QUASAR

BEYOND THE FRAMEWORK

Tip & Tricks with Quasar

COMPONENT TESTING

Who I am

Paolo Caleffi

- Reggio Emilia, Italy
- Founder and CTO @ Dreamonkey
- Quasar core team member, automatic testing support

Github: @IlCallo

Twitter: @pcalloc

Quasar & Dreamonkey

- Working with (and on) Quasar since 2020
- Used Quasar in different contexts
  (static websites, food deliveries, finance, data processing, industrial IoT, office management, presentations, medical field)

Other Dreamonkey company insights

- Based in Italy
- Web apps and digital integrations for companies
- 1-day dedicated to internal R&D projects or open source
- Remote-friendly

Main topics

1. Component testing: what is it and when to use it
2. Caveats with Quasar components
3. Common patterns

1

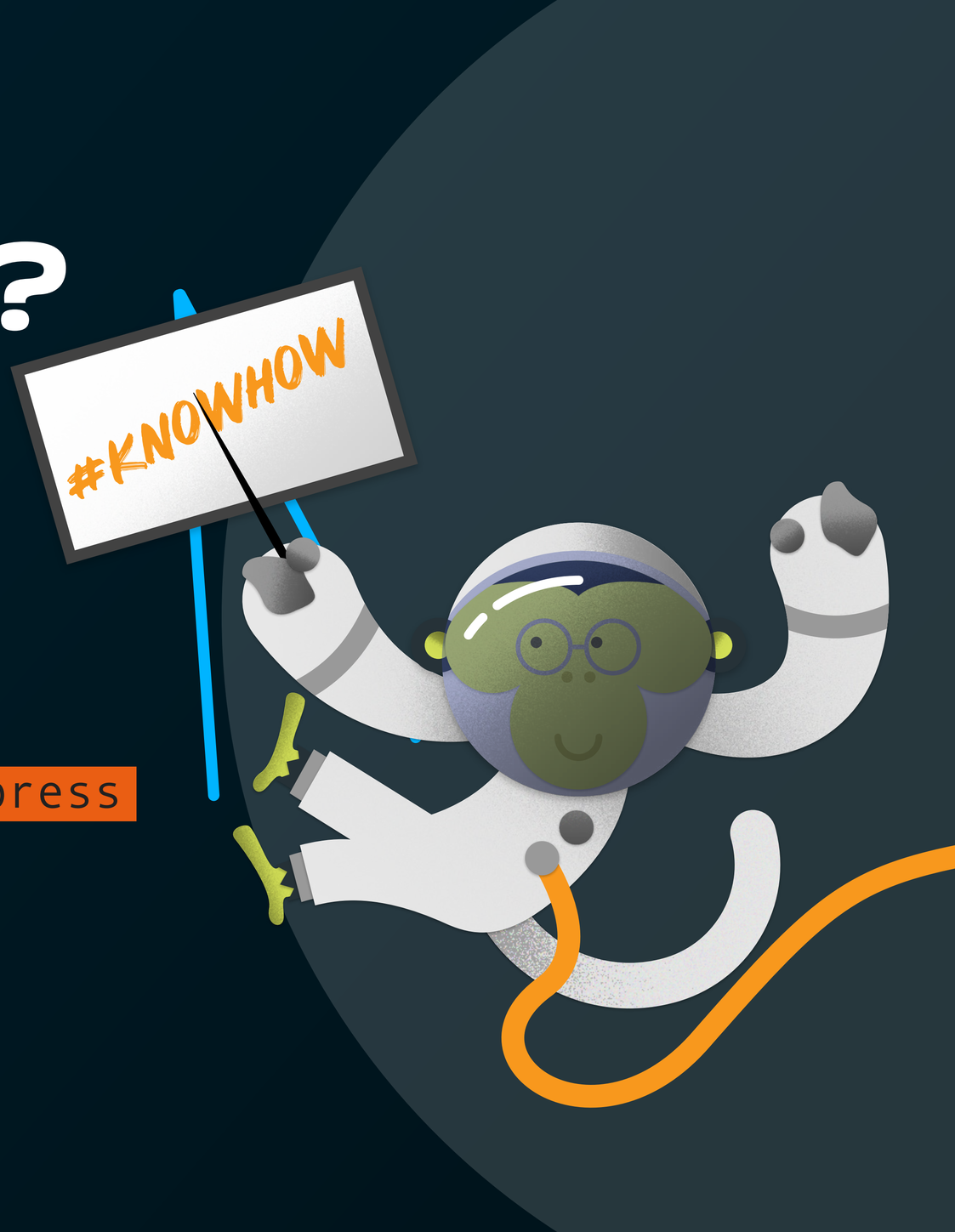
COMPONENT TESTING

What is it?

- Modeled after component-based architecture of modern frameworks
- Feature/integration test
- Test components in isolation (needs mocks for Router, Store, etc).
- Tackle many of Jest limitations

When to use it?

- Testing reusable, complex, but dumb components
- Component libraries
- `quasar ext add @quasar/testing-e2e-cypress`



2 CAVEATS WHILE USING QUASAR COMPONENTS

QSelect

QuasarSelect.vue

```vue
<template>
  <q-select
    data-cy="select"
    v-model="selected"
    label="test options selection"
    :options="options"
    :loading="loading"
    :disable="disable"
  />
</template>
```

QuasarSelect.spec.ts

```ts
cy.dataCy('select').click();
cy.withinSelectMenu(() => {
  cy.contains('Option 1').click();
});

// Becomes...

cy.dataCy('select').select('Option 1');
```

QuasarSelect.spec.ts

```ts
cy.dataCy('select').click();
cy.withinSelectMenu(() => {
  cy.get('.q-item').eq(1).click();
});

// Becomes...

cy.dataCy('select').select(1);
```

QuasarSelect.spec.ts

```ts
function dataCySelect(dataCyId: string) {
  return cy.dataCy(dataCyId).closest('.q-select');
}

// `cy.dataCy('select')` won't work in this case,
// as it won't get the root q-select element
dataCySelect('select').should('have.attr', 'aria-disabled', 'true');
```

QuasarSelect.spec.ts

```ts
// Wait for loading to complete
cy.dataCy('select').get('.q-spinner').should('not.exist');

cy.dataCy('select').select('Option 3');
```

.select( ) limitations

- Won’t yeld anything
- Won’t deselect already selected options (multiple QSelect)
- Cannot select by option value
- Ignores the original command options (eg. ‘force: true’)

QCheckbox/ QToggle/ QRadioButton

.check( ) limitations

- Won’t yeld anything
- Won’t accept parameters

QuasarCheckComponents.vue

```vue
<template>
  <q-checkbox data-cy="checkbox" v-model="checked" />
</template>
```

QuasarCheckComponents.spec.ts

```ts
cy.dataCy('checkbox').click();
cy.dataCy('checkbox').should('have.attr', 'aria-checked', 'true');

// Or...

function dataCyCheckbox(dataCyId: string) {
  return cy.dataCy(dataCyId).then(($quasarCheckbox) => {
    return cy.get('input:checkbox', {
      withinSubject: $quasarCheckbox,
    });
  });
}

dataCyCheckbox('checkbox-inside-form').check();
dataCyCheckbox('checkbox-inside-form').should('be.checked');

// Becomes...

cy.dataCy('checkbox').check();
cy.dataCy('checkbox').should('be.checked');
```

Dialog

Dialog.spec.ts

```
cy.withinDialog((el) => {
  cy.wrap(el).should('contain', message);
  cy.dataCy('ok-button').click();
});
```

Dialog.spec.ts

```
// The helper won't check for the dialog to be closed
// when the callback completes
cy.withinDialog({
  persistent: true,
  fn: (el) => {
    cy.wrap(el).should('contain', message);
  },
});
```

Dialog.spec.ts

```
// More than a dialog can be open on the screen at the same time
cy.dataCy('open-first-dialog').click();
cy.withinDialog({
  dataCy: 'my-first-dialog',
  fn: () => {
    cy.dataCy('open-second-dialog').click();
    cy.withinDialog({
      selector: '.cy-my-second-dialog',
      fn: () => {
        // ...
      },
    });
  },
});
```

Quasar plugins

DarkPlugin.vue

```vue
<template>
  <q-card data-cy="dark-card" :dark="$q.dark.isActive">
    {{ $q.dark.isActive ? 'Dark ' : 'Light' }} content
  </q-card>
</template>
```

DarkPlugin.spec.ts

```ts
cy.dataCy('dark-card')
  .should('not.have.class', 'q-dark')
  .then(() => {
    Dark.set(true);
  });

cy.dataCy('dark-card')
  .should('have.class', 'q-dark')
  .then(() => {
    Cypress.vueWrapper.vm.$q.dark.set(false);
  });

cy.dataCy('dark-card').should('not.have.class', 'q-dark');
```

3 COMMON PATTERNS

Interacting with Listeners

ComponentWithListener.spec.ts

```
const fn = cy.stub();

mount(ComponentWithListener, {
  props: {
    // This is how Vue internally codifies listeners,
    // defining a prop prepended with `on` and camelCased
    'onUpdate:modelValue': fn,
  },
});

cy.dataCy('button')
  .click()
  .then(() => {
    expect(fn).to.be.calledWith('some-value');
  });
```

Manage a reactive v-model

VModelComponent.vue

```vue
<template>
  <div>
    <span data-cy="model-value">{{ modelValue }}</span>
    <button
      data-cy="button"
      @click="$emit('update:modelValue', modelValue.substring(1))"
    >
      Remove first letter
    </button>
  </div>
</template>
```

VModelComponent.spec.ts

```ts
const text = 'Quasar';

mount(VModelComponent, {
  props: {
    modelValue: text,
    'onUpdate:modelValue': (emittedValue: string) =>
      Cypress.vueWrapper.setProps({ modelValue: emittedValue }),
  },
});

cy.dataCy('button').click();
cy.dataCy('model-value').should('contain', 'uasar');

// Becomes...

const model = ref('Quasar');

mount(VModelComponent, {
  props: {
    ...vModelAdapter(model),
  },
});

cy.dataCy('button').click();
cy.dataCy('model-value')
  .should('contain', 'uasar')
  .then(() => {
    // You cannot access `model.value` in a synchronous way,
    // you need to chain checks on it to a Cypress command
    // or you'll be testing the initial value.
    expect(model.value).to.equal('uasar');
  });
```

DO YOU <3 QUASAR AS WE DO?

- Consider starting a monthly donation
  (https://donate.quasar.dev)
- If you like Quasar AND testing, help us add tests to Quasar UI
  (https://github.com/quasarframework/quasar/tree/dev/ui/test/cypress)
- If you like Dreamonkey as a workplace, submit an application
  (https://dreamonkey.com/en/work-with-us)
- If you like Dreamonkey as a consulting firm, get in touch
  (https://dreamonkey.com/en/contacts/request-quotation)

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

SO LONG AND THANKS FOR ALL THE FISH

Github: @IlCallo

Twitter: @pcalloc

Q & A