



Open Source Your Components

Darren Jennings

Tech Lead, Kong

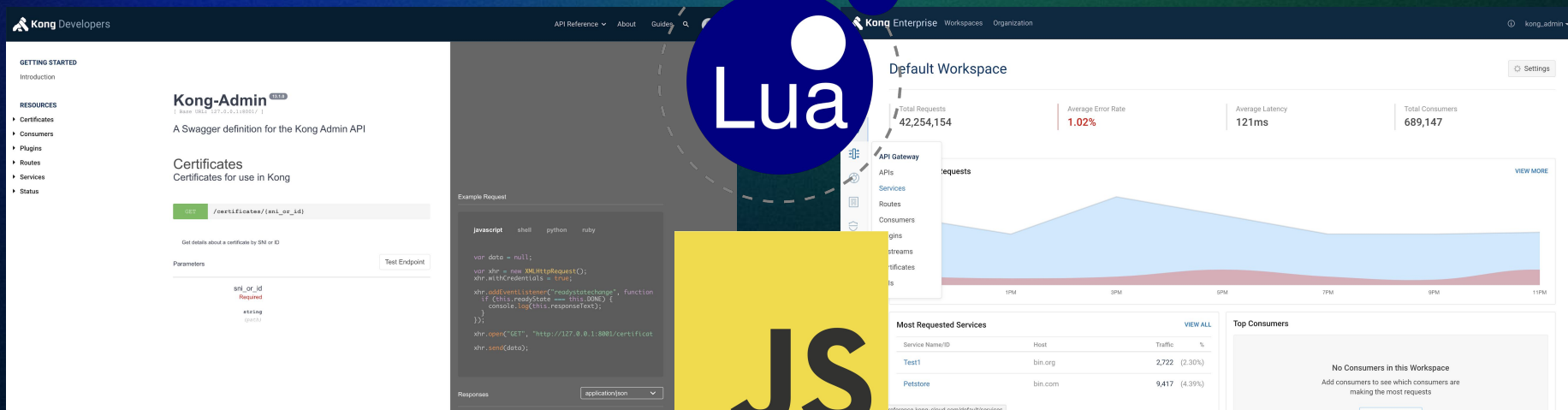
@darrenjennings · guuu.io

konghq.com

Community is the DNA of Kong



Frontend at Kong



The screenshot shows the Kong Admin interface. On the left, the 'Kong-Admin' section is visible, showing a Swagger definition for the Kong Admin API. The main area displays the 'Default Workspace' dashboard with various metrics and charts. A blue circle with the word 'Lua' is overlaid on the top right, and a yellow square with 'JS' is overlaid on the bottom right. A dashed line connects the two overlays.

Kong-Admin
A Swagger definition for the Kong Admin API

Certificates
Certificates for use in Kong

Example Request

```
var data = null;
var xhr = new XMLHttpRequest();
xhr.withCredentials = true;
xhr.addEventListener("readystatechange", function() {
  if (this.readyState === this.DONE) {
    console.log(this.responseText);
  }
});
xhr.open("GET", "http://127.0.0.1:8001/certificate");
xhr.send(data);
```

Default Workspace

Total Requests: 42,254,154
Average Error Rate: 1.02%
Average Latency: 121ms
Total Consumers: 689,147

Most Requested Services

Service Name/ID	Host	Traffic	%
Test1	bin.org	2,722	(2.30%)
Petstore	bin.com	9,417	(4.39%)

Top Consumers

No Consumers in this Workspace
Add consumers to see which consumers are making the most requests



Problems

- *Explosion* of components
- Composability / Reusability
- *Many* consumers
- #oss vs. private
- Maintenance cost
- Documentation
- Design / Prototyping

Ship it!



6 Patterns For #OSS Success



Emitting @events



Transparent wrappers



Slots



SSR



Data reducers



Provider Pattern

vue-autosuggest

1. Emitting Events over a Function Prop

onSelected → @selected

Instead of passing *onAction* as a property, instead pass in an event, and then, emit it inside your component.

1. Easier to tell which properties are functions and events
2. `this.$emit('selected', item)`
3. `@selected === v-on:selected`
4. Ergonomic for vue users

```
<template>
  <vue-autosuggest
    :suggestions="{data:['Frodo', 'Samwise']}"
    @selected="selectHandler" />
</template>

<script>
export default {
  methods: {
    selectHandler(item) {
      this.selected = item
    }
  }
}
</script>
```

2. Transparent Wrappers

Take events and pass them down through to the element that they should logically be applied to. Useful when you are wrapping a child component in multiple container divs.

- Utilize `$listeners`
- Avoid ugly nested Object props
 - `inputProps: { onClick: {} }`
- Future proof by spreading ...

```
<template>
  <vue-autosuggest
    :suggestions="[{'data': ['Frodo', 'Samwise']}]"
    @click="clickHandler"
    @selected="selectHandler" />
</template>
```

```
<template>
  <div>
    <input
      v-on="listeners"
      v-model="searchInput"
      type="text" />
    <ul>
      <li :key="item.id"
        v-for="item in filteredSuggestions"
        @click="$emit('selected', item)">
        {{ item.Name }}
      </li>
    </ul>
  </div>
</template>
```

```
computed: {
  listeners() {
    return {
      ...this.$listeners
    }
  }
}
```


VUE SCOPED SLOTS



HOW DO THEY WORK?

3. Scoped Slots






Invert control of the template by giving users the data and letting them do the rendering.



```
[
  {
    name: "Frodo",
    avatar: "frodo.png"
  },
  {
    name: "Gandalf",
    avatar: "Gandalf.png"
  }
]
```

```
<template>
  <vue-autosuggest
    :suggestions="suggestions"
    @selected="selectHandler">
```

```
</template>
```

-  Home
-  Getting Started
-  Sending And Managing Data
-  Managing Results
-  Building Search UI

VUE 
[What is Vue InstantSearch?](#)
[Getting started](#)
[Installation](#)
[Upgrade guides](#)

WIDGETS

[Widgets showcase](#)
[Customize an existing widget](#)
[Create your own widgets](#)

GOING FURTHER

[Conditional display](#)
[Security](#)
[Routing URLs](#)
[Plug analytics](#)
[Improve performance](#)

Results page with autocomplete

This is focused on integrating a `ais-search-box` with an autocomplete linked to a results page. To implement this we use the library [Vue Autosuggest](#) that provides a component to create an autocomplete menu. Once we have this component, we need to wrap it with our `ais-autocomplete` component. The component exposes three interesting props for this use case: `indices`, and `refine`. `indices` contains the list of suggestions, and `refine` is a function that takes a query to retrieve our relevant suggestions.

vue

```

1  <template>
2    <div>
3      <ais-instant-search
4        :search-client="searchClient"
5        index-name="demo_ecommerce"
6      >
7        <ais-configure
8          :hitsPerPage="5"
9          :restrictSearchableAttributes="['name']"
10       />
11       <ais-autocomplete>
12         <template slot-scope="{ currentRefinement, indices, refine }">
13           <vue-autosuggest
14             :suggestions="indicesToSuggestions(indices)"
15             @selected="onSelect"
16             :input-props="{
17               style: 'width: 100%',
18               onChange: refine,
19               placeholder: 'Search here...',
20             }"
21           >

```

 Vue 

ON THIS PAGE

[Overview](#)
[Results page with autocomplete](#)

```
<ais-autocomplete>
  <template slot-scope="{ currentRefinement, indices, refine }">
    <vue-autosuggest
      :suggestions="indicesToSuggestions(indices)"
      @selected="onSelect"
      :input-props="{
        style: 'width: 100%',
        onChange: refine,
        placeholder: 'Search here...',
      }"
    >
      <template slot-scope="{ suggestion }">
        <ais-highlight
          :hit="suggestion.item"
          attribute="name"
          v-if="suggestion.item.name"
        />
        <strong>$ {{ suggestion.item.price }}</strong>
        
      </template>
    </vue-autosuggest>
  </template>
</ais-autocomplete>
```

4. Server Side Rendering

Even if you are not using it today, users of your components might use SSR.

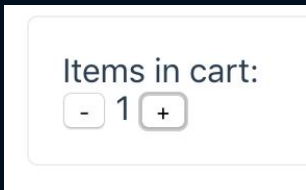
DOs and DO NOTs

- **DO** remember that Node.js and the Browser are two different environments so things like `window` and `document` are not available on the server.
- **DO** put global side-effects in mounted such as `addEventListener` or `setTimeout`.
- **DO NOT** put anything in `beforeMount` or `mounted()` that changes the template or alters the state.
- **DO** use Nuxt.js as it gives you a ton of behavior out of the box such as prefetching, html generation, and component caching. Use Codesandbox Nuxt.js template to try it out.



5. Data Reducers

- Even more control!



```
<template>
  <counter :data-reducer="counterReduce"/>
</template>

<script>
import counter from "../components/counter";

export default {
  components: { counter },
  methods: {
    counterReduce(state, changes) {
      return {
        ...changes,
        count: changes.count > 3 ? 3 : changes.count
      }
    }
  }
}
</script>
```

<https://guuu.io/beautiful-vue/data-reducer/>

```
<template>
  <div>
    <button @click="crement(-1)">-</button>
    {{ count }}
    <button @click="crement(1)">+</button>
  </div>
</template>

<script>

// Iterates over the changes to set $data on vue instance
function setData(vm, changes) {
  for(let item in changes) {
    vm[item] = changes[item]
    vm.$nextTick()
  }
}

export default {
  name: "counter",
  props: {
    dataReducer: {
      type: Function,
      required: false,
      default: (state, changes) => {
        return {
          ...changes
        }
      }
    },
  },
  data() {
    return {
      count: 0
    };
  },
  methods: {
    crement(amount) {
      const changes = this.dataReducer(this.$data, { count: this.count + amount })
      setData(this, changes)
    }
  }
};
</script>
```

6. Provider Pattern

Component that renders another component to provide or inject some context or state.

- Composition over inheritance
- Child components can remain the same.
- Turn on/off functionality
- Renders child using default slot



```
<template>
  <div>
    <slot :is-allowed="ok"/>
  </div>
</template>
```



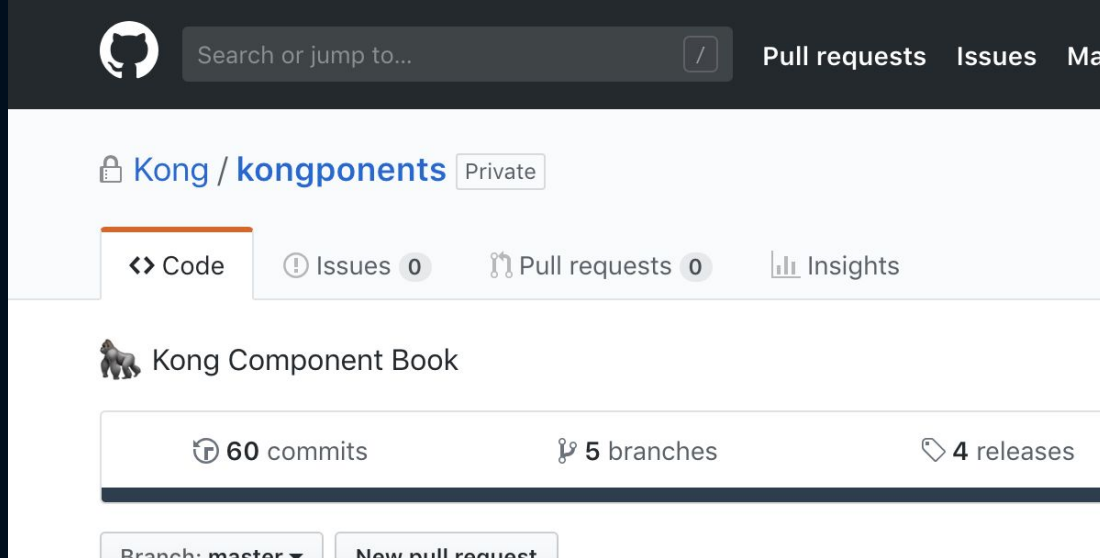
```
<RbacValidate :restricted="{ {
  path: '/services',
  actions: ['read']
} }">
  <template scope="{ isAllowed }">
    <div v-if="isAllowed">
      <Entity />
    </div>
    <div v-else>
      <EmptyState />
    </div>
  </template>
</RbacValidate>
```



Kongponents



- Internal Kong Vue Component library
- Published privately to NPM
- Independent Versioning
- Monorepo
 - Lerna
 - Yarn Workspaces
- Storybook
- `kpm` CLI for better DX
- Jest, vue-test-utils
- Future:
 - UMD
 - css variables, theming
 - #oss
 - enhanced CLI
 - Vuepress
 - ???



```
"dependencies": {
  "@kongponents/kalert": "0.0.1-beta.3",
  "@kongponents/kcard": "^0.0.1-beta.4",
  "@kongponents/kemptystate": "^0.0.1-beta.3",
  "@kongponents/kicon": "0.0.1-beta.6",
  "@kongponents/ktable": "0.0.1-beta.3",
```



Darren Jennings

@darrenjennings



JS DO IT #nike

3:28 PM - 10 Oct 2018 from San Francisco, CA

1 Like



Thanks Niji!

“If we don’t talk about what we love
and why, we’re doomed.”

@DavidDark

Without patterns:

My technology rocks
Your technology sucks

Blow up the Monolith
The Monolith is majestic



With patterns:

Benefits, Drawbacks *Choices*

Microservices fit our business needs.
Monoliths are right for our organization at this time.



Demo

We're Hiring!

<https://konghq.com/careers/>

Questions?