] Incorrecto

```
JS
Vue.component('myComponent', {
  // ...
})
```

```
JS
import myComponent from './MyComponent.vue'
```

```
JS
export default {
  name: 'myComponent',
  // ...
}
```

```
JS
export default {
  name: 'my-component',
  // ...
}
```

[#Correcto-15] Correcto

```js
Vue.component('MyComponent', {
  // ...
})
```

```js
Vue.component('my-component', {
  // ...
})
```

```js
import MyComponent from './MyComponent.vue'
```

```js
export default {
  name: 'MyComponent',
  // ...
}
```

Componentes con nombres completos [#Componentes-con-nombres-completos-altamente-recomendado] ALTAMENTE RECOMENDADO

Nombres de componentes deben tener palabras completas, en vez de abreviaciones.

[#Incorrecto-16] Incorrecto

```
components/
|- SdSettings.vue
|- UProfOpts.vue
```

[#Correcto-16] Correcto

```
components/
|- StudentDashboardSettings.vue
|- UserProfileOptions.vue
```

Notación de nombres de propiedades [#Notación-de-nombres-de-propiedades-altamente-recomendado] ALTAMENTE RECOMENDADO

Los nombres de propiedades siempre deben utilizar camelCase al declararse, pero kebab-case en plantillas y JSX [../guide/render-function.html#JSX] .

[#Incorrecto-17] Incorrecto

JS
```
props: {
  'greeting-text': String
}
```

HTML
```
<WelcomeMessage greetingText="hi"/>
```

[#Correcto-17] Correcto

JS
```
props: {
  greetingText: String
}
```

HTML
```
<WelcomeMessage greeting-text="hi"/>
```

Elementos multi-atributo [#Elementos-multi-atributo-altamente-recomendado] ALTAMENTE RECOMENDADO

Elementos con múltiples atributos deben ocupar múltiples líneas, con un atributo por línea.

JSX [../guide/render-function.html#JSX]

[#Incorrecto-18] Incorrecto

```HTML
<img src="https://vuejs.org/images/logo.png" alt="Vue Logo">
```

```HTML
<MyComponent foo="a" bar="b" baz="c"/>
```

[#Correcto-18] Correcto

```HTML
<img
  src="https://vuejs.org/images/logo.png"
  alt="Vue Logo"
>
```

```HTML
<MyComponent
  foo="a"
  bar="b"
  baz="c"
/>
```

Expresiones simples en templates [#Expresiones-simples-en-templates-altamente-recomendado] ALTAMENTE RECOMENDADO

Plantillas de componentes deben incluir expresiones simples, con expresiones más complejas refactorizadas en propiedades computadas o métodos.

qué *cómo*

[#Incorrecto-19] Incorrecto

```HTML
{{
  fullName.split(' ').map(function (word) {
    return word[0].toUpperCase() + word.slice(1)
  }).join(' ')
}}
```

[#Correcto-19] Correcto

```HTML
<!-- En un template -->
{{ normalizedFullName }}
```

```JS
// La expresión compleja ha sido movida a una propiedad computada
computed: {
  normalizedFullName: function () {
    return this.fullName.split(' ').map(function (word) {
      return word[0].toUpperCase() + word.slice(1)
    }).join(' ')
  }
}
```

Propiedades computadas simples [#Propiedades-computadas-simples-altamente-recomendado] #ALTAMENTE RECOMENDADO

Propiedades computadas complejas deben ser separadar en propiedades más simples, siempre que sea posible.

▶ **Explicación Detallada**

[#Incorrecto-20] Incorrecto

```js
computed: {
  price: function () {
    var basePrice = this.manufactureCost / (1 - this.profitMargin)
    return (
      basePrice -
      basePrice * (this.discountPercent || 0)
    )
  }
}
```

[#Correcto-20] Correcto

```js
computed: {
  basePrice: function () {
    return this.manufactureCost / (1 - this.profitMargin)
  },
  discount: function () {
    return this.basePrice * (this.discountPercent || 0)
  },
  finalPrice: function () {
    return this.basePrice - this.discount
  }
}
```

Comillas en los valores de los atributos [#Comillas-en-los-valores-de-los-atributos-altamente-recomendado] ALTAMENTE RECOMENDADO

Los valores de atributos HTML no vacíos siempre deben estar dentro de comillas (simples o dobles, la que no sea utilizada en JS).

evitar

[#Incorrecto-21] Incorrecto

```HTML
<input type=text>
```

```HTML
<AppSidebar :style={width:sidebarWidth+'px'}>
```

[#Correcto-21] Correcto

```HTML
<input type="text">
```

```HTML
<AppSidebar :style="{ width: sidebarWidth + 'px' }">
```

Abreviación de directivas [#Abreviación-de-directivas-altamente-recomendado] ALTAMENTE RECOMENDADO

Las abreviación de directivas (`:` para `v-bind` y `@` para `v-on:`) deben ser utilizadas siempre o nunca.

[#Incorrecto-22] Incorrecto

```HTML
<input
  v-bind:value="newTodoText"
  :placeholder="newTodoInstructions"
>
```

```HTML
<input
  v-on:input="onInput"
  @focus="onFocus"
>
```

[#Correcto-22] Correcto

```HTML
<input
  :value="newTodoText"
  :placeholder="newTodoInstructions"
>
```

```HTML
<input
  v-bind:value="newTodoText"
  v-bind:placeholder="newTodoInstructions"
>
```

```HTML
<input
  @input="onInput"
  @focus="onFocus"
>
```

```HTML
<input
  v-on:input="onInput"
  v-on:focus="onFocus"
>
```

Reglas de prioridad C: Recomendadas (Minimizando Elecciones Arbitrarias y Sobrecarga Cognitiva) [#Reglas-de-prioridad-C-Recomendadas-Minimizando-Elecciones-Arbitrarias-y-Sobrecarga-Cognitiva]

Orden de las opciones en un componente/instancia [#Orden-de-las-opciones-en-un-componente-instancia-recomendado] RECOMENDADO

Opciones de componentes/instancias deben ser ordenadas consistentemente.