Composing Components

Damian Dulisz

GitHub: @shentao

Twitter: @damiandulisz

Vue.js Core Team

news.vuejs.org

Official Vue.js Newsletter and News Portal

Composing Components?

Vue-multiselect

(2.1.3)

The most complete selecting solution for Vue.js

Stars 4k | license MIT | downloads 136k/m | dependencies none

Pick badges

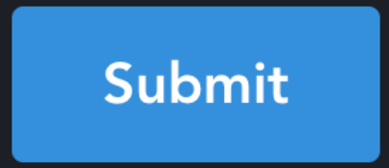
- Single / multiple select
- Dropdowns
- Searchable
- Tagging
- Server-side Rendering support
- Vuex support by default
- Ajax support
- Fully configurable
- +99% test coverage
- No dependencies

View on GitHub

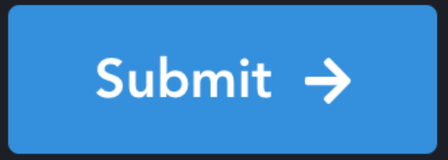
Getting started & examples

Experiment!

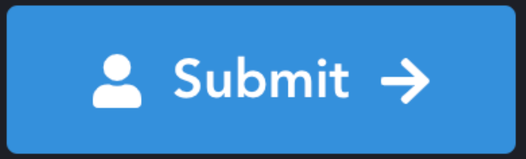
1. *Create a button component that can display text specified in the parent component*



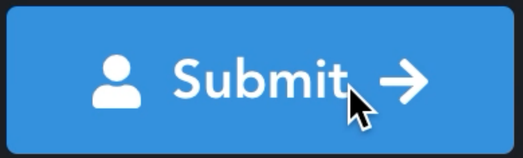
2. *Allow the button to display an icon of choice on the right side of the text*



3. *Make it possible to have icons on either side or even both sides*



4. *Make it possible to replace the content with a loading spinner*



5. *Make it possible to replace an icon with a loading spinner*

Done!

Popular solution

```
<template>
  <button type=“button" class=“nice-button“>
    {{ text }}
  </button>
</template>

<script>
export default {
  props: ['text']
}
</script>
```

```
<template>
  <button type=“button" class=“nice-button“>
    <PulseLoader v-if="isLoading" color="#fff" size="12px">
    <template v-else>
      <template v-if="iconLeftName">
        <PulseLoader v-if="isLoadingLeft" color="#fff" size="6px">
        <AppIcon v-else :icon=“iconLeftName"/>
      </template>
      {{ text }}
      <template v-if="iconRightName">
        <PulseLoader v-if="isLoadingRight" color="#fff" size="6px">
        <AppIcon v-else :icon=“iconRightName”/>
      </template>
    </template>
  </button>
</template>

<script>
export default {
  props: ['text', 'iconLeftName', 'iconRightName', ‘isLoading’,
'isLoadingLeft', 'isLoadingRight']
}
</script>
```


WELL
THAT ESCALATED QUICKLY
quickmeme.com

```
<template>
  <button type=“button" class=“nice-button“>
    <PulseLoader v-if="isLoading" color="#fff" size="12px">
    <template v-else>
      <template v-if="iconLeftName">
        <PulseLoader v-if="isLoadingLeft" color="#fff" size="6px">
        <AppIcon v-else :icon=“iconLeftName"/>
      </template>
      {{ text }}
      <template v-if="iconRightName">
        <PulseLoader v-if="isLoadingRight" color="#fff" size="6px">
        <AppIcon v-else :icon=“iconRightName"/>
      </template>
    </template>
  </button>
</template>

<script>
export default {
  props: ['text', 'iconLeftName', 'iconRightName', ‘isLoading’,
'isLoadingLeft', 'isLoadingRight']
}
</script>
```

Let’s call it the props-based solution

props-based solution

Is it wrong?

props-based solution

Is it wrong?

No.

It does the job.

props-based solution

Is it good, then?

props-based solution

Is it good, then?

Not exactly.

props-based solution

Problems

props-based solution

Problems

- New requirements increase complexity
- Multiple responsibilities
- Lots of conditionals in the template
- Low flexibility
- Hard to maintain

What about this solution?

What about this solution?

```
<template>
  <button type=“button" class=“nice-button“>
    <slot/>
  </button>
</template>
```

What about this solution?

```
<template>
  <button type=“button" class=“nice-button“>
    <slot/>
  </button>
</template>
```

Covers all 5 requirements

```
<template>
  <button type=“button" class=“nice-button“>
    <slot/>
  </button>
</template>
```

Usage:

```
<AppButton>
  Submit
  <PulseLoader v-if="isLoading" color="#fff" size="6px"/>
  <AppIcon v-else icon=“arrow-right"/>
</AppButton>
```

Live coding time!

Other use cases

“Provider” components

“Provider” components

```
<ApolloQuery
  :query="require('../graphql/HelloWorld.gql')"
  :variables="{ name }"
>
  <template slot-scope="{ result: { loading, error, data } }">
    <!-- Loading -->
    <div v-if="loading" class="loading apollo">Loading...</div>
    <!-- Error -->
    <div v-else-if="error" class="error apollo">An error occured</div>
    <!-- Result -->
    <div v-else-if="data" class="result apollo">{{ data.hello }}</div>
```

From Vue Apollo by @akryum

“Provider” components

```
<ApolloQuery
  :query="require('../graphql/HelloWorld.gql')"
  :variables="{ name }"
>
  <template slot-scope="{ result: { loading, error, data } }">
    <!-- Loading -->
    <div v-if="loading" class="loading apollo">Loading...</div>
    <!-- Error -->
    <div v-else-if="error" class="error apollo">An error occured</div>
    <!-- Result -->
    <div v-else-if="data" class="result apollo">{{ data.hello }}</div>
```

From Vue Apollo by @akryum

“Provider” components

```
<ApolloQuery
  :query="require('../graphql/HelloWorld.gql')"
  :variables="{ name }"
>
  <template slot-scope="{ result: { loading, error, data } }">
    <!-- Loading -->
    <div v-if="loading" class="loading apollo">Loading...</div>
    <!-- Error -->
    <div v-else-if="error" class="error apollo">An error occured</div>
    <!-- Result -->
    <div v-else-if="data" class="result apollo">{{ data.hello }}</div>
```

From Vue Apollo by @akryum

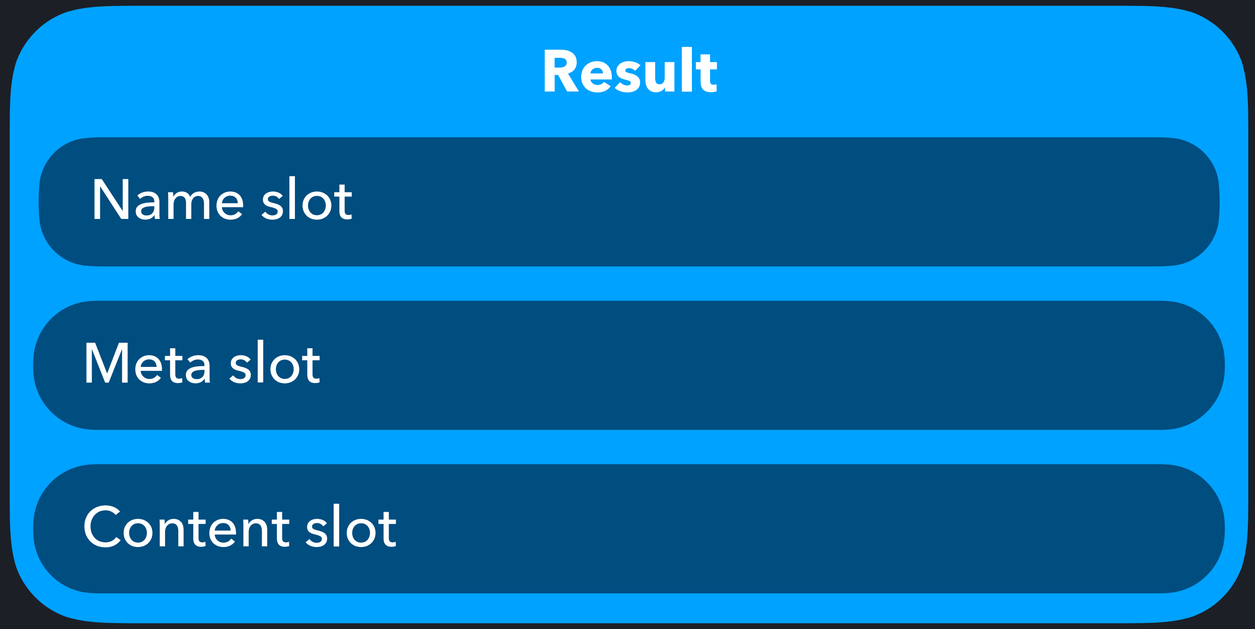
“Provider” components

```
<ApolloQuery
  :query="require('../graphql/HelloWorld.gql')"
  :variables="{ name }"
>
  <template slot-scope="{ result: { loading, error, data } }">
    <!-- Loading -->
    <div v-if="loading" class="loading apollo">Loading...</div>
    <!-- Error -->
    <div v-else-if="error" class="error apollo">An error occured</div>
    <!-- Result -->
    <div v-else-if="data" class="result apollo">{{ data.hello }}</div>
```

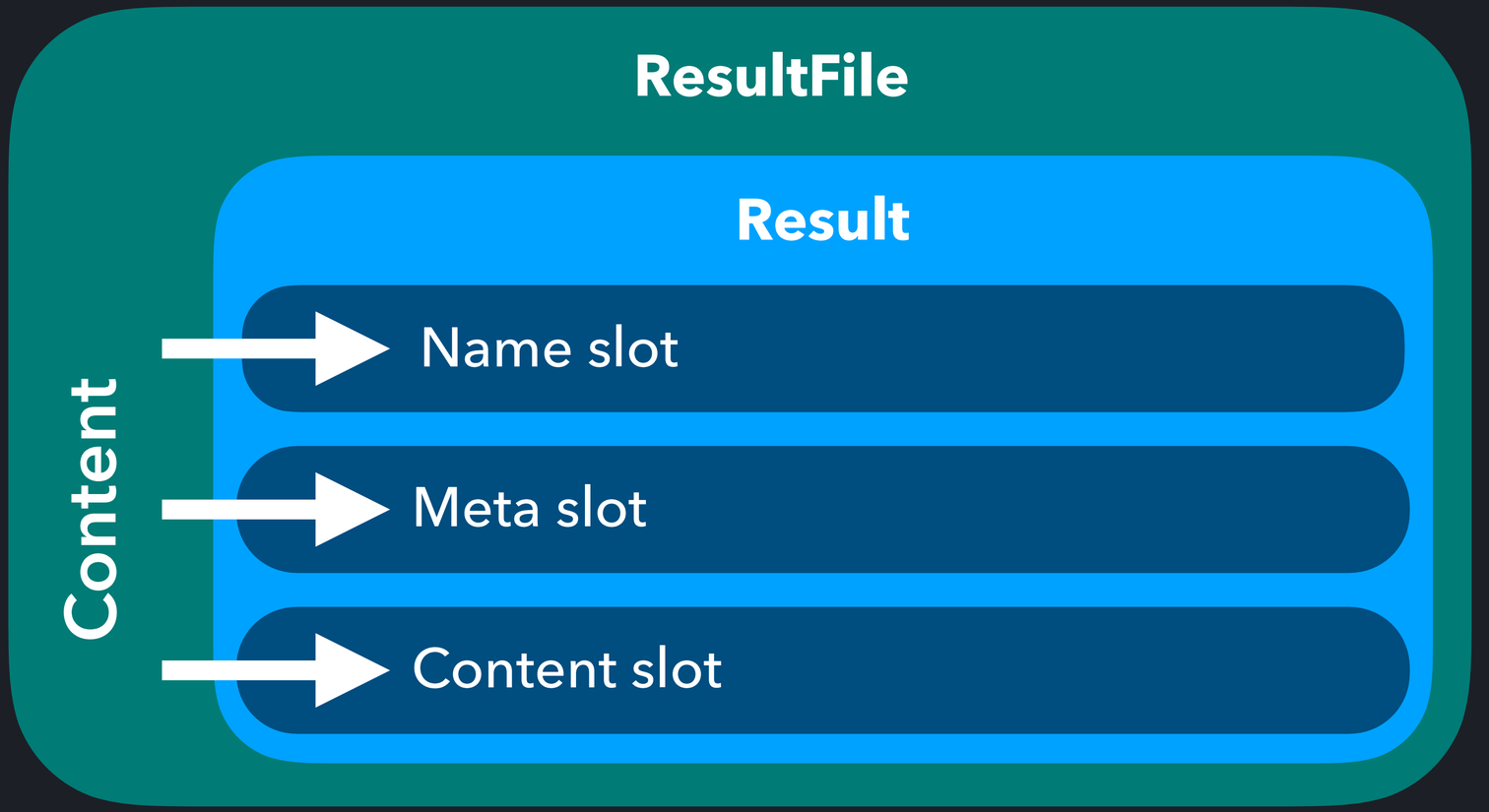
From Vue Apollo by @akryum

“Composition over inheritance”

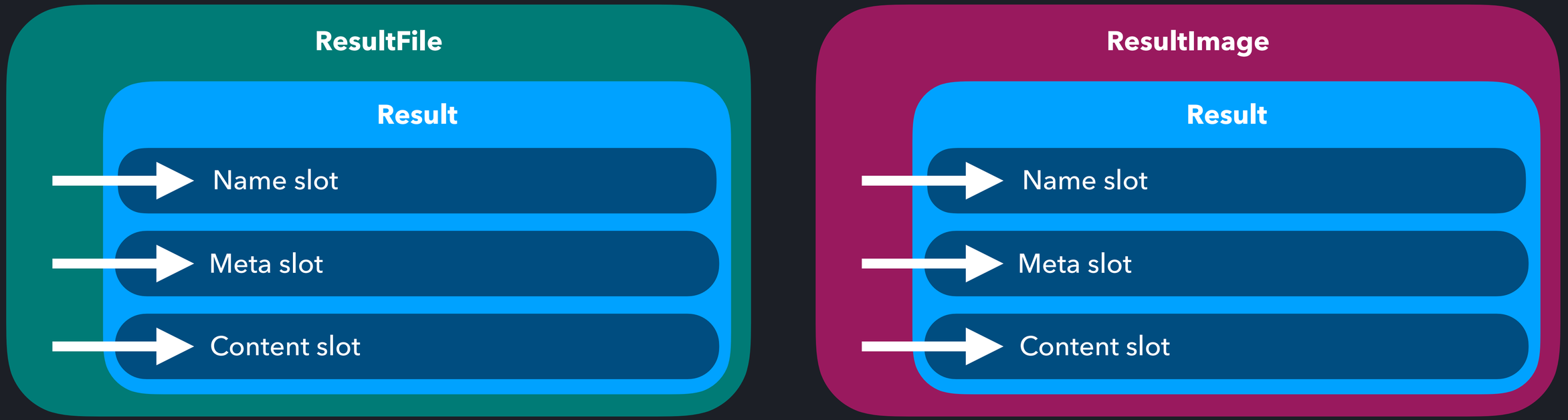
“Composition over inheritance”



“Composition over inheritance”



"Composition over inheritance"



Composition > Configuration

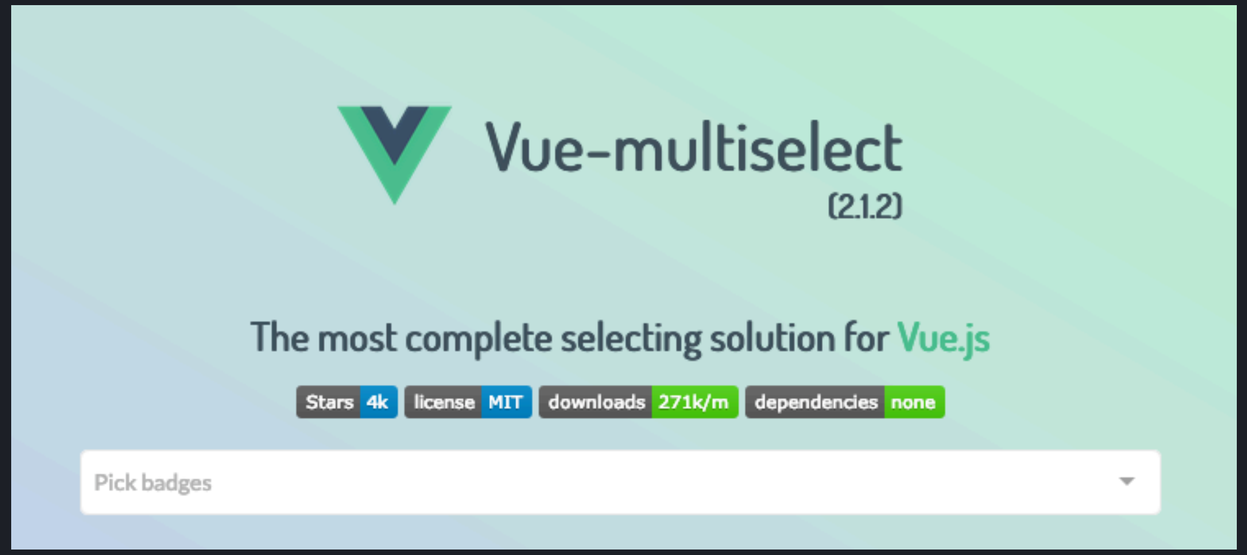
Should you always use composition
to build components?

Should you always use composition
to build components ?

No.

Often props-based solutions are good enough

Just don’t get stuck, ok?



40+ available props

~140 LoC inside <template>

What about Mixins?

It’s used for composition, right?

What about Mixins?

- Hard to separate concerns / leaking concerns
- No way to share fragments of a template
- Possible conflicts, hidden properties
- Not scalable long term

Future?


I NEED AN ANSWER NOW.

Hooks

https://css-tricks.com/what-hooks-mean-for-vue/

Sarah Drasner

Keep the things simple.

Keep the things simple.
Slots help with that.

Keep the things simple.

Slots help with that.

Refactor often.

Bonus!

Slots changes in Vue v2.6

What’s new in v2.6

Unified `v-slot` directive

```
<base-layout>
  <template slot="header">
    <h1>Here might be a page title</h1>
  </template>

  <p>Main content.</p>
  <p>And another one.</p>

  <template slot="footer">
    <p>Here's some contact info</p>
  </template>
</base-layout>
```

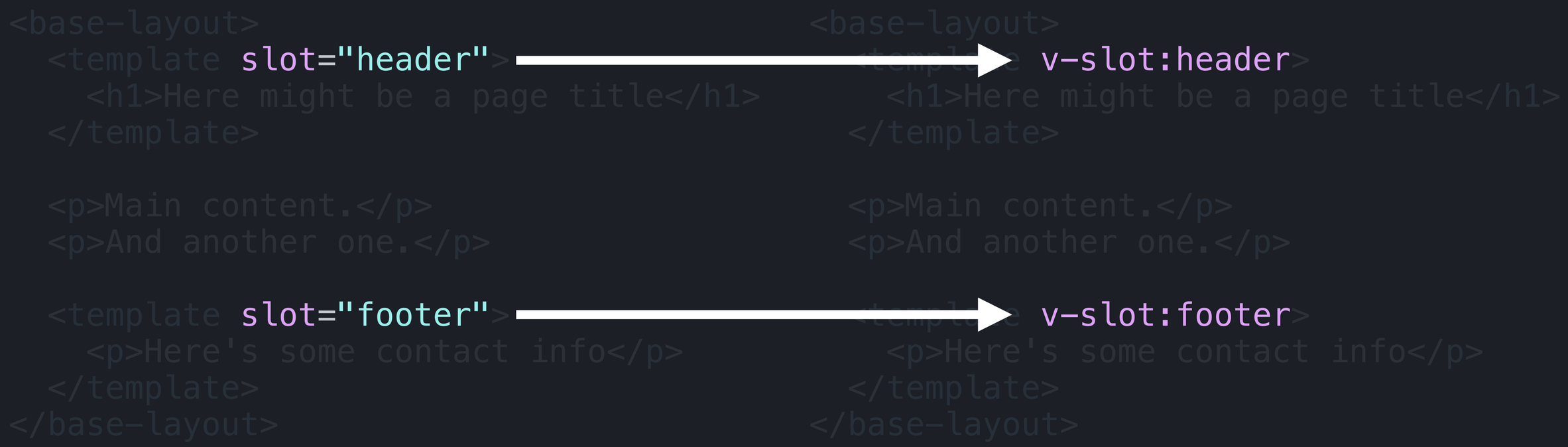
```
<base-layout>
  <template v-slot:header>
    <h1>Here might be a page title</h1>
  </template>

  <p>Main content.</p>
  <p>And another one.</p>

  <template v-slot:footer>
    <p>Here's some contact info</p>
  </template>
</base-layout>
```

What's new in v2.6

Unified `v-slot` directive



What's new in v2.6

Unified `v-slot` directive

```
<todo-list :todos="todos">
  <template slot="todo" slot-scope="{ todo }">
    {{ todo.text }}
  </template>
</todo-list>
```

```
<todo-list :todos="todos">
  <template v-slot:todo="{ todo }">
    {{ todo.text }}
  </template>
</todo-list>
```

What’s new in v2.6

Unified `v-slot` directive

```
<todo-list :todos="todos">
  <template slot="todo" slot-scope="{ todo }">
    {{ todo.text }}
  </template>
</todo-list>
```

```
<todo-list :todos="todos">
  <template v-slot:todo="{ todo }">
    {{ todo.text }}
  </template>
</todo-list>
```

What’s new in v2.6

v-slot directive shorthand

```
<todo-list :todos="todos">
  <template v-slot:todo="{ todo }">
    {{ todo.text }}
  </template>
</todo-list>
```

```
<todo-list :todos="todos">
  <template #todo=“{ todo }">
    {{ todo.text }}
  </template>
</todo-list>
```

What’s new in v2.6

`v-slot` directive shorthand

```
<todo-list :todos="todos">
  <template v-slot:todo="{ todo }">
    {{ todo.text }}
  </template>
</todo-list>
```

```
<todo-list :todos="todos">
  <template #todo="{ todo }">
    {{ todo.text }}
  </template>
</todo-list>
```

What’s new in v2.6

Dynamic Slot Names

```
<base-layout>
  <template v-slot:[dynamicSlotName]>
    ...
  </template>
</base-layout>
```

Demo

Thank you!

Damian Dulisz

GitHub: @shentao

Twitter: @damiandulisz

Questions?

Damian Dulisz

GitHub: @shentao

Twitter: @damiandulisz