QUASAR

BEYOND THE FRAMEWORK

Dreamonkey

A Deep Dive

QUASAR AND TYPESCRIPT

Who I am

Yusuf Kandemir

- Istanbul, Türkiye
- Full-stack developer
- Employee at Dreamonkey
- Quasar core team member, know-it-all Wizard

Github: @yusufkandemir

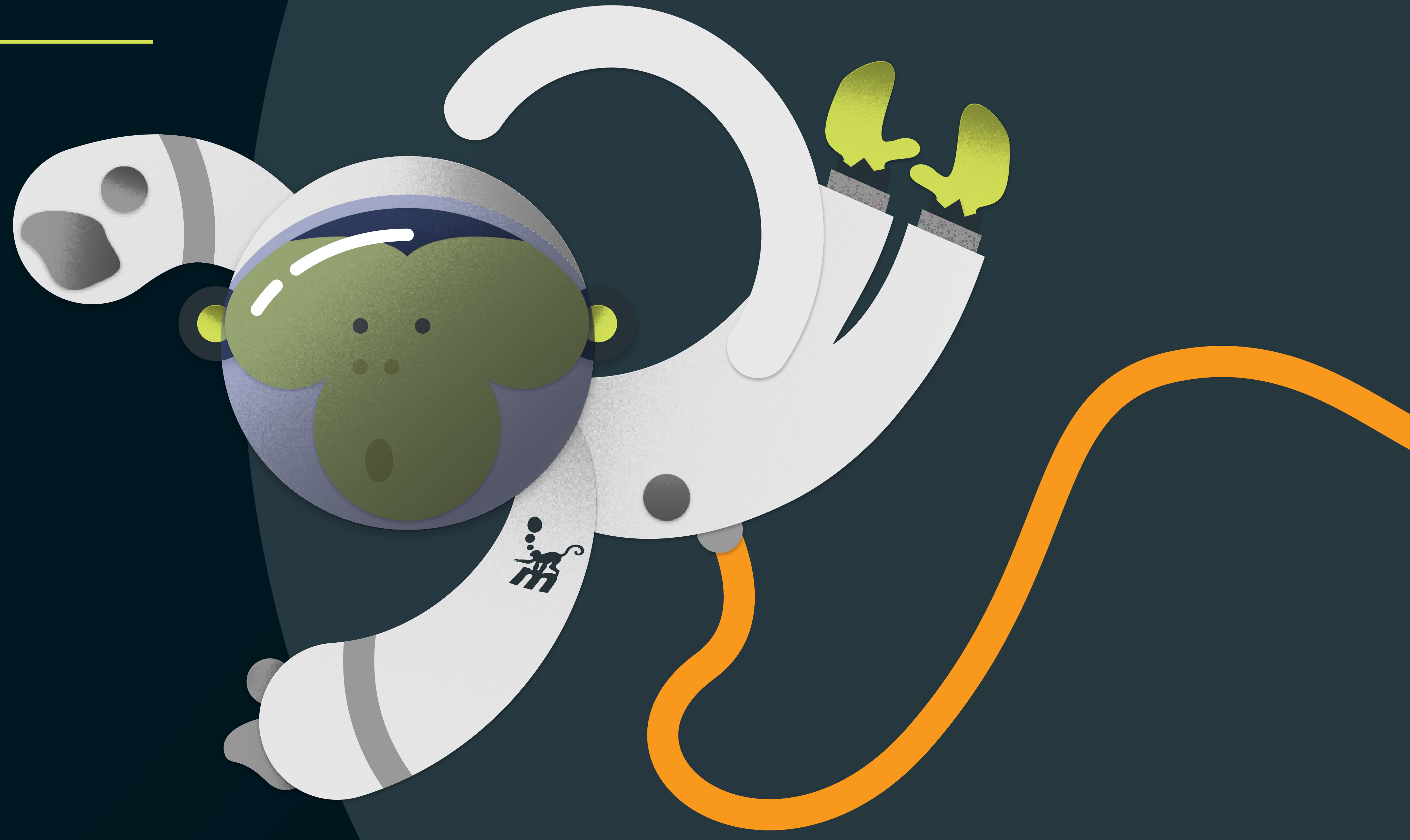
Twitter: @yusuf1kandemir

Dreamonkey

Working at Dreamonkey since 2021

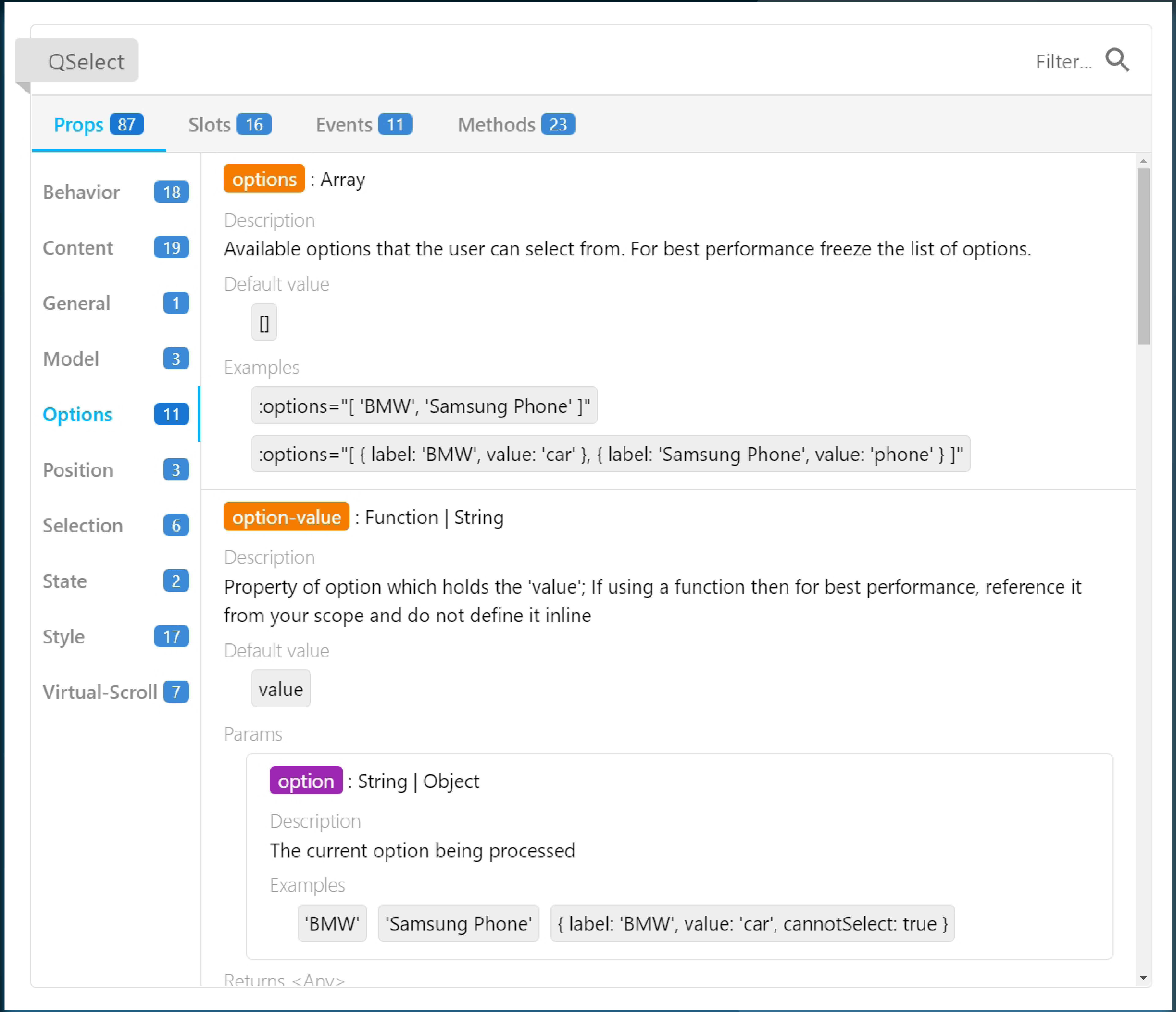
Company insights relevant to me

- 1-day dedicated to internal R&D projects or open source
- Remote-friendly
- People-oriented
- Continuous team feedback/communication
- Periodic 1-to-1s among developers
- Company funded courses



Quasar Docs API Card

Used a lot, but what's behind it?



QSelect

Filter...

Props 87 | Slots 16 | Events 11 | Methods 23

Behavior 18
Content 19
General 1
Model 3
Options 11
Position 3
Selection 6
State 2
Style 17
Virtual-Scroll 7

options : Array

Description

Available options that the user can select from. For best performance freeze the list of options.

Default value

```
[]
```

Examples

```
:options="[ 'BMW', 'Samsung Phone' ]"
```

```
:options="[ { label: 'BMW', value: 'car' }, { label: 'Samsung Phone', value: 'phone' } ]"
```

option-value : Function | String

Description

Property of option which holds the 'value'; If using a function then for best performance, reference it from your scope and do not define it inline

Default value

```
value
```

Params

option : String | Object

Description

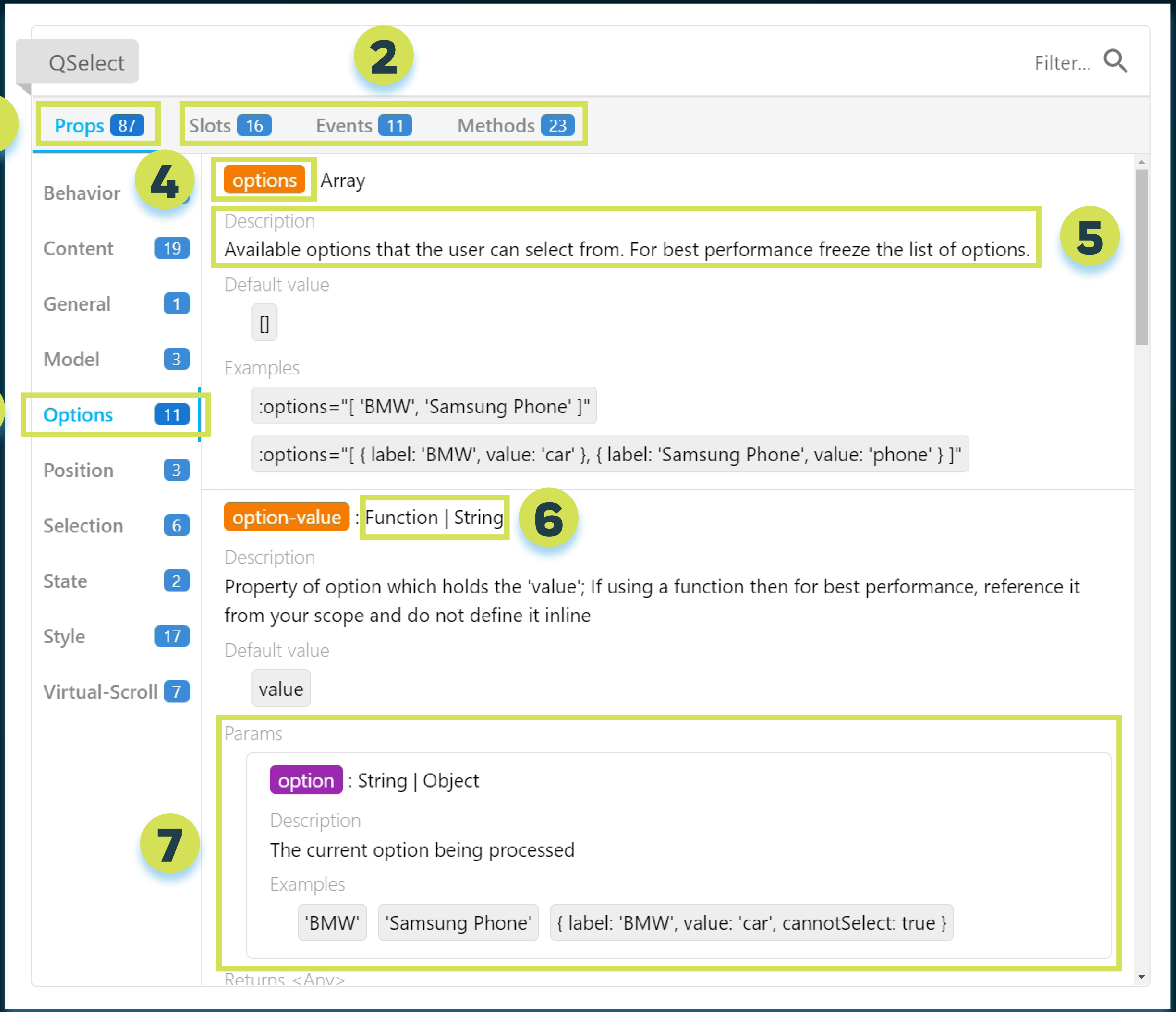
The current option being processed

Examples

`'BMW'` `'Samsung Phone'` `{ label: 'BMW', value: 'car', cannotSelect: true }`

Returns <Any>

Docs API Card and JSON API



options Array

Description
Available options that the user can select from. For best performance freeze the list of options.

Default value
[]

Examples
:options="['BMW', 'Samsung Phone']"
:options="[{ label: 'BMW', value: 'car' }, { label: 'Samsung Phone', value: 'phone' }]"

option-value : Function | String

Description
Property of option which holds the 'value'; If using a function then for best performance, reference it from your scope and do not define it inline

Default value
value

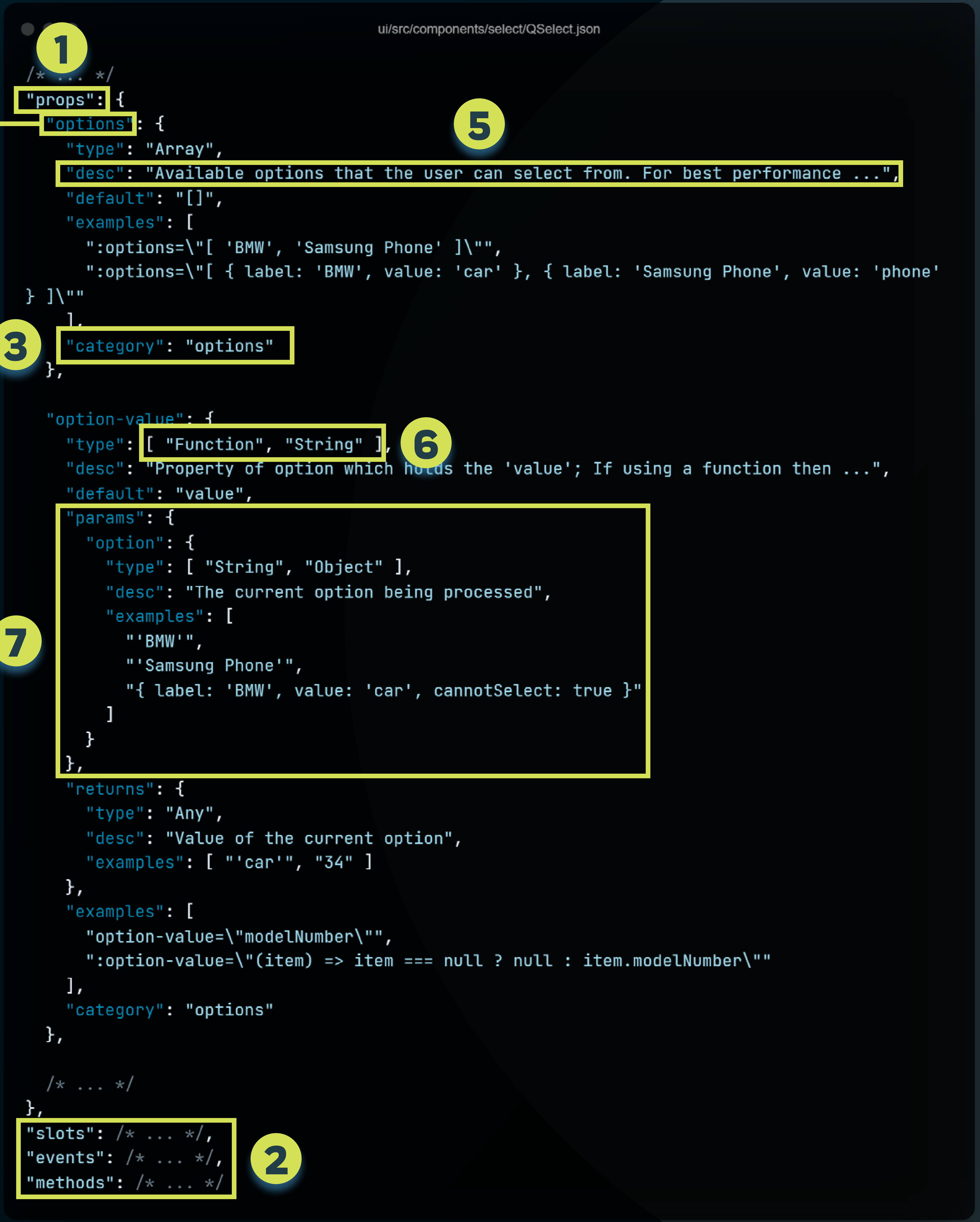
Params

option : String | Object

Description
The current option being processed

Examples
'BMW' 'Samsung Phone' { label: 'BMW', value: 'car', cannotSelect: true }

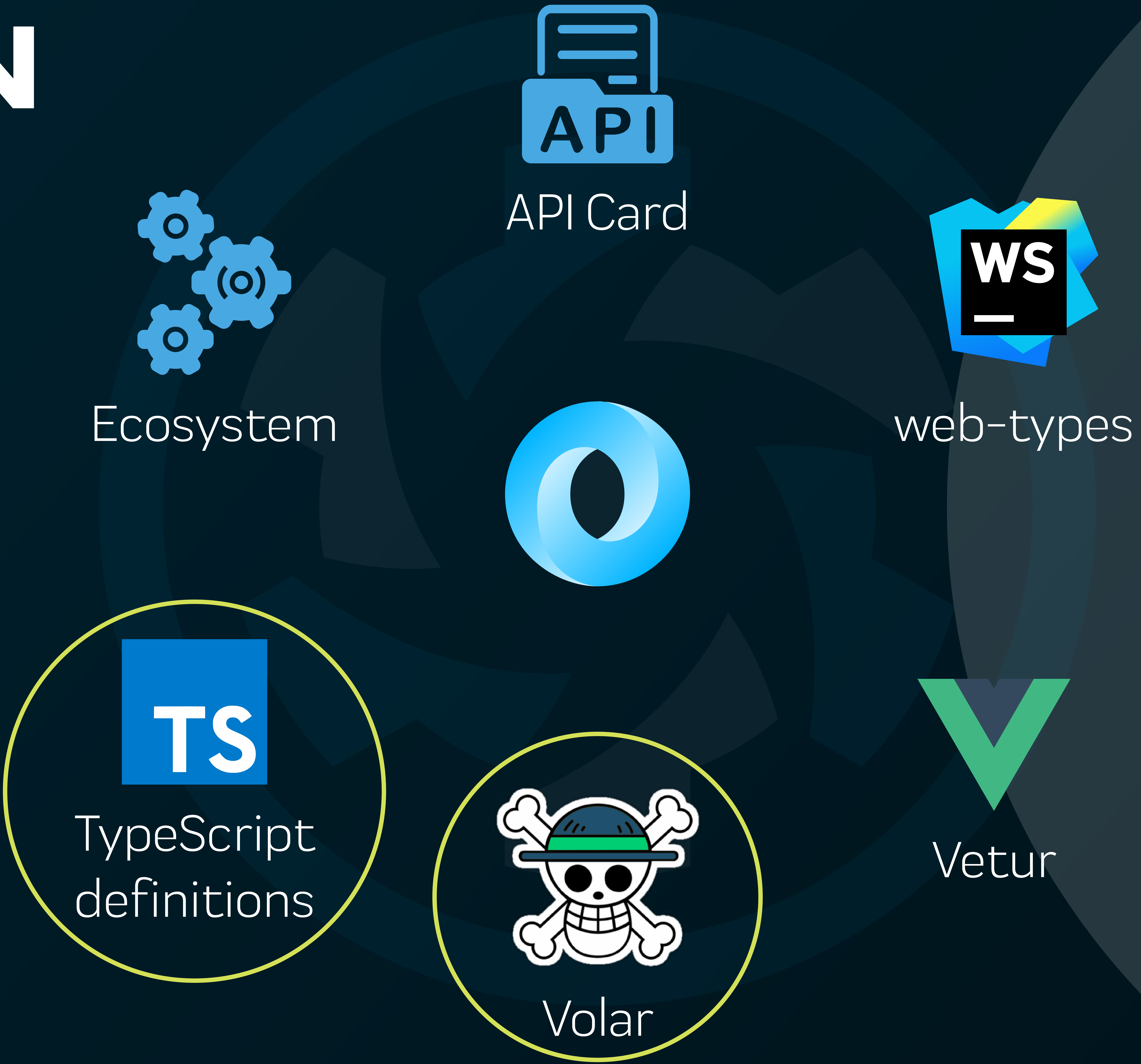
Returns <Any>



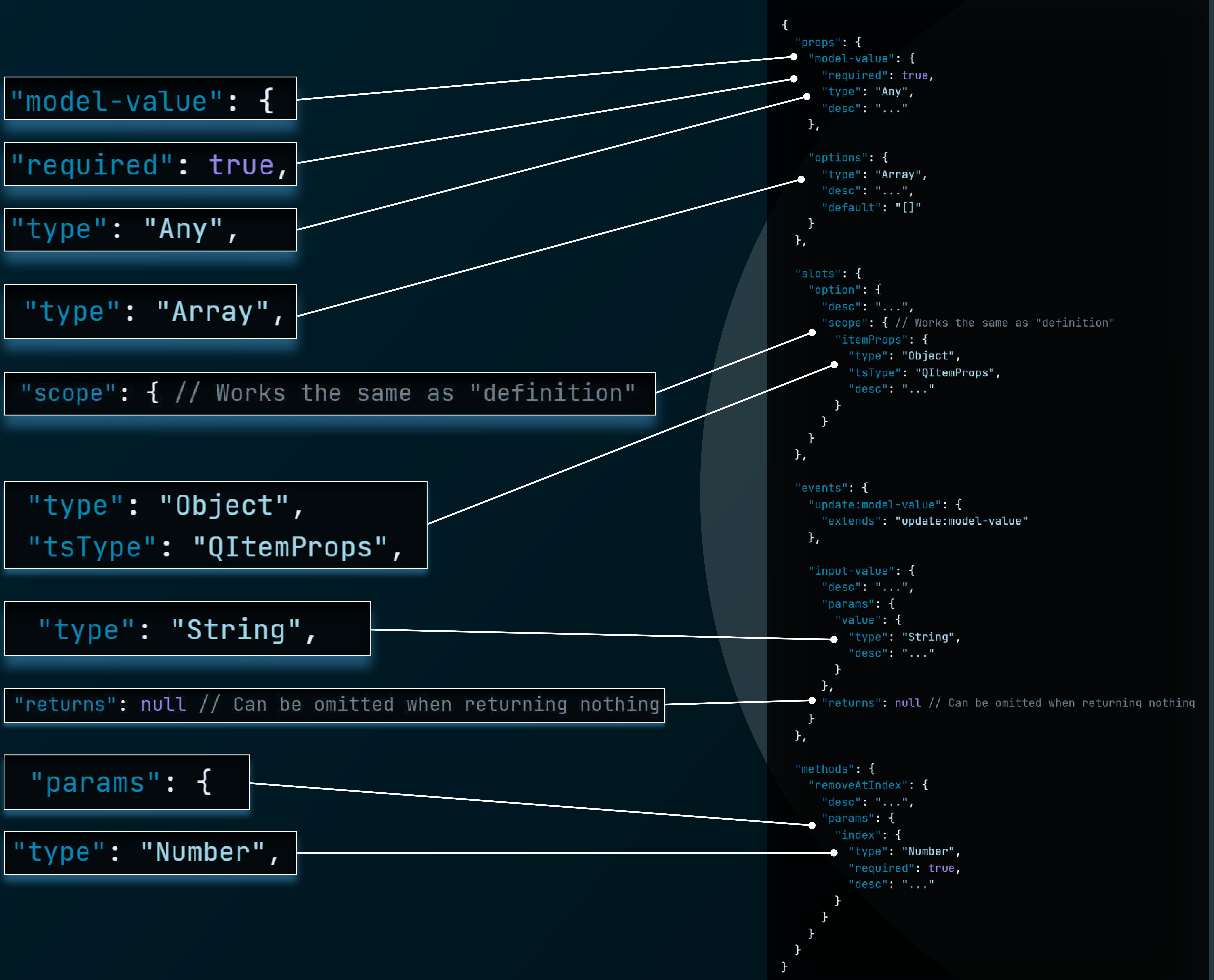
```
/* ... */
"props": {
  "options": {
    "type": "Array",
    "desc": "Available options that the user can select from. For best performance ...",
    "default": "[]",
    "examples": [
      ":options=\"[ 'BMW', 'Samsung Phone' ]\"",
      ":options=\"[ { label: 'BMW', value: 'car' }, { label: 'Samsung Phone', value: 'phone'
} ]\""
    ],
    "category": "options"
  },

  "option-value": {
    "type": [ "Function", "String" ],
    "desc": "Property of option which holds the 'value'; If using a function then ...",
    "default": "value",
    "params": {
      "option": {
        "type": [ "String", "Object" ],
        "desc": "The current option being processed",
        "examples": [
          "'BMW'",
          "'Samsung Phone'",
          "{ label: 'BMW', value: 'car', cannotSelect: true }"
        ]
      }
    },
    "returns": {
      "type": "Any",
      "desc": "Value of the current option",
      "examples": [ "'car'", "34" ]
    },
    "examples": [
      "option-value=\"modelNumber\"",
      ":option-value=\"(item) => item === null ? null : item.modelNumber\""
    ],
    "category": "options"
  },

  /* ... */
},
"slots": /* ... */,
"events": /* ... */,
"methods": /* ... */
```

JSON API

The hidden core



Generating TS using JSON API

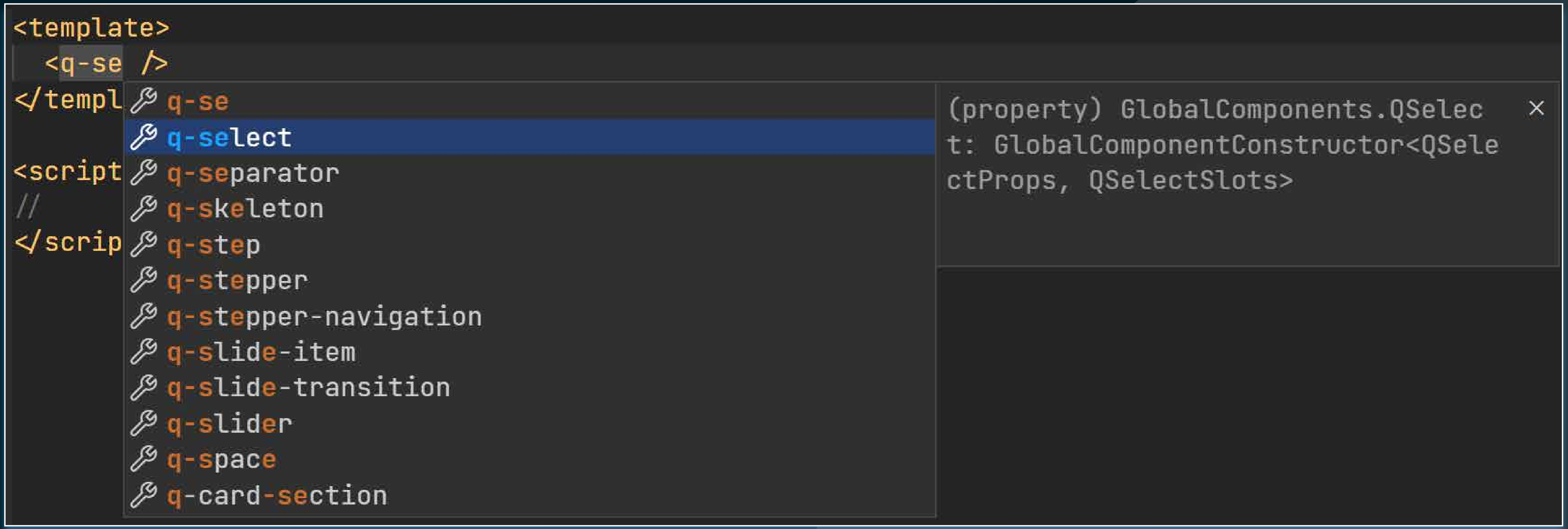


```
{
  "props": {
    "model-value": {
      "required": true,
      "type": "Any",
      "desc": "..."
    },

    "options": {
      "type": "Array",
      "desc": "...",
      "default": "[]"
    }
  },

  "slots": {
    "option": {
      "desc": "...",
      "scope": { // Works the same as "definition"
        "itemProps": {
          "type": "Object",
          "tsType": "QItemProps",
          "desc": "..."
        }
      }
    }
  },

  "events": {
    "update:model-value": {
      "extends": "update:model-value"
    },

    "input-value": {
      "desc": "...",
      "params": {
        "value": {
          "type": "String",
          "desc": "..."
        }
      },
      "returns": null // Can be omitted when returning nothing
    }
  },

  "methods": {
    "removeAtIndex": {
      "desc": "...",
      "params": {
        "index": {
          "type": "Number",
          "required": true,
          "desc": "..."
        }
      }
    }
  }
}
```

Volar Support: features

1

Component name auto-complete



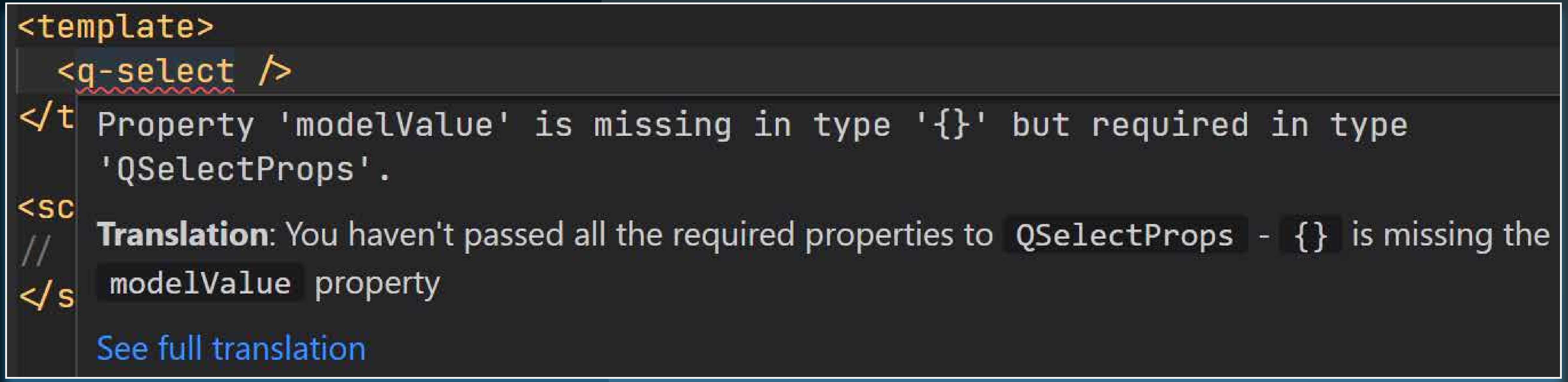
```
<template>
  <q-se />
</templ
<script
//
</scrip
```

```
q-se
q-select
q-separator
q-skeleton
q-step
q-stepper
q-stepper-navigation
q-slide-item
q-slide-transition
q-slider
q-space
q-card-section
```

```
(property) GlobalComponents.QSelect: GlobalComponentConstructor<QSelectProps, QSelectSlots>
```

2

Required prop validation



```
<template>
  <q-select />
</t
<sc
//
</s
```

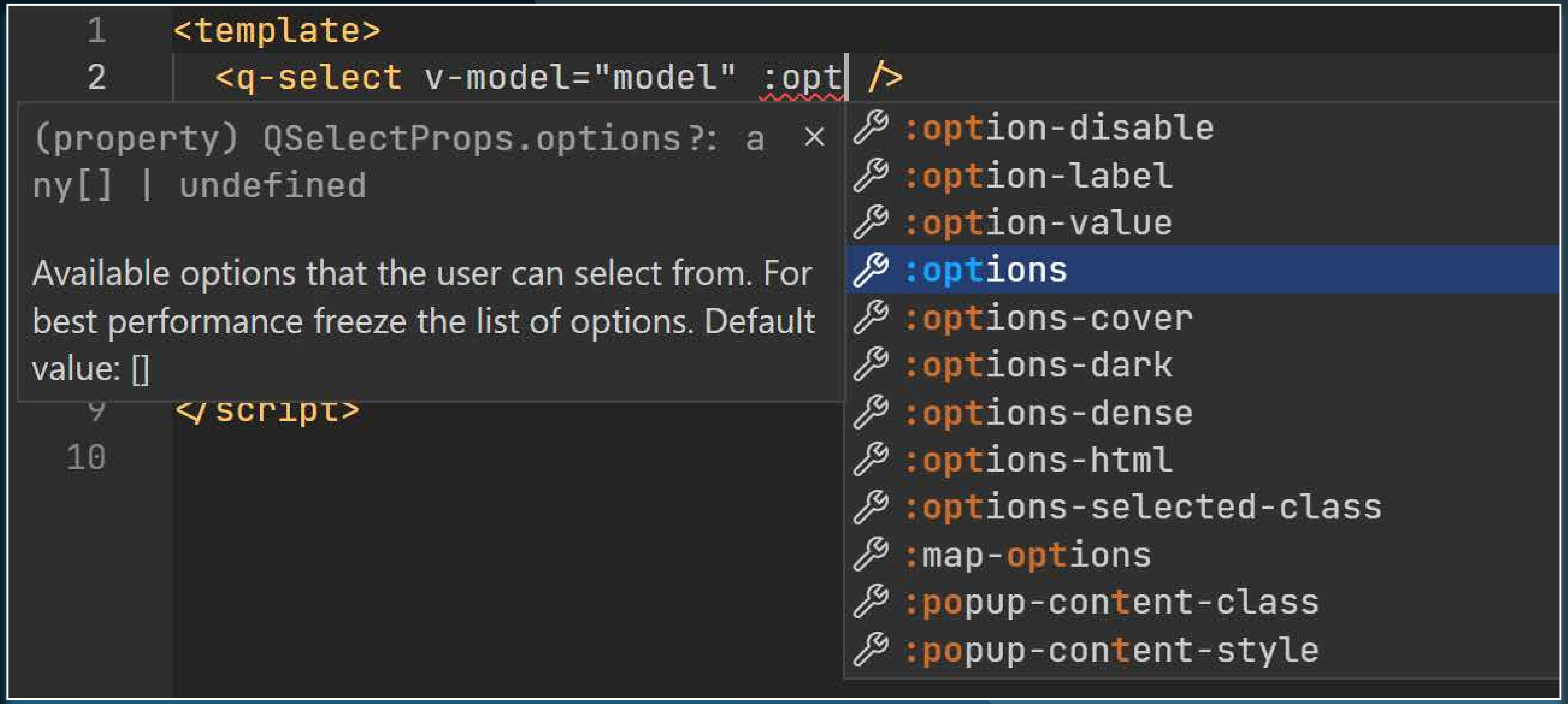
```
Property 'modelValue' is missing in type '{}' but required in type
'QSelectProps'.
```

Translation: You haven't passed all the required properties to `QSelectProps` - `{}` is missing the `modelValue` property

See full translation

3

Prop auto-complete, types, and description



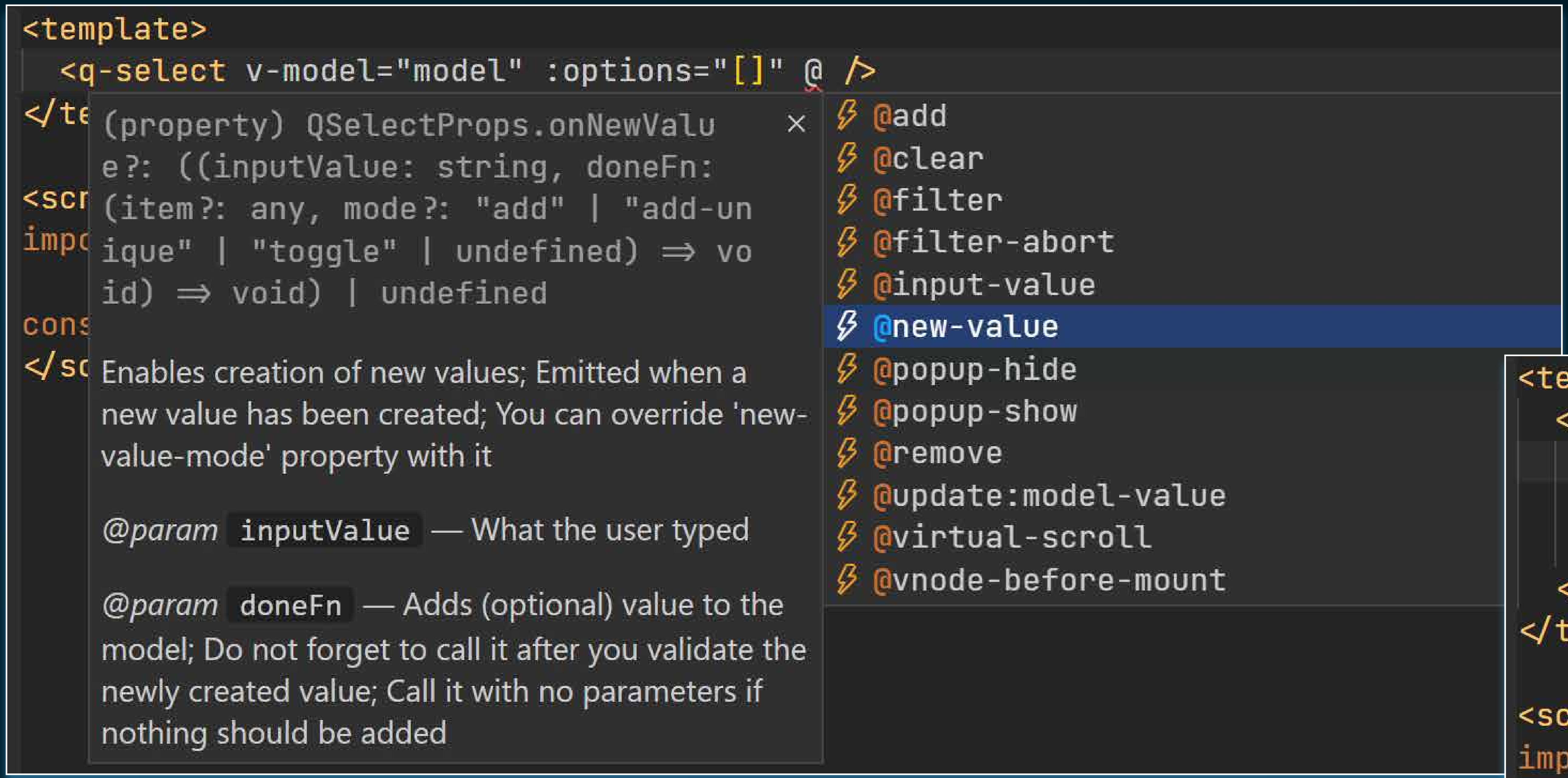
```
<template>
  <q-select v-model="model" :opt />
</script>
```

```
(property) QSelectProps.options?: any[] | undefined
```

Available options that the user can select from. For best performance freeze the list of options. Default value: []

```
:option-disable
:option-label
:option-value
:options
:options-cover
:options-dark
:options-dense
:options-html
:options-selected-class
:map-options
:popup-content-class
:popup-content-style
```

4 Event auto-complete



```
<template>
  <q-select v-model="model" :options="[]" @ />
</te
<scr
impo
cons
</sc
```

```
(property) QSelectProps.onNewValu
e?: ((inputValue: string, doneFn:
(item?: any, mode?: "add" | "add-un
ique" | "toggle" | undefined) ⇒ vo
id) ⇒ void) | undefined
```

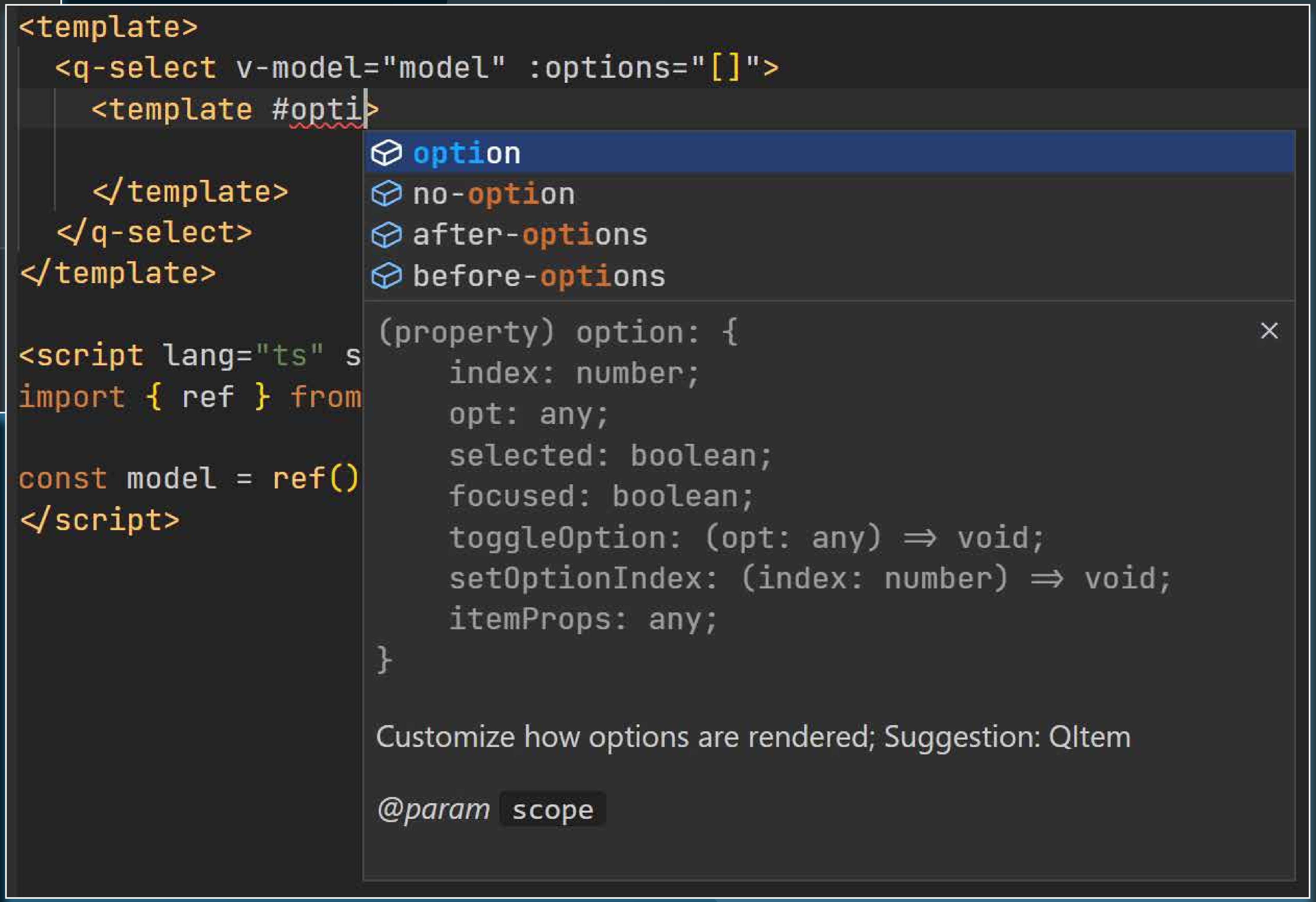
Enables creation of new values; Emitted when a new value has been created; You can override 'new-value-mode' property with it

@param `inputValue` — What the user typed

@param `doneFn` — Adds (optional) value to the model; Do not forget to call it after you validate the newly created value; Call it with no parameters if nothing should be added

```
@add
@clear
@filter
@filter-abort
@input-value
@new-value
@popup-hide
@popup-show
@remove
@update:model-value
@virtual-scroll
@vnode-before-mount
```

5 Slot auto-complete, types, and description



```
<template>
  <q-select v-model="model" :options="[]">
    <template #opti>

    </template>
  </q-select>
</template>

<script lang="ts" s
import { ref } from

const model = ref()
</script>
```

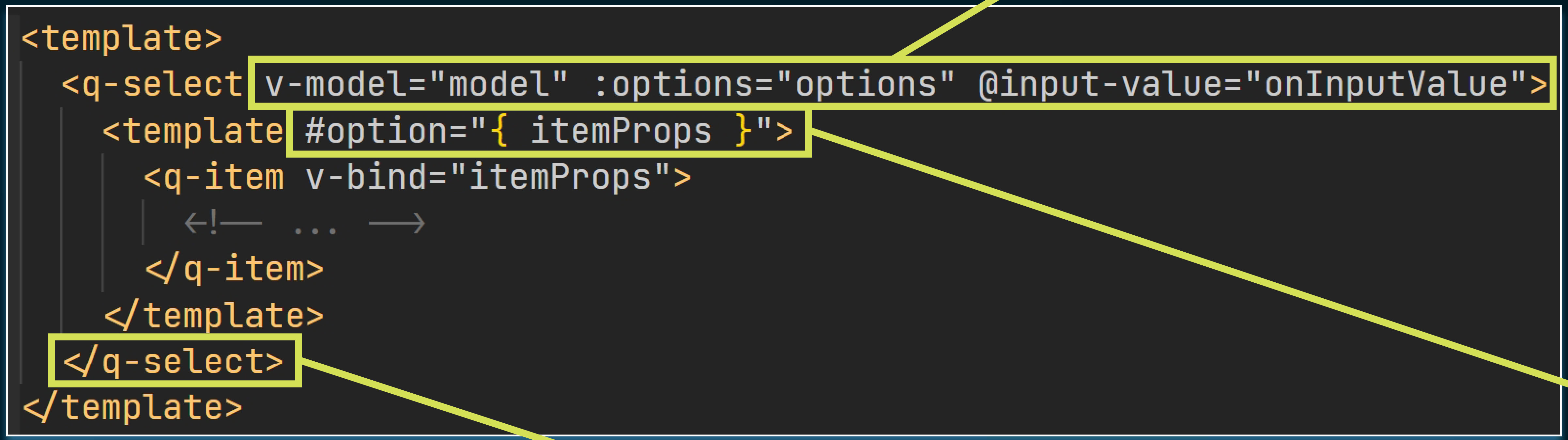
```
option
no-option
after-options
before-options
```

```
(property) option: {
    index: number;
    opt: any;
    selected: boolean;
    focused: boolean;
    toggleOption: (opt: any) ⇒ void;
    setOptionIndex: (index: number) ⇒ void;
    itemProps: any;
}
```

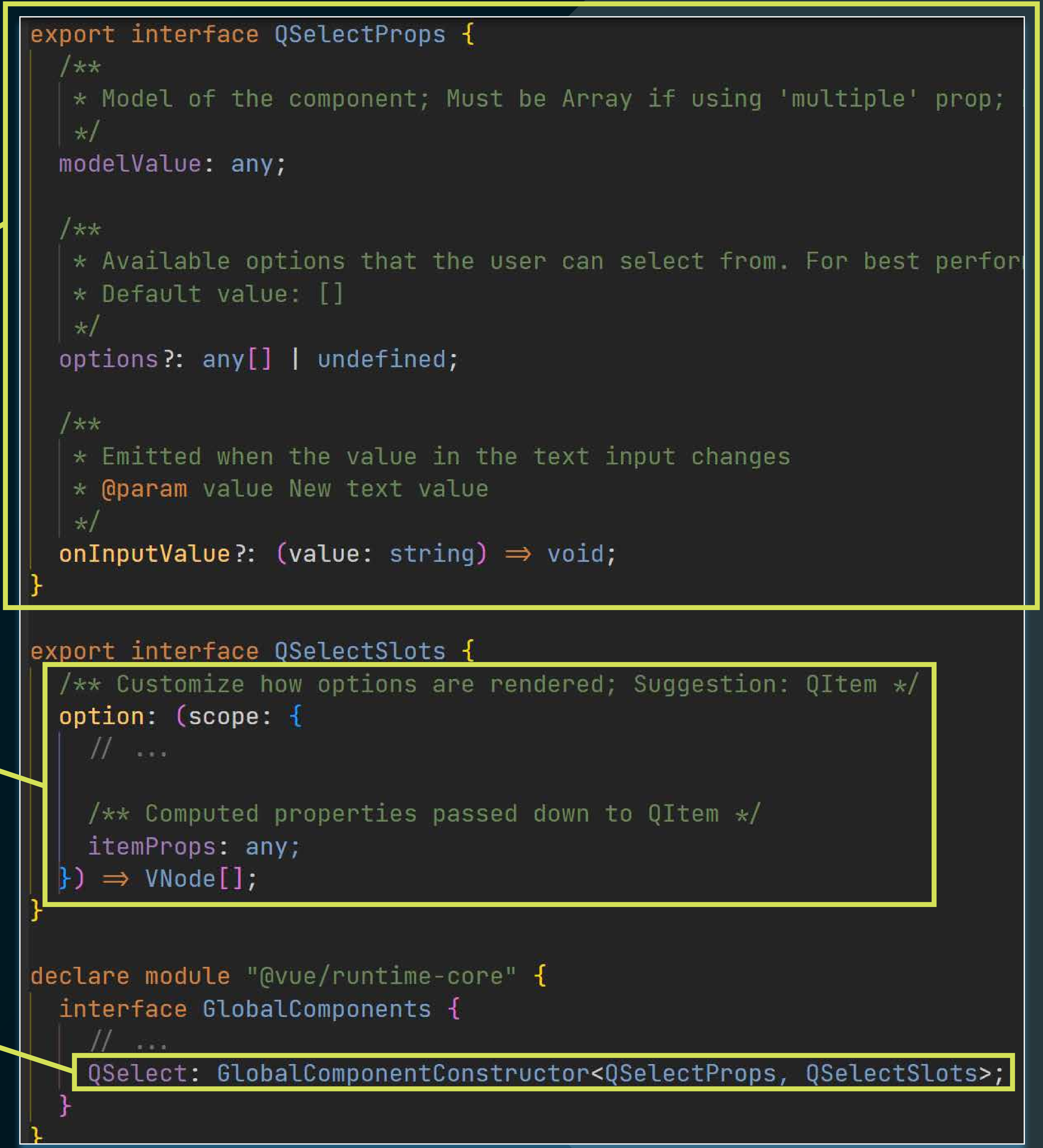
Customize how options are rendered; Suggestion: QItem

@param `scope`

Volar Support: under the hood



```
<template>
  <q-select v-model="model" :options="options" @input-value="onInputValue">
    <template #option="{ itemProps }">
      <q-item v-bind="itemProps">
        <!-- ... -->
      </q-item>
    </template>
  </q-select>
</template>
```

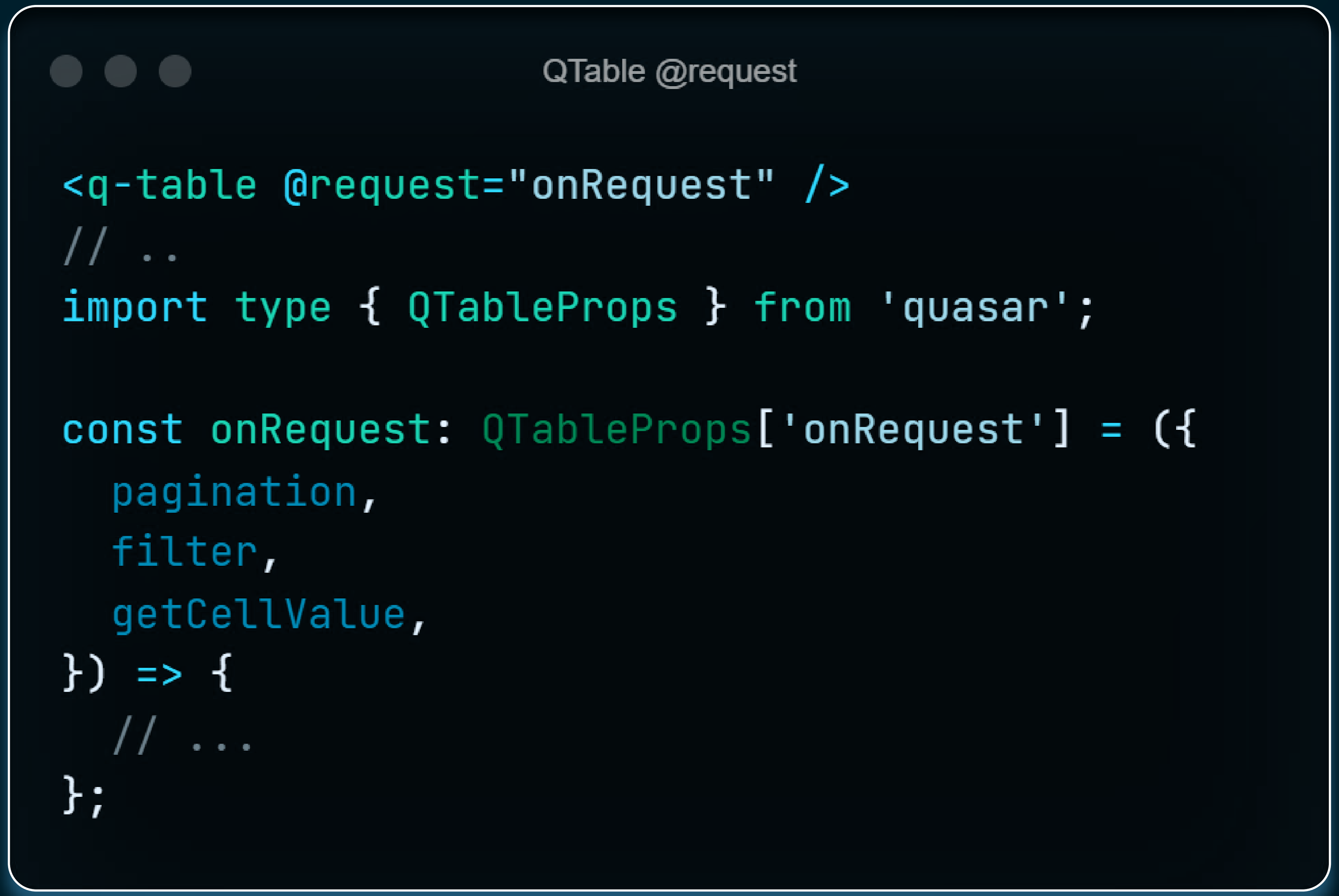


```
export interface QSelectProps {
  /**
   * Model of the component; Must be Array if using 'multiple' prop;
   */
  modelValue: any;

  /**
   * Available options that the user can select from. For best perfor
   * Default value: []
   */
  options?: any[] | undefined;

  /**
   * Emitted when the value in the text input changes
   * @param value New text value
   */
  onInputValue?: (value: string) => void;
}

export interface QSelectSlots {
  /** Customize how options are rendered; Suggestion: QItem */
  option: (scope: {
    // ...

    /** Computed properties passed down to QItem */
    itemProps: any;
  }) => VNode[];
}

declare module "@vue/runtime-core" {
  interface GlobalComponents {
    // ...
    QSelect: GlobalComponentConstructor<QSelectProps, QSelectSlots>;
  }
}
```

Utilizing All the Types

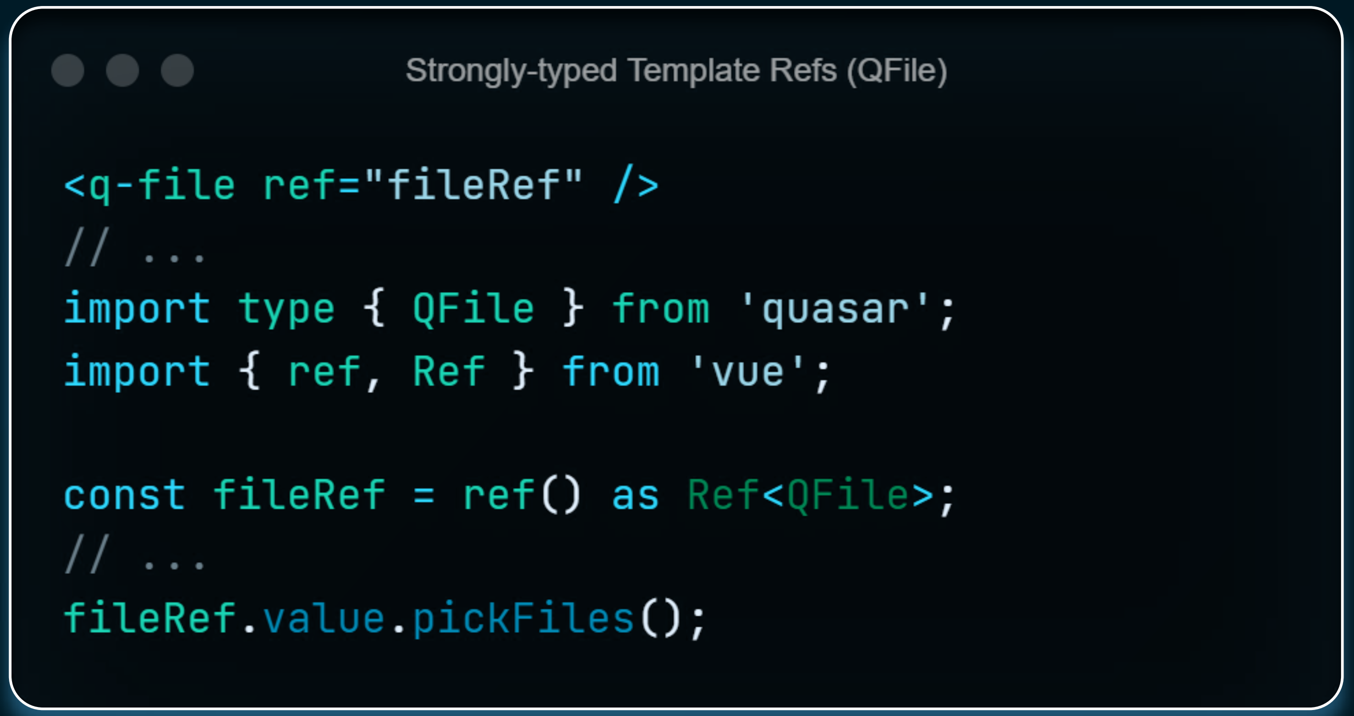
Generated types

QTable @request



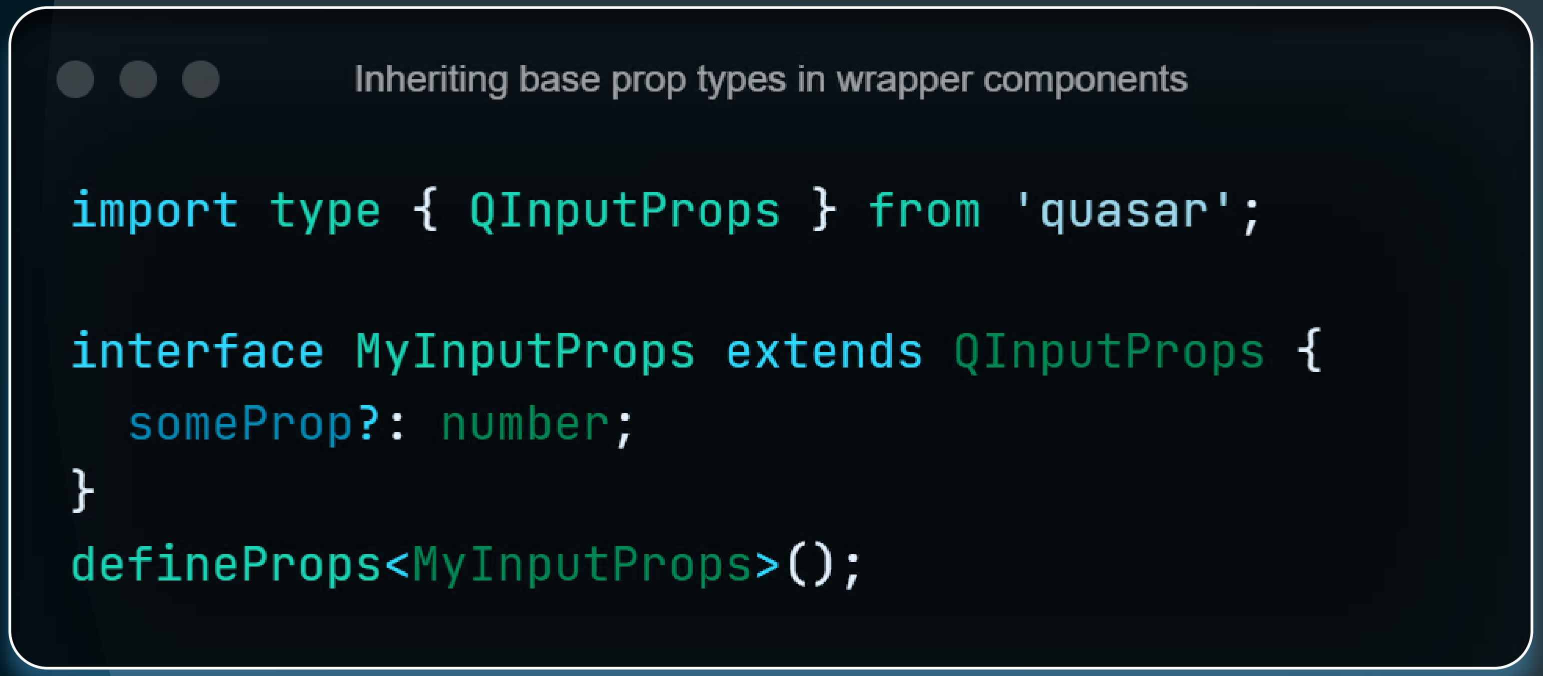
```
<q-table @request="onRequest" />
// ..
import type { QTableProps } from 'quasar';

const onRequest: QTableProps['onRequest'] = ({
  pagination,
  filter,
  getCellValue,
}) => {
  // ...
};
```

Strongly-typed Template Refs (QFile)



```
<q-file ref="fileRef" />
// ...
import type { QFile } from 'quasar';
import { ref, Ref } from 'vue';

const fileRef = ref() as Ref<QFile>;
// ...
fileRef.value.pickFiles();
```

Inheriting base prop types in wrapper components

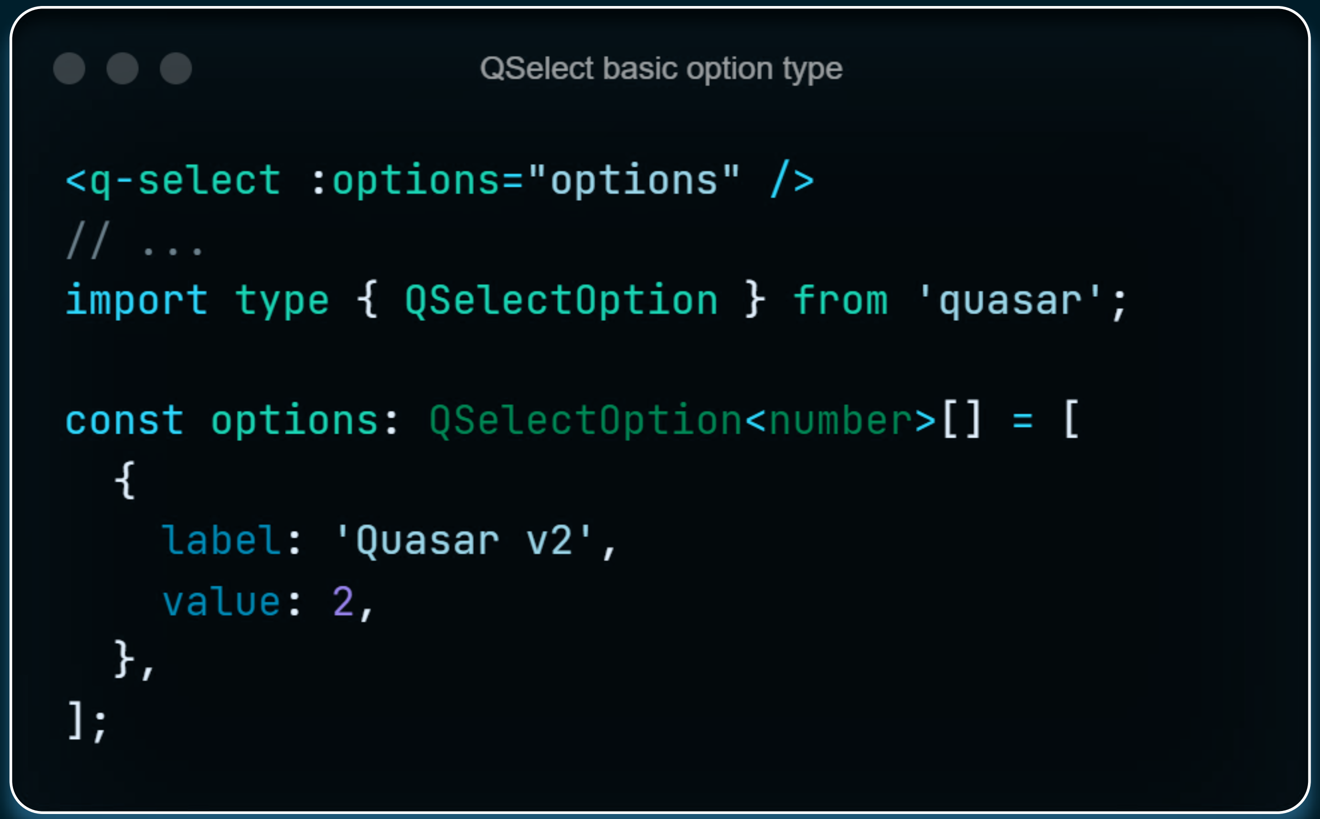


```
import type { QInputProps } from 'quasar';

interface MyInputProps extends QInputProps {
  someProp?: number;
}
defineProps<MyInputProps>();
```

Utilizing All the Types

Crafted types

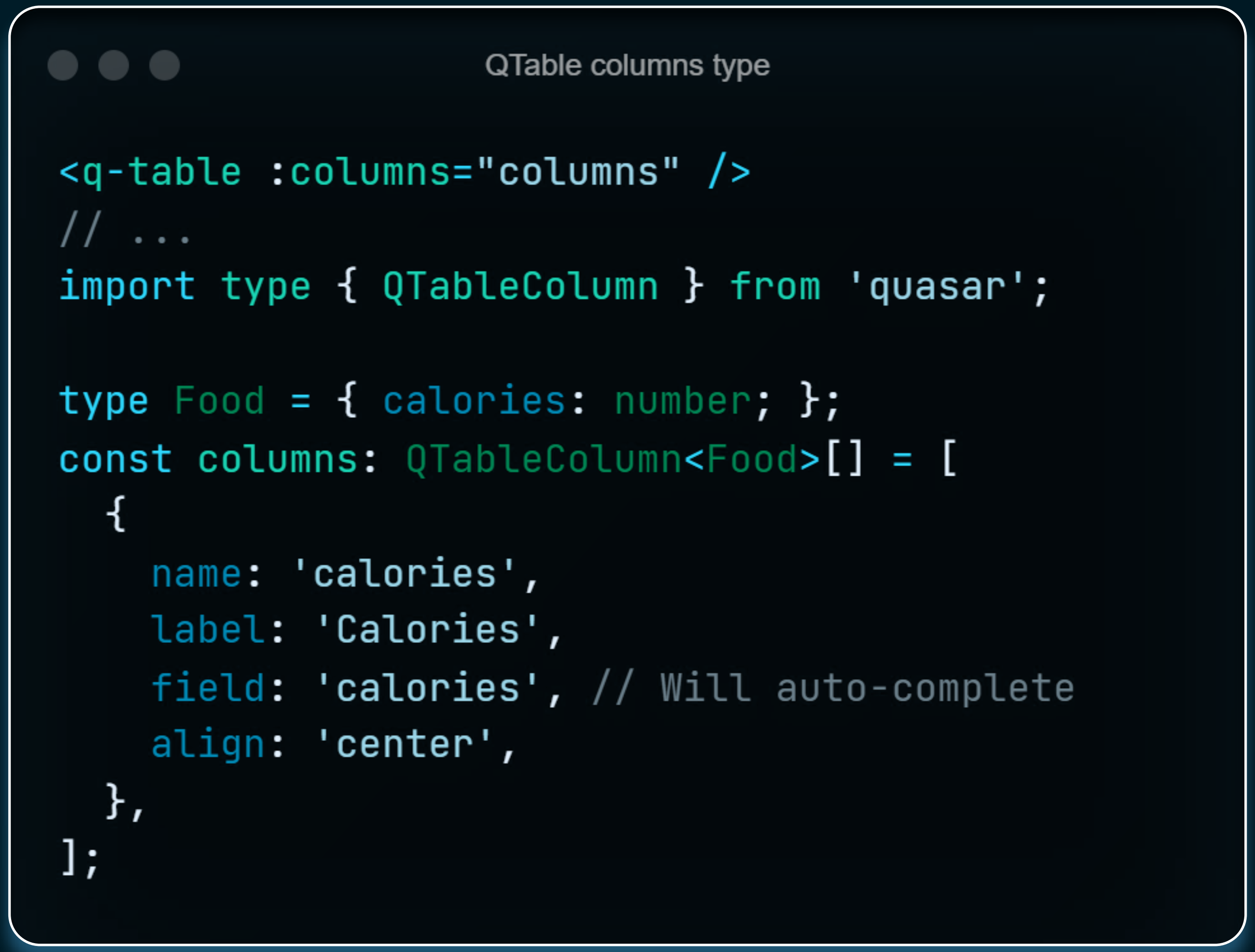
QSelect basic option type



QSelect basic option type

```
<q-select :options="options" />
// ...
import type { QSelectOption } from 'quasar';

const options: QSelectOption<number>[] = [
  {
    label: 'Quasar v2',
    value: 2,
  },
];
```

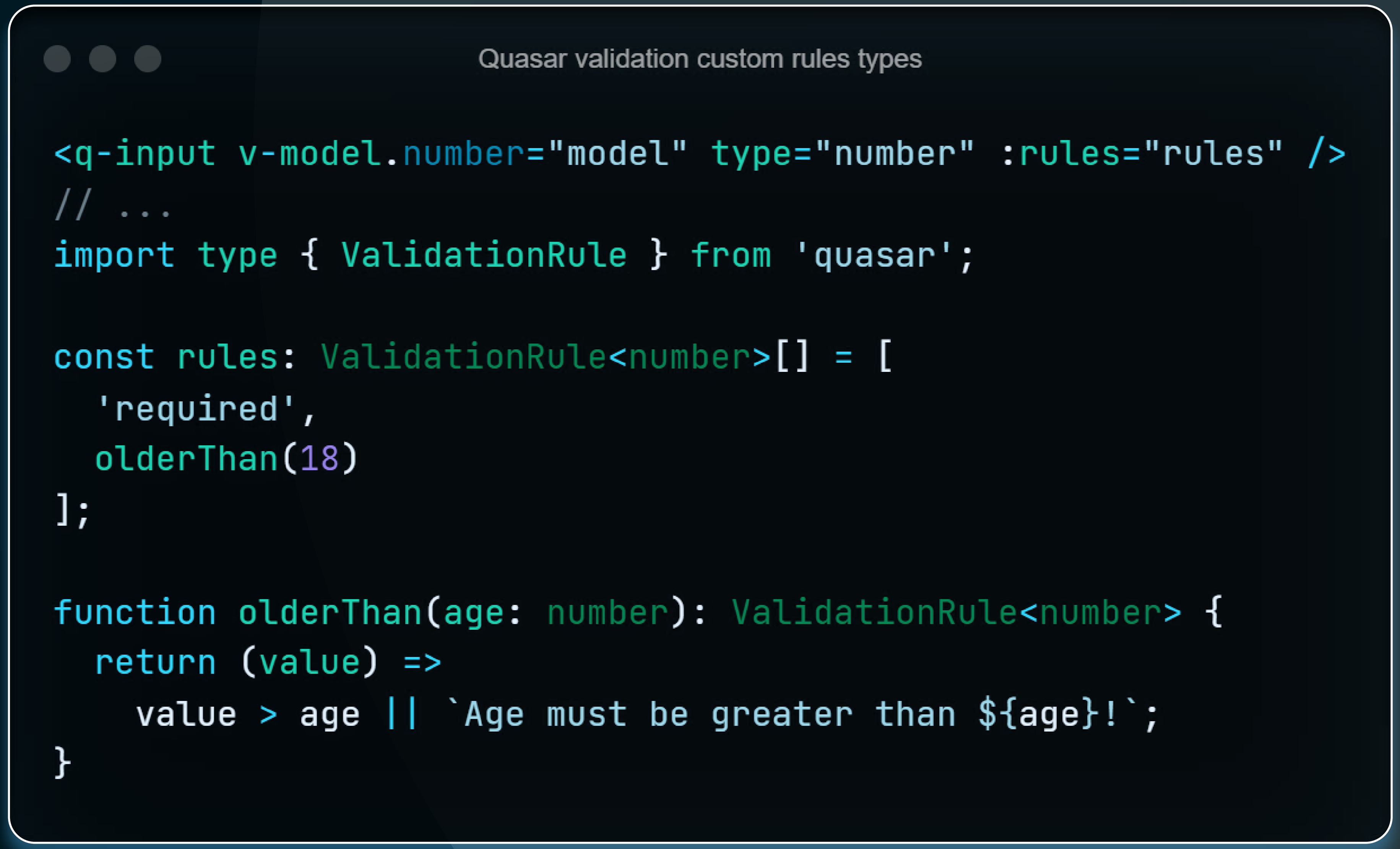
QTable columns types



QTable columns type

```
<q-table :columns="columns" />
// ...
import type { QTableColumn } from 'quasar';

type Food = { calories: number; };
const columns: QTableColumn<Food>[] = [
  {
    name: 'calories',
    label: 'Calories',
    field: 'calories', // Will auto-complete
    align: 'center',
  },
];
```

Quasar validation custom rules types

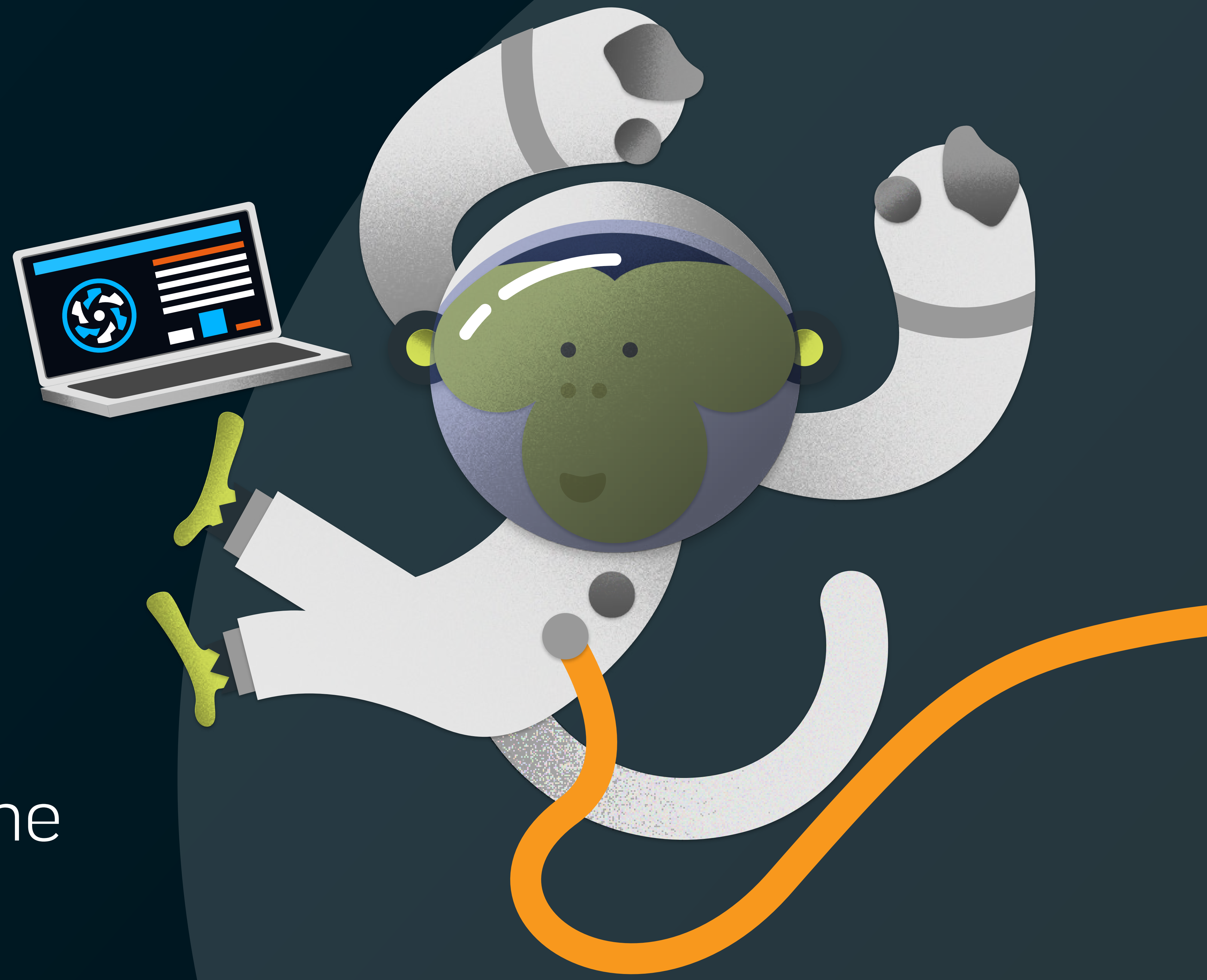


Quasar validation custom rules types

```
<q-input v-model.number="model" type="number" :rules="rules" />
// ...
import type { ValidationRule } from 'quasar';

const rules: ValidationRule<number>[] = [
  'required',
  olderThan(18)
];

function olderThan(age: number): ValidationRule<number> {
  return (value) =>
    value > age || `Age must be greater than ${age}!`;
}
```

DO YOU <3 QUASAR AS WE DO?

- Consider starting a monthly donation (https://donate.quasar.dev)
- If you like to help shaping Quasar’s future, contact me (https://chat.quasar.dev)
- If you like Dreamonkey as a workplace, submit an application (https://dreamonkey.com/en/work-with-us)



Dreamonkey

Evolution in motion

Dreamonkey Srl
P. IVA/CF 02722510357
via Mazzacurati 3B
42019 Scandiano [RE]
Italy

www.dreamonkey.com
info@dreamonkey.com
+39 348 663 8611

THANK YOU

Github: @yusufkandemir

Twitter: @yusuf1kandemir

Q & A