VUE MASTERCLASS DAY 2

WELCOME

SCHEDULE

09:00 - Coffee + welcome

09:10 - Recap of day 1

Scaling your code base

09:15 - What do we mean by scaling?

09:25 - What are the challenges?

Vue CLI

09:35 - What is it and what does it do?

09:50 - Walkthrough

10:10 - Vue UI

10:20 - Coffee break

Code organisation

10:40 - Single file components

10:50 - Organising code by feature

11:00 - Exercise + solution

State management

11:20 - Why state management is hard

11:30 - Solution 1: Prop drilling

11:40 - Exercise + solution

12:00 - Lunch!

State management continued

12:45 - Composing components

12:55 - Global state using Vuex

13:10 - Exercise + solution

13:40 - Async actions

13:45 - Exercise + solution

14:10 - Coffee break

Smart and dumb components

14:30 - The problem

14:40 - Possible solutions

Routing

14:50 - HTML5 history api

15:00 - Vue-router

15:10 - Exercise + solution

Testing components

15:35 - Why

15:45 - Introducing Jest + vue-utils

15:55 - Do’s and don’ts when testing components

16:00 - Exercise + solution

Error handling

16:30 - Why you should handle errors

16:40 - Using error boundaries

16:50 - Exercise + solution

Round up

17:20 - Recap

RECAP OF DAY 1

WHAT WE LEARNED ON DAY 1

- Progressively introduce Vue in your application
- Vue manages the DOM state connection
- How reactivity works in Vue
- Props, state
- Transitions

we build a Slack app! 🎉

SCALING YOUR CODE BASE

SCALING?

- Multiple team members working on the same project
- Introducing code conventions and enforcing those code conventions
- Improved workflows
- Adding continuous integration
- Reliable tooling

FRONT-END CHALLENGES

- How to combine multiple source files?
- Preprocessors? How do you configure these preprocessors?
- What libraries do you use?
- How do you run build scripts? Webpack, rollup, gulp, grunt whatever.

GRUNT / GULP

```
var gulp = require('gulp');
var taskListing = require('gulp-task-listing');

// Get tasks from tasks directory
require('./tasks/assets');
require('./tasks/clean');
require('./tasks/compress');
require('./tasks/critical');
require('./tasks/demo');
require('./tasks/html');
require('./tasks/images');
require('./tasks/revision');
require('./tasks/scripts');
require('./tasks/styles');
require('./tasks/watch');
require('./tasks/rss');

gulp.task('default', taskListing.withFilters(null, 'default'));
```

VUE CLI

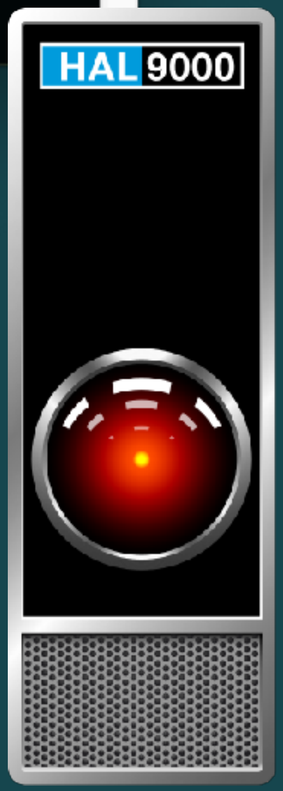
projects — -bash — 146×22

```
Usage: vue <command> [options]

Options:
  -V, --version                              output the version number
  -h, --help                                 output usage information

Commands:
  create [options] <app-name>                create a new project powered by vue-cli-service
  add [options] <plugin> [pluginOptions]     install a plugin and invoke its generator in an already created project
  invoke [options] <plugin> [pluginOptions]  invoke the generator of a plugin in an already created project
  inspect [options] [paths...]               inspect the webpack config in a project with vue-cli-service
  serve [options] [entry]                    serve a .js or .vue file in development mode with zero config
  build [options] [entry]                    build a .js or .vue file in production mode with zero config
  ui [options]                               start and open the vue-cli ui
  init [options] <template> <app-name>       generate a project from a remote template (legacy API, requires @vue/cli-init)
  config [options] [value]                   inspect and modify the config
  upgrade [semverLevel]                      upgrade vue cli service / plugins (default semverLevel: minor)
  info                                       print debugging information about your environment

  Run vue <command> --help for detailed usage of given command.

Remcos-MBP:projects remco$ vue create my-first-app
```

What can i do
for you?

projects — node ~/.nvm/versions/node/v10.15.0/bin/vue create sample-app — 110×16

```
Vue CLI v3.4.0

  Update available: 3.9.1

? Please pick a preset: Manually select features
? Check the features needed for your project: (Press <space> to select, <a> to toggle all, <i> to inve
rt selection)
>◉ Babel
 ◯ TypeScript
 ◯ Progressive Web App (PWA) Support
 ◯ Router
 ◯ Vuex
 ◯ CSS Pre-processors
 ◉ Linter / Formatter
 ◯ Unit Testing
 ◯ E2E Testing
```



THE RESULT

✅ Cross browser JS support

✅ Bundle your JS with one command

✅ Pre-configured support of popular front-end tooling

✅ Pre-configured testing frameworks

✅ Local development server with hot-reloading

serve Compiles and hot-reloads for development

Stop task | vue-cli-service serve | Output | Dashboard | Analyzer

Dashboard — Open app • Parsed

Status	Errors	Warnings
Success	0	0

Assets	Modules	Dependencies
2.0MB (Parsed)	1.8MB (Parsed)	1.7MB 92.26%

Idle (5s)

Speed stats

Global Average 2.28s	Mobile Edge 66.44s	2G 57.03s	3G Slow 39.76s	3G Basic 10.14s
3G Fast 9.99s	4G 1.92s	LTE 1.38s	Dial Up 315s	DSL 10.55s
Cable 3.18s	FIOS 0.79s			

Assets

	Parsed	Global	3G Slow	3G Fast	
app.js	**1.9MB**	**2.26s**	**39.36s**	**9.89s**	⚠
about.js	12.1kB	0.04s	0.64s	0.21s	
img/logo.82b9c7a5.png	6.7kB	0.04s	0.53s	0.18s	
favicon.ico	1.1kB	0.03s	0.42s	0.16s	

Dependencies

vue	553.8kB
sockjs-client	466.4kB
vue-router	173.4kB
core-js	142.9kB
html-entities	140.7kB

'Dev/vue-ui/scotch-vue:serve started

8/22/2018, 4:02:14 PM

🏋️ EXERCISE TIME! 🏋️

https://github.com/voorhoede/vue-masterclass-day-2

npm install

npm run serve

branch: exercise1

CODE ORGANISATION

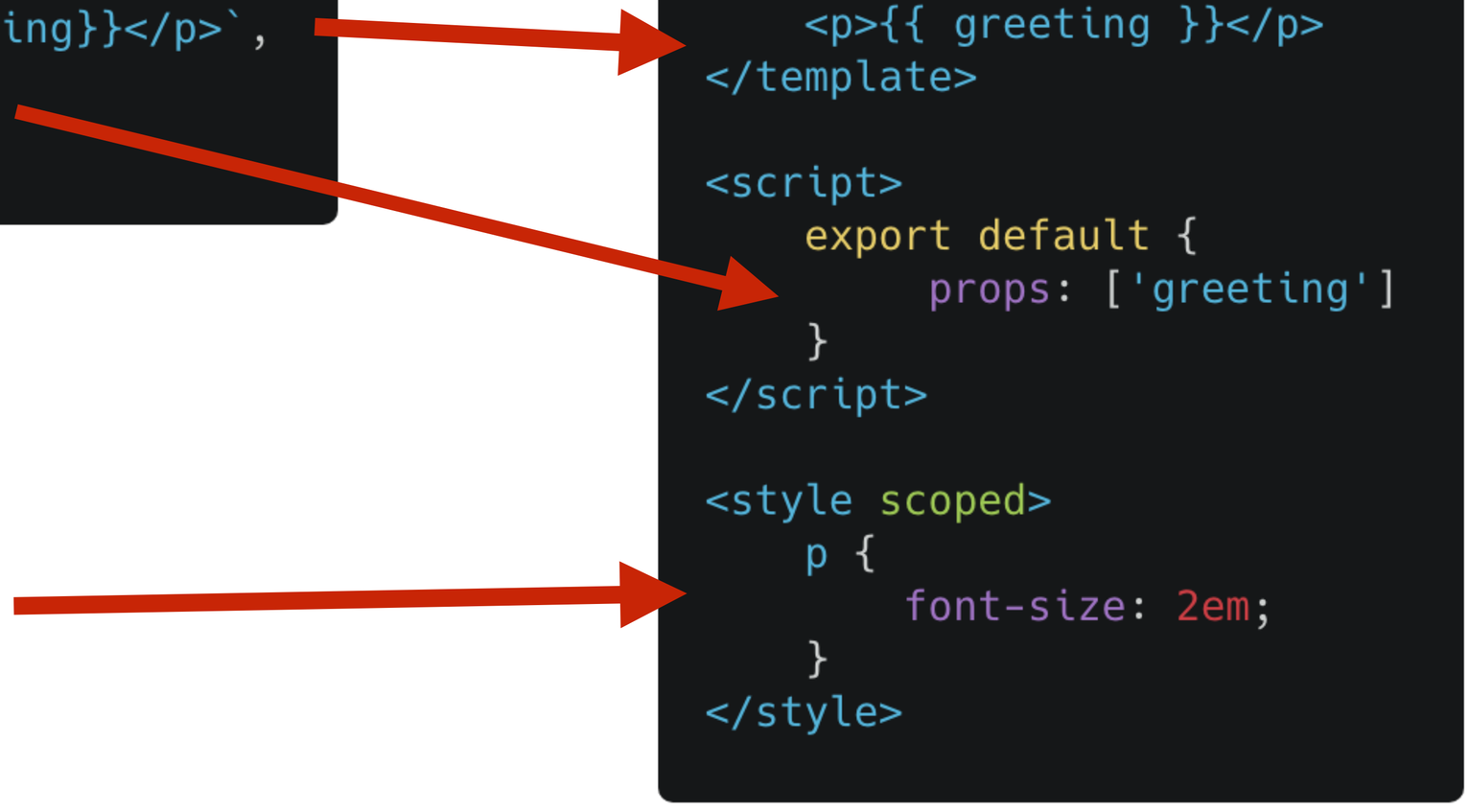
SINGLE FILE COMPONENTS

```
<template>
    <p>{{ greeting }}</p>
</template>

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

<style scoped>
    p {
        font-size: 2em;
    }
</style>
```

MIGRATING DAY 1 COMPONENTS

```
Vue.extend({
    template: `<p>{{greeting}}</p>`,
    props: ['greeting'],
});
```

```
.some-component p {
    font-size: 2em;
}
```



```
<template>
    <p>{{ greeting }}</p>
</template>

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

<style scoped>
    p {
        font-size: 2em;
    }
</style>
```

MIGRATING DAY 1 COMPONENTS

```vue
<template>
    <list></list>
</template>

<script>
  // import relative to current directory
  import List from '../list.vue';

  // or import relative to src directory
  import List from '@/components/list.vue';

  export default {
    components: {
        List,
    }
  }
</script>
```

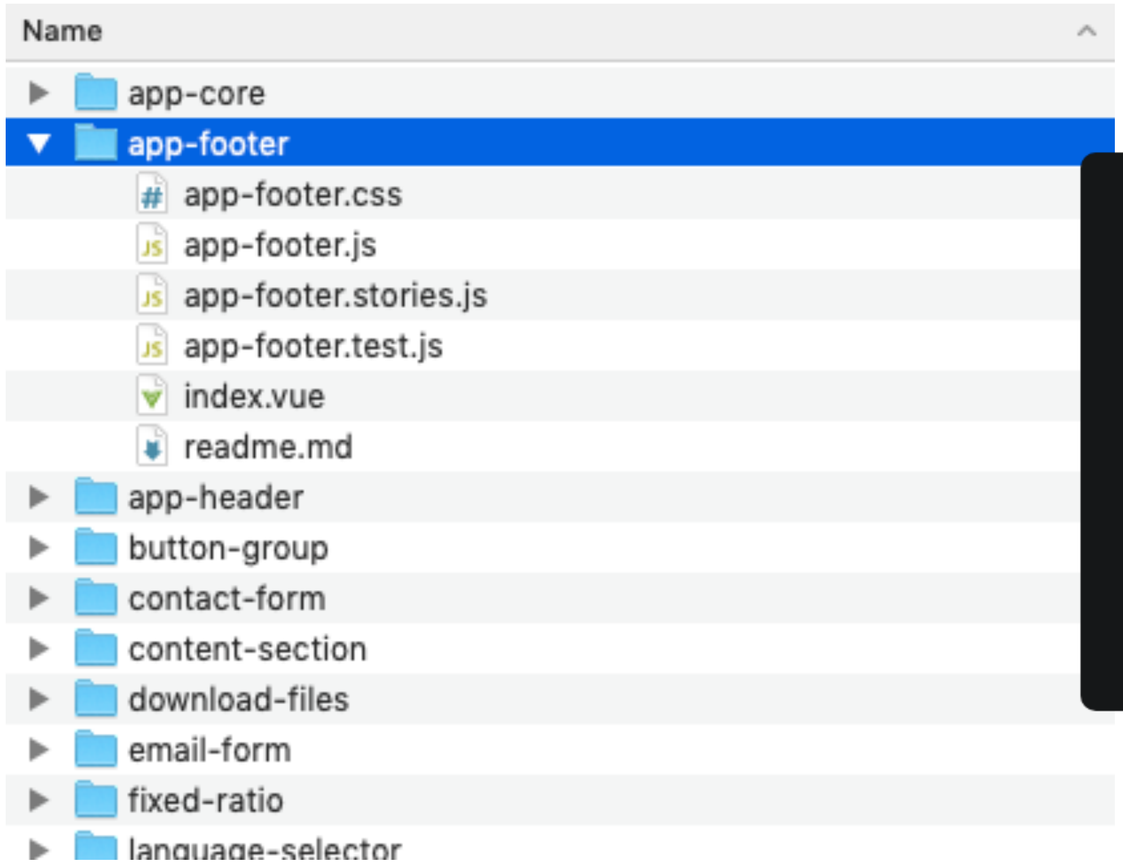
importing components is required!

SFC IN PRACTICE

243 lines (226 sloc) | 6.22 KB — Raw | Blame | History

```vue
<template>
  <div class="contact-form">
    <div v-if="title || contactPerson" class="contact-form__header">
      <h2
        v-if="title"
        class="contact-form__header h3"
      >
        {{ title }}
      </h2>
      <div v-if="contactPerson" class="contact-form__contact-person">
        <responsive-image :image="contactPerson.image"/>
        <dl>
          <dt class="sr-only">{{ $t('name') }}</dt>
          <dd class="h5">{{ contactPerson.name }} {{ contactPerson.lastName }}</dd>
          <template v-if="contactPerson.jobTitle">
            <dt class="sr-only">{{ $t('job_title') }}</dt>
            <dd class="body-petite">{{ contactPerson.jobTitle }}</dd>
          </template>
          <dt class="sr-only">{{ $t('email') }}</dt>
          <dd class="body-petite">
            <a
              class="link"
              href="mailto:post@voorhoede.nl"
            >post@voorhoede.nl</a>
          </dd>
        </dl>
      </div>
    </div>
    <form
      @submit.prevent="submit"
      method="POST"
      :name="form['form-name']"
      :action="localeUrl({ name: 'contact-slug', params: { slug: 'confirmation' } })"
      class="contact-form__form"
      data-netlify="true"
      netlify-honeypot="magic-castle"
      :novalidate="useCustomValidation"
    >
      <fieldset>
        <legend class="sr-only">{{ ariaLabelOrTitle }}</legend>
        <input type="hidden" name="form-name" :value="form['form-name']">
        <input type="text" name="subject" :value="form.name" class="hidden"/>
        <label class="hidden">
          Don't fill this out if you're human:
          <input v-model="form.magicCastle" name="magic-castle">
        </label>
```

COMPONENT FOLDER



index.vue

```vue
<template>
    <footer></footer>
</template>

<script src="./app-footer.js"></script>
<style src="./app-footer.css"></style>
```

organising folders by feature

VUE 3

```
<script>
export default {
    setup(props) {
        // State
        const todos = value([]);

        // Methods
        const addTodo = (todo) => todos.push(todo);

        // Computed
        const todoCount = computed(() => todos.length);

        return {
            todos,
            addTodo,
            todoCount,
        }
    }
}
</script>
```

```
<script>
function useTodos() {
    const todos = value([]);
    const addTodo = (todo) => todos.push(todo);
    const todoCount = computed(() => todos.length);
    return { todos, addTodo, todoCount };
}

export default {
    setup(props) {
        const todos = useTodos();

        return {
            ...todos,
            // other features
        }
    }
}
</script>
```

organising functionality by feature

🏋️ EXERCISE TIME! 🏋️

Create a emoji picker component

Requirements

- Multiple buttons containing emojis
- Clicking on a emoji should emit a '**pick**' event (use this.$emit)

branch: exercise1

🏋️ EXERCISE TIME! 🏋️

solution

branch: exercise1-solution

SCALING
STATE MANAGEMENT

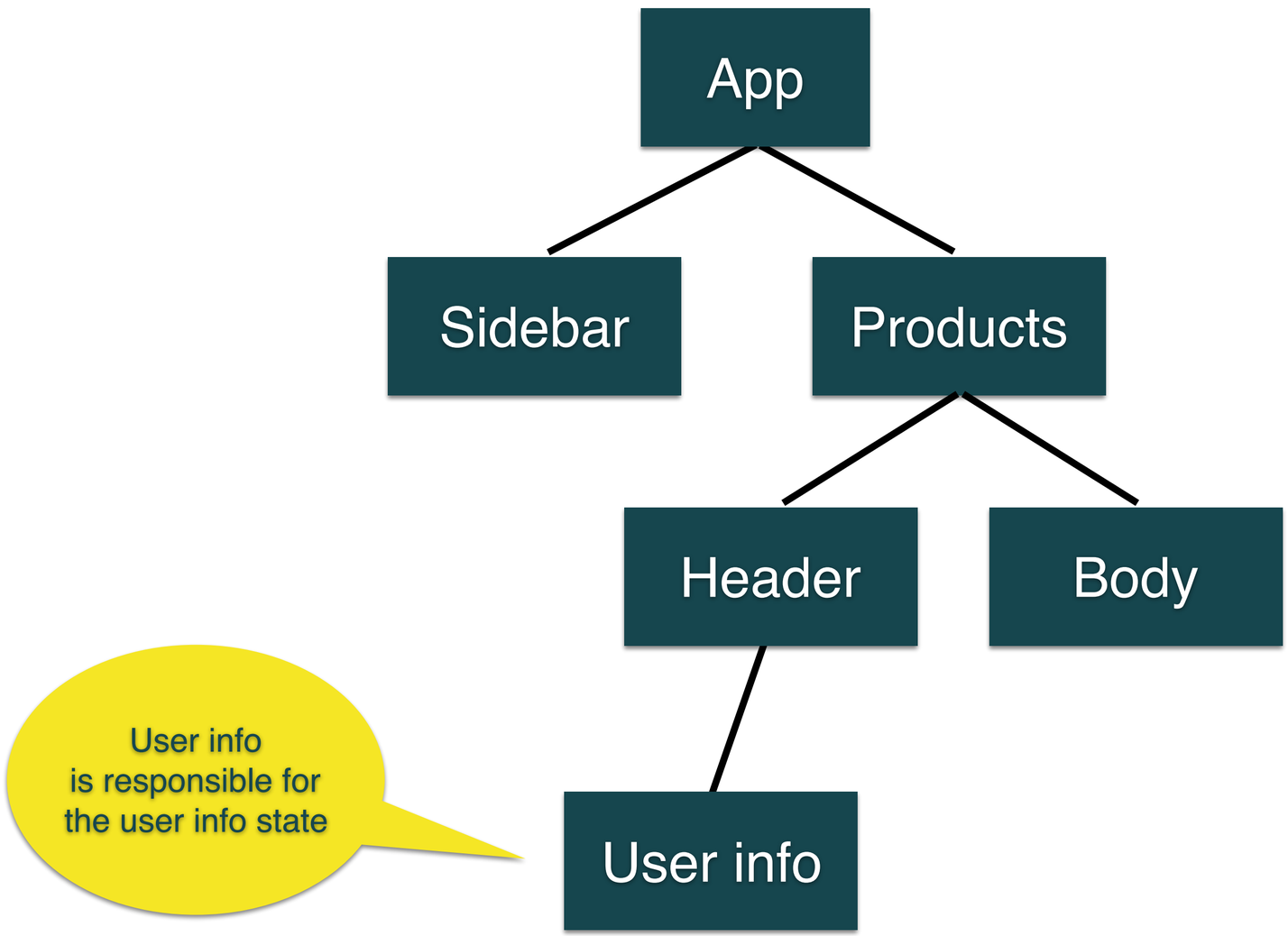
STATE MANAGEMENT

What **owns** the state?

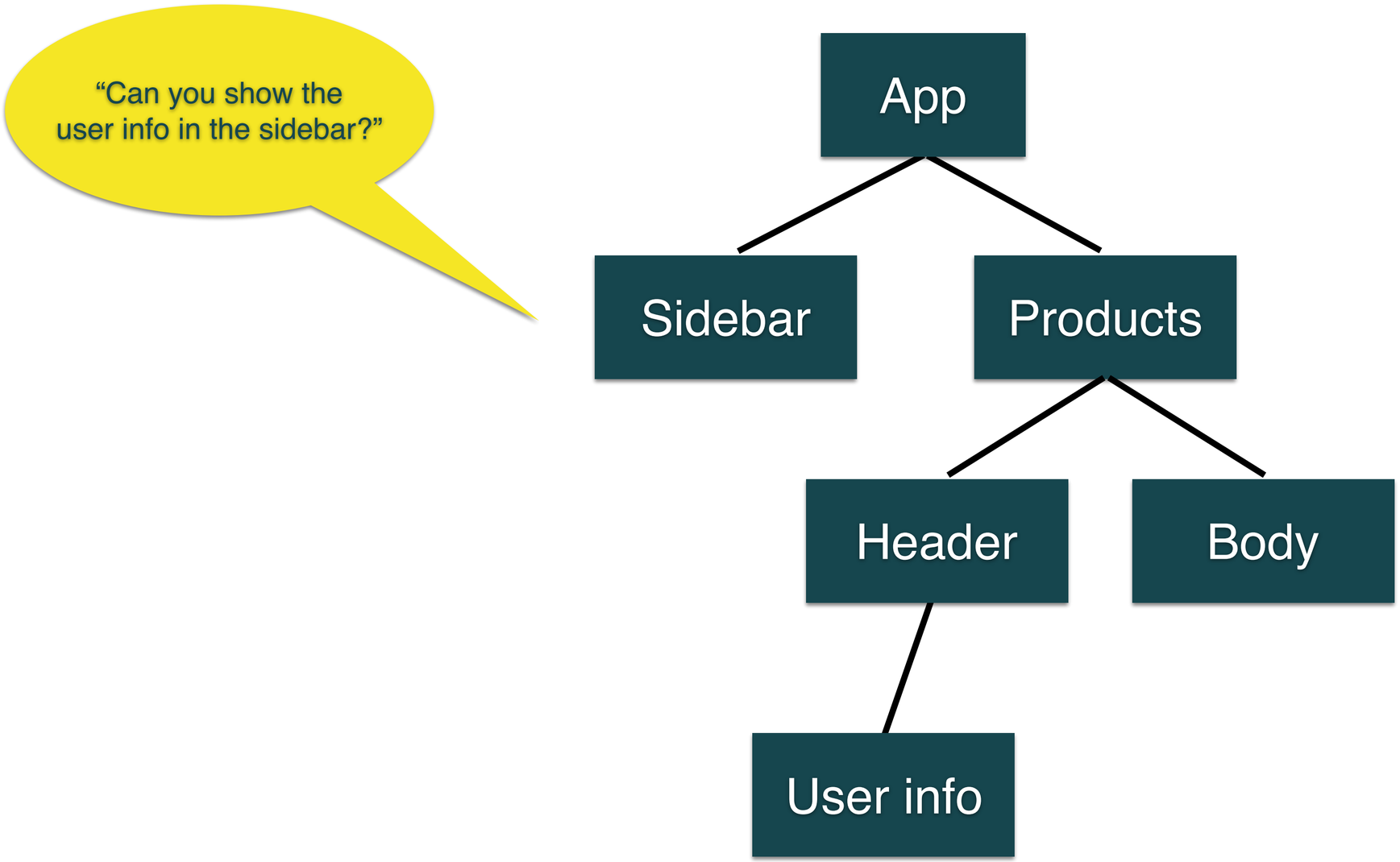
How do i **share** state with x?

FIGURING OUT THE BEST SOLUTION HAS A NEGATIVE IMPACT ON PRODUCTIVITY

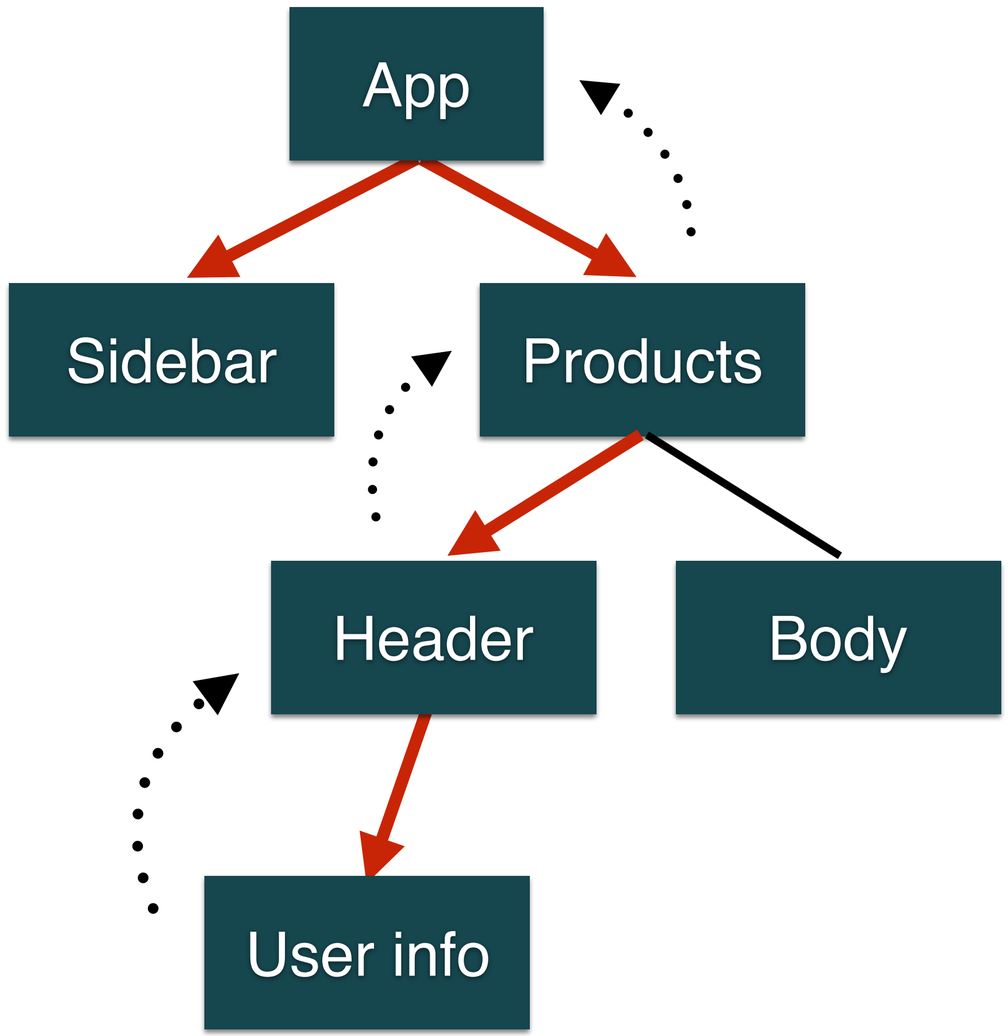
GETTING STATE FROM X TO Y



GETTING STATE FROM X TO Y

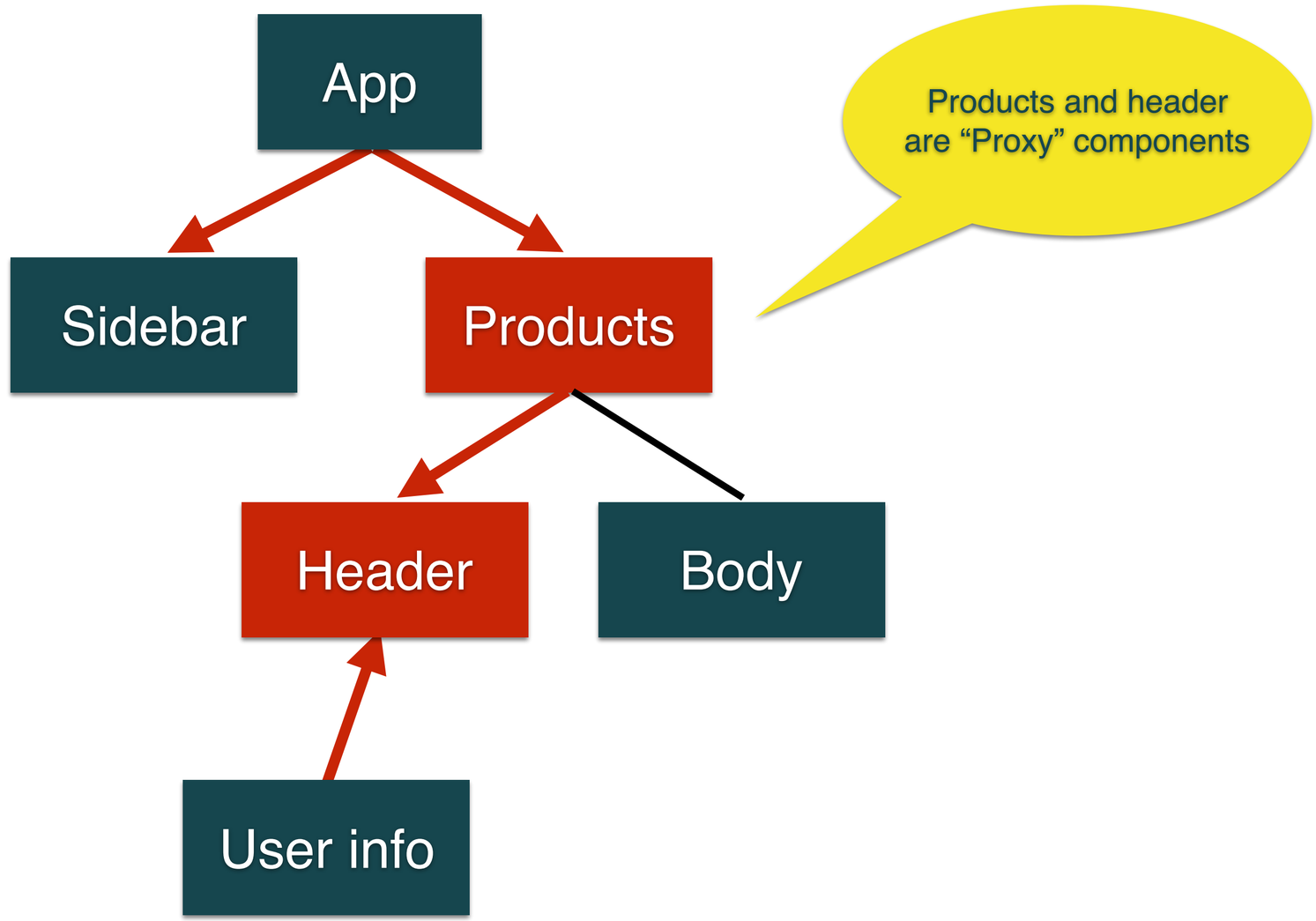


PROP DRILLING 



DON’T FORGET ABOUT EVENTS

PROXY COMPONENTS



🏋️ EXERCISE TIME! 🏋️

Use props to pass down messages

branch: exercise2

🏋️ EXERCISE TIME! 🏋️

solution

branch: exercise2-solution

COMPOSING COMPONENTS

App.vue

```vue
<main>
  <sidebar :userInfo="userInfo" />

  <products>
    <template v-slot:header>
        <user-info :info="userInfo" />
    </template>

    <template v-slot:body>
        Buy more products!
    </template>
  </products>
</main>
```

GLOBAL STATE

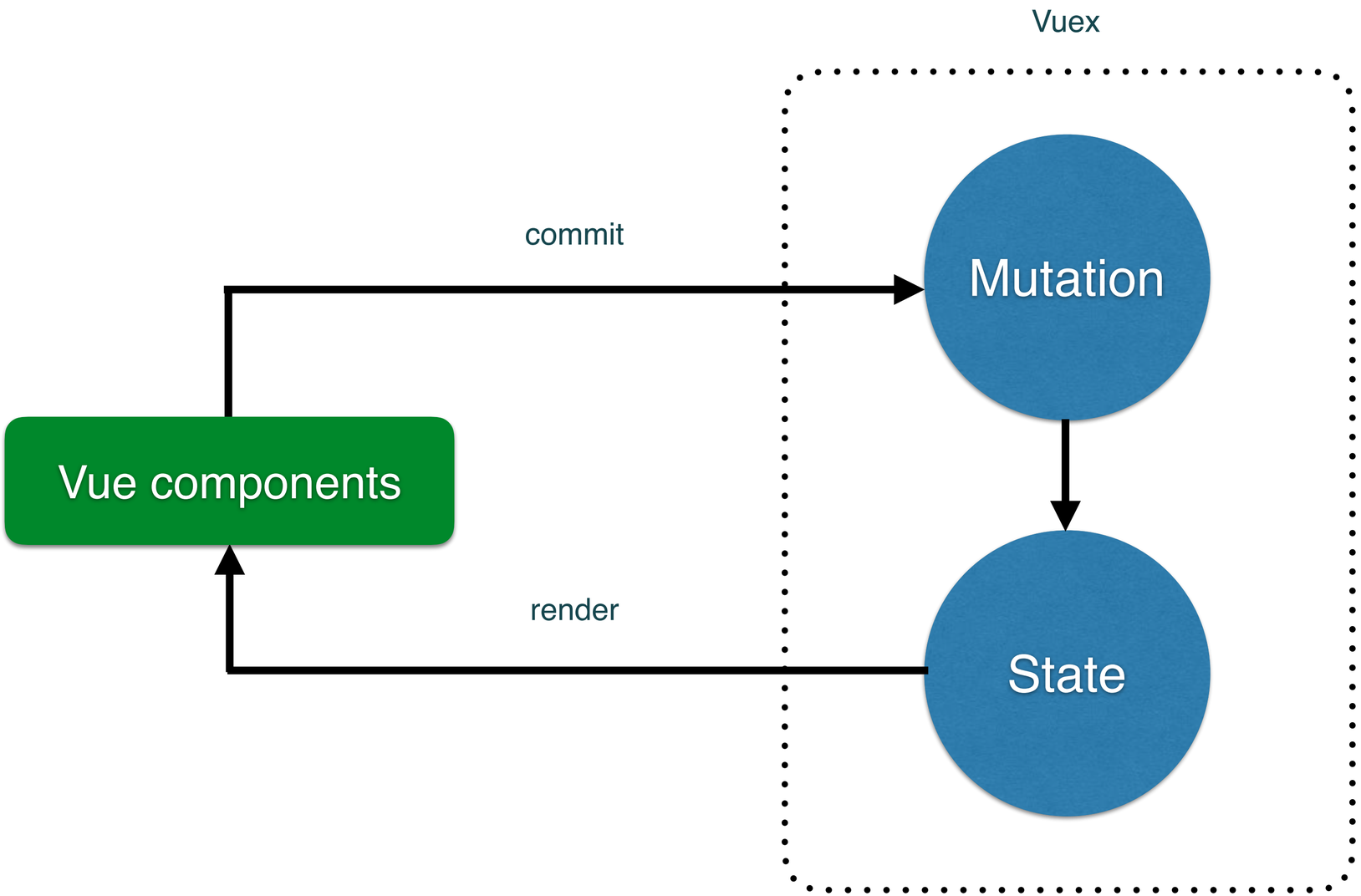
```
window.chat = {
    userId : null,
    users : [],
    channels : [],
    currentChannel : null,
    messages : [],
    searching : false,
    searchText : ''
}

window.app = new Vue({
    el: '.app',
    data : window.chat,
});
```

THE PROBLEMS OF GLOBAL STATE

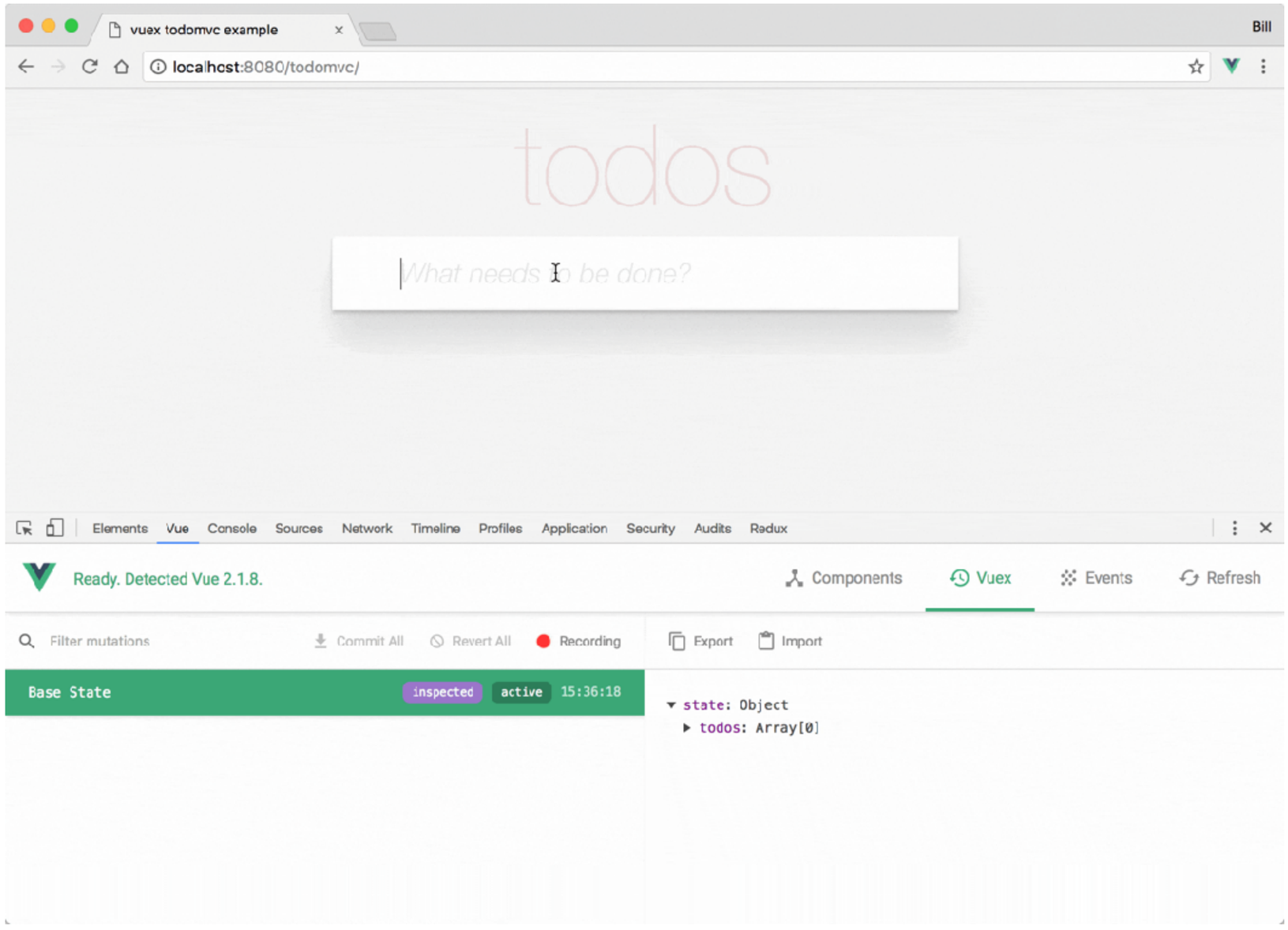
1. Abuse
2. Program state becomes unpredictable
3. Hard to debug (why did this state change?)

“Using global state with your fellow programmers is like using the same toothbrush with your friends - you can but you never know when someone will decide to shove it up his bum.”

INTRODUCING VUEX



DEVTOOLS



https://medium.com/vue-mastery/how-to-use-the-vue-devtools-af95191ff472

SETUP VUEX

```
const store = new Vuex.Store({
    state: {
        count: 0,
    },

    mutations: {
        // your mutations
    },

    // let vuex do extra checks to prevent
    // mutating state outside mutations
    strict: true,
});

// Install vuex plugin
Vue.use(Vuex);

// make your application vuex aware by
// injecting the store
new Vue({
    store,
});
```

🏋️ EXERCISE TIME! 🏋️

Add a vuex-store

branch: exercise3

🏋️ EXERCISE TIME! 🏋️

solution

branch: exercise3-solution

DOING YOUR FIRST MUTATION

```
methods: {
    addTodo(todo) {
        this.$store.commit('addTodo', todo);
    },

    // or

    ...mapMutations([
        'addTodo'
    ])
}
```

```
Mutation

addTodo(state, todo) {
    state.todos.push(todo)
}
```

READING OUT STATE

```
computed: {
    todos() {
        // or you can access $store.state.todos in your template
        return this.$store.state.todos;
    },

    // or

    ...mapState({
        todos: state => state.todos,
    })
}
```

DO’S AND DON’TS

Do’s

- Keep mutations pure
- State should be serializable to json
- State should be flat (or normalized like an database)

Don’ts

- Fetch data in mutations
- Save DOM elements in your state
- Structure data using nested objects

GETTERS

```
computed: {
    finishedTodos() {
        // or you can access $store.getters.finishedTodos in your template
        return this.$store.getters.finishedTodos;
    },

    // or

    ...mapGetters([
        'finishedTodos'
    ]),
}
```

```
getters: {
    finishedTodos(state) {
        return state.todos.filter(todo => todo.finished);
    }
}
```

WHAT WE LEARNED ABOUT VUEX

1. Vuex is a optional **Vue plugin**
2. Vuex makes global state **traceable** and **debuggable**
3. **Commits** trigger mutations
4. **Only mutations can modify state**
5. Getters are global computed properties

🏋️ EXERCISE TIME! 🏋️

Add mutations for messages and users

Connect state

branch: exercise4

🏋️ EXERCISE TIME! 🏋️

solution

branch: exercise4-solution

ACTIONS

```
actions: {
    addTodo({ state, commit, dispatch }, todo) {
        // you can do multiple mutations
        commit('addTodo', todo);
        commit('increaseTodoCount');

        // you can read but can't modify state
        state.todos.length = 0; // NOT ALLOWED
    }
}
```

TRIGGERING ACTIONS

```
methods: {
    addTodo(todo) {
        this.$store.dispatch('addTodo', todo);
    },

    // or

    ...mapActions([
        'addTodo'
    ]),
}
```

ACTIONS CAN BE ASYNC!

```
actions: {
    async addTodo({ commit }, todo) {
        // show a loader indicating that we are adding a todo
        commit('ADDING_TODO', true);

        await fetch('/add-todo', {
            method: 'POST',
            body: todo,
        });

        commit('APPEND_TODO', todo);
        commit('ADDING_TODO', false);
    }
}
```

🏋️ EXERCISE TIME! 🏋️

Connect the app to the api

branch: exercise5

🏋️ EXERCISE TIME! 🏋️

solution

branch: exercise5-solution

MODULES

```
// all the state and mutations concerning todos
const todos = {
    state: {},
    mutations: {},
}

// all the state and mutations concerning users
const users = {
    state: {},
    mutations: {},
}

new Vuex.Store({
    modules: {
        todos,
        users,
    }
});
```

SMART AND DUMB COMPONENTS

WHY COMPONENTS AGAIN?

- testability
- reliability
- reusability
- extensibility

WHAT WE BUILT SO FAR

…requires mocking a **global store**.

Which makes it hard(er) to **Test**
and less **reliable**

❌ Testability and reliability

WHAT WE BUILT SO FAR

…is directly connected to **global state**

Which makes it hard(er) to reuse in different places

❌ Reusable & Extensible

SPLIT

Components 💄

- How things **look**
- Gets data from **props**
- No global dependencies

Containers / Views 🛠️

- How things **work**
- Fetches data from **external sources**
- **Hard dependencies** on other app parts

1 - 1 RELATION

Header → **HeaderContainer**

1 - * RELATION

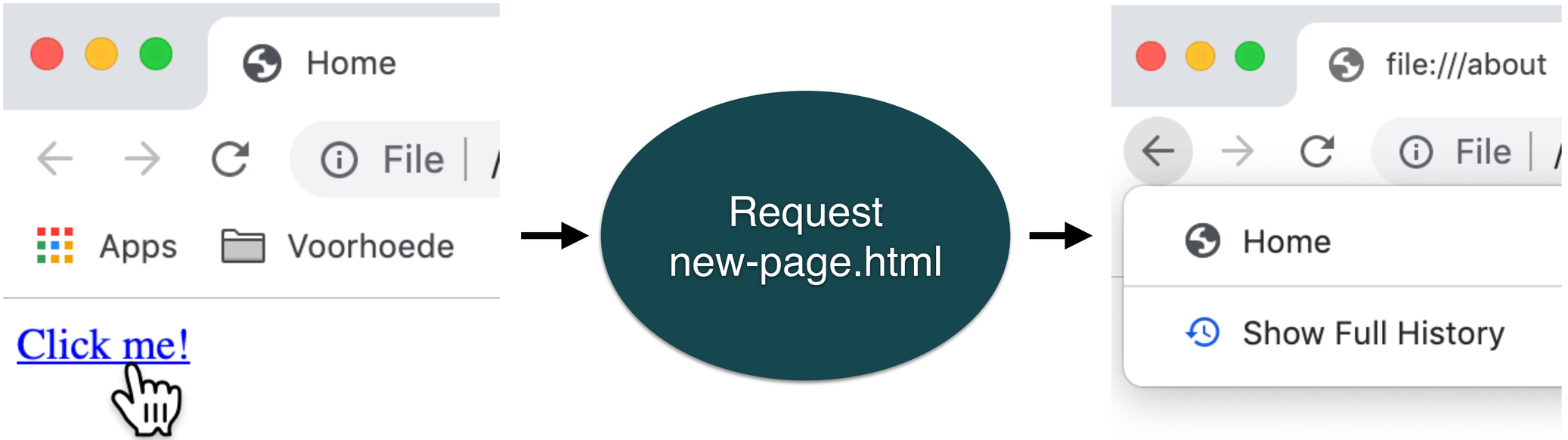
ProductPage.vue

```
<template>
  <page>

    <template v-slot:header>
        <header-container />
    </template>

    <template v-slot:sidebar>
        <sidebar-container />
    </template>

    <template v-slot:body>
        <product-list />
    </template>

  </page>
</template>
```

using multiple containers / components as part of a page

ROUTING

TRADITIONAL ROUTING

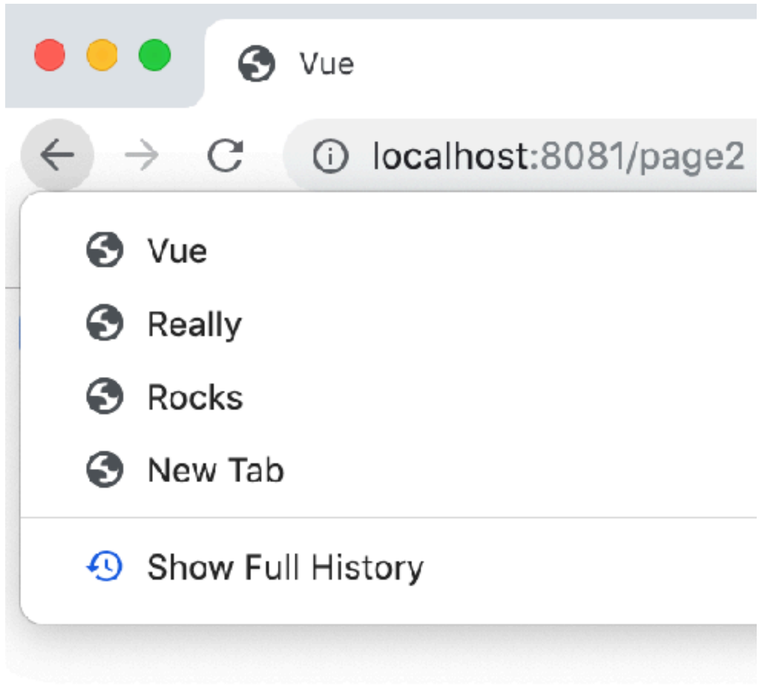


YOU DON’T WANT PAGE REFRESHES IN SPA’S

HTML5 HISTORY API

HTML5 history api

```
history.pushState({}, 'rocks', 'page1');
history.pushState({}, 'really', 'page2');
history.pushState({}, 'vue', 'page2');
```



ROUTING THE HARD WAY

```
// when clicking on a link (for example)
function navigateToUrl(url) {
    history.pushState({}, '', url);
    showPage(url);
}

// handle back-button
history.onpopstate = () => {
    showPage(document.location.pathname);
}
```

ROUTING THE EASY WAY

```
const router = new VueRouter({
    // define your "routes"
    routes: [
        {
            path: '/product/:id',
            component: ProductPage,
            name: 'product',
        },
    ]
});

// Install vue-router plugin
Vue.use(VueRouter);

// Make your app vue-router "aware"
new Vue({
    router,
});
```

LAZY ROUTES

```
{
    path: '/product/:id',

    // the import statement will make sure to load
    // the component on-demand
    component: () => import('./ProductPage.vue'),
}
```

✨ Generates a **separate js file** to be loaded on **demand** ✨

ROUTER-VIEW

```
<main>
    <router-view />
</main>
```

Show the component belonging
to the current route

NAMED VIEWS

```
[
  {
    path: '/products/:id',
    components: {
        header: ProductHeader,
        sidebar: ProductSidebar,
        main: ProductsPage,
    }
  }
]
```

```
<router-view name="header" />
<router-view name="sidebar" />
<router-view name="main" />
```

CHILD ROUTES

```
[
  {
    path: '/product',
    component: ProductPage,
    children: [
      // these routes are relative to the parent path
      {
        path: 'edit',
        component: ProductEditForm,
      }
    ]
  }
]
```

ProductPage

```
<form>
    <!-- ProductEditForm wil be injected here -->
    <router-view />
</form>
```

Great for **tabs** or **modals** which should have their own route!

ROUTER-LINK

```
<router-link to="/products">Products overview</router-link>
```

Using dynamic parts

```
<router-link :to="{ path: '/product', params: { id: 1 } }">Product 1</router-link>
```

ROUTER-LINK

❌

```
<router-link :to="{ path: '/products' }">Products</router-link>
```

✅

```
<router-link :to="{ name: 'Products' }">Products</router-link>
```

ACCESS ROUTER FROM COMPONENT

```
// get the id passed as the param to the current route
this.$route.params.id;

// programmatically change the url (without a router-link)
this.$router.push({ name: 'products' });
```

WHAT WE LEARNED ABOUT VUE-ROUTER

1. Vue-router is **config based**
2. It's best to make your route have **names**
3. **named views** are used to assign **multiple components** to a route
4. you can specify **child routes**
5. use router-link or $router.push to go to a route
6. use router-view to show the component belonging to the current route

Do you still need to handle routing on the server when using client side routing?

🏋️ EXERCISE TIME! 🏋️

Add a new route to change the channel

branch: exercise6

🏋️ EXERCISE TIME! 🏋️

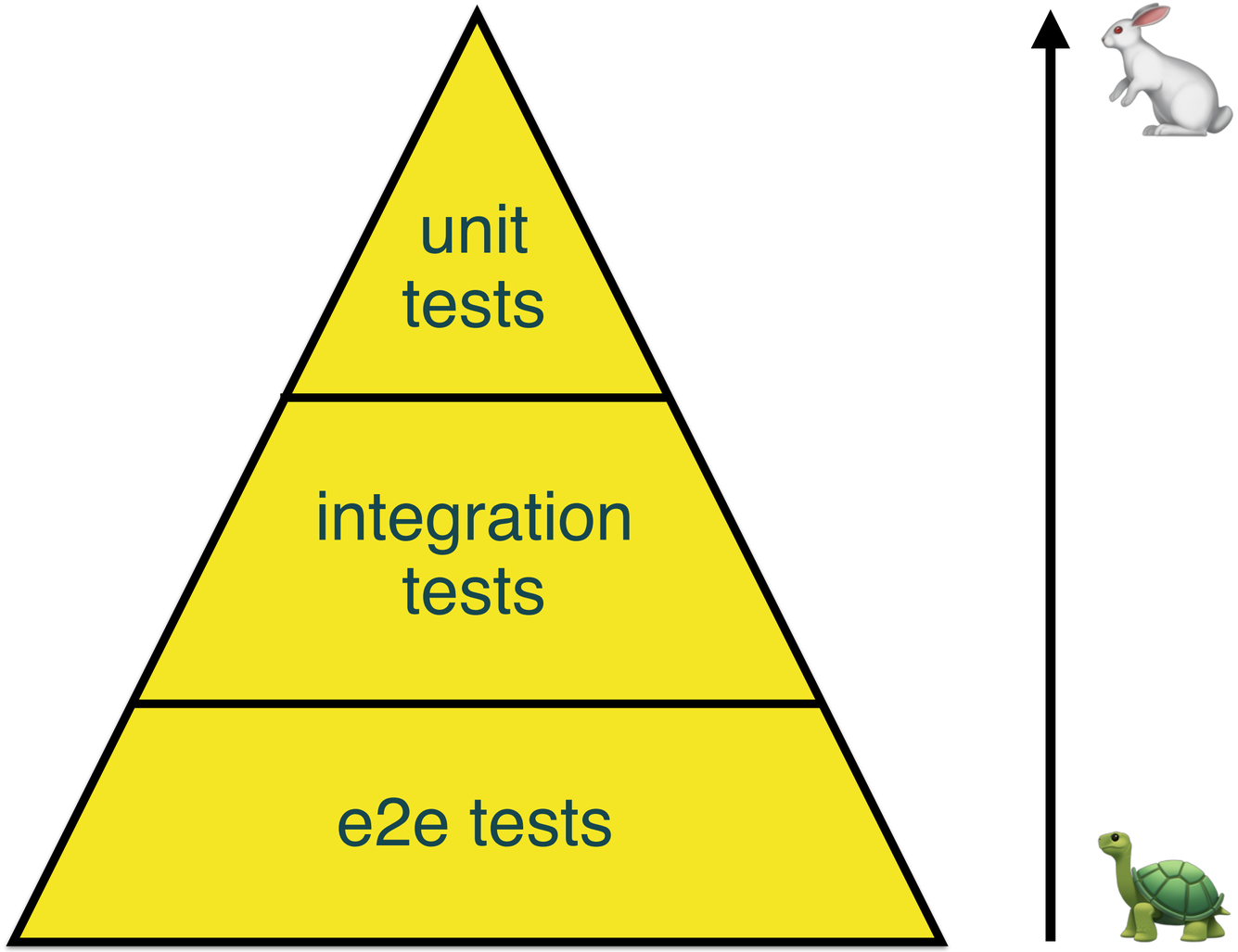
solution

branch: exercise6-solution

TESTING COMPONENTS

Tests hold component **requirements** as test cases and gives **trust** that the component works according to those requirements.

THE TESTING PYRAMID



THE STATE OF UNIT TESTING IN JS

Mocha

JSUnit

Tape

Ava

Sinon

Jest

Jasmine

Enzyme

Karma

Unit.js

Chai

Much frameworks, very overwhelming!

JEST

“Jest is a delightful JavaScript Testing Framework with a focus on simplicity”

- **CLI runner** which runs tests in a simulated browser env (jsdom)
- Easy to use **api**
- Ability to **mock / stub** out functions / modules
- Can generate **code coverage** results

Available as vue-cli plugin -> **@vue/cli-plugin-unit-jest**

JEST

```
// optionally describe your test suite
describe('Tests for component', () => {

    // give your tests a meaningful name
    it('expect 1+1 to be 2', () => {
        expect(1+1).toBe(2);
    });

});
```

```
PASS ./test.test.js
  Tests for component
    ✓ expect 1+1 to be 2 (10ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        1.366s
Ran all test suites.
```

https://jestjs.io/docs/en/expect

JEST WITH @VUE/TEST-UTILS

```
import { mount } from '@vue/test-utils';
import Component from 'component.vue';

it('Renders succesfully', () => {
    // mount the component so that we can query its content
    const wrapper = mount(Component);
    expect(wrapper.text()).toBe('succes!');
});
```

JEST WITH @VUE/TEST-UTILS

```
import { mount } from '@vue/test-utils';
import Component from 'component.vue';

it('renders succesfully', () => {
    // mount the component so that we can query its content
    const wrapper = mount(Component);

    // rerender the component with the given props
    wrapper.setProps({
        name: 'Test',
    });

    // click on a button inside the component
    wrapper.find('button').trigger('click');

    // test whether the component has emitted an event
    expect(wrapper.emitted.clicked).toBeTruthy();
});
```

DO TEST INPUT AND OUTPUT

```
import { mount } from '@vue/test-utils';
import Component from 'component.vue';

// 1. requirement
it('can not be clicked when disabled', () => {
    // 2. how a developer would use it
    const wrapper = mount(Component, {
        propsData: {
            enabled: false,
        }
    });

    // 3. how a end user uses your component
    wrapper.find('button').trigger('click');

    // 4. what you expect
    expect(wrapper.emitted.clicked).not.toBeTruthy();
});
```

DON’T TEST IMPLEMENTATION

```
import { mount } from '@vue/test-utils';
import Component from 'component.vue';

it('can not be clicked when disabled', () => {
    const wrapper = mount(Component);

    // set button disabled
    wrapper.setData({
        buttonDisabled: true,
    });

    // call the button handler
    wrapper.vm.buttonClickedHandler();

    expect(wrapper.emitted.clicked).not.toBeTruthy();
});
```

USER ORIENTATED SELECTORS

```
import { mount } from '@vue/test-utils';
import Component from 'component.vue';

it('should submit', () => {
    const wrapper = mount(Component);
    wrapper.find('form').trigger('submit');
    expect(wrapper.emitted().submit).toBe(true);
});

it('should be expanded', () => {
    expect(wrapper.find('form').attributes('aria-expanded')).toBe('true');
});

it('should show some text', () => {
    expect(wrapper.find('form').text()).toMatch('Register');
});
```

USER ORIENTATED SELECTORS

```
import { mount } from '@vue/test-utils';
import Component from 'component.vue';

it('should submit', () => {
    const wrapper = mount(Component);
    wrapper.find('form > button.sb-submit').trigger('submit');
    expect(wrapper.emitted().submit).toBe(true);
});

it('should be expanded', () => {
    expect(wrapper.find('form.expanded').exists()).toBe(true);
});

it('should show some text', () => {
    expect(wrapper.find('form > h1:first-child').text()).toBe('Register');
});
```

DATA-TESTID AS AN ALTERNATIVE

```
import { mount } from '@vue/test-utils';
import Component from 'component.vue';

it('should exist', () => {
    const wrapper = mount(Component);
    expect(wrapper.find("[data-testid='button']").exists()).toBe(true);
});
```

DATA-TESTID AS AN ALTERNATIVE

test-directive.js

```
/*
    <button v-test="test-button">Click me</button>
*/
export default {
  inserted(el, binding) {
    if(process.env.NODE_ENV === 'testing' || process.env.NODE_ENV === 'development') {
      el.setAttribute("data-testid", binding.arg);
    }
  }
}
```

MORE DOS AND DON'TS

Do

- **Predictable** output
- **Fast** running tests

Don't

- Unpredictable / random output
- Long running tests

🏋️ EXERCISE TIME! 🏋️

create a test for the message-field component

branch: exercise7

🏋️ EXERCISE TIME! 🏋️

solution

branch: exercise7-solution

ERROR HANDLING

TYPICAL ERRORS IN JS

Uncaught TypeError: Cannot read property

TypeError: ‘undefined’ is not a function

TypeError: null is not an object

TypeError: ‘undefined’ is not a object

TypeError: Cannot read property ‘length’

Uncaught RangeError: Maximum call stack

Uncaught TypeError: Cannot set property

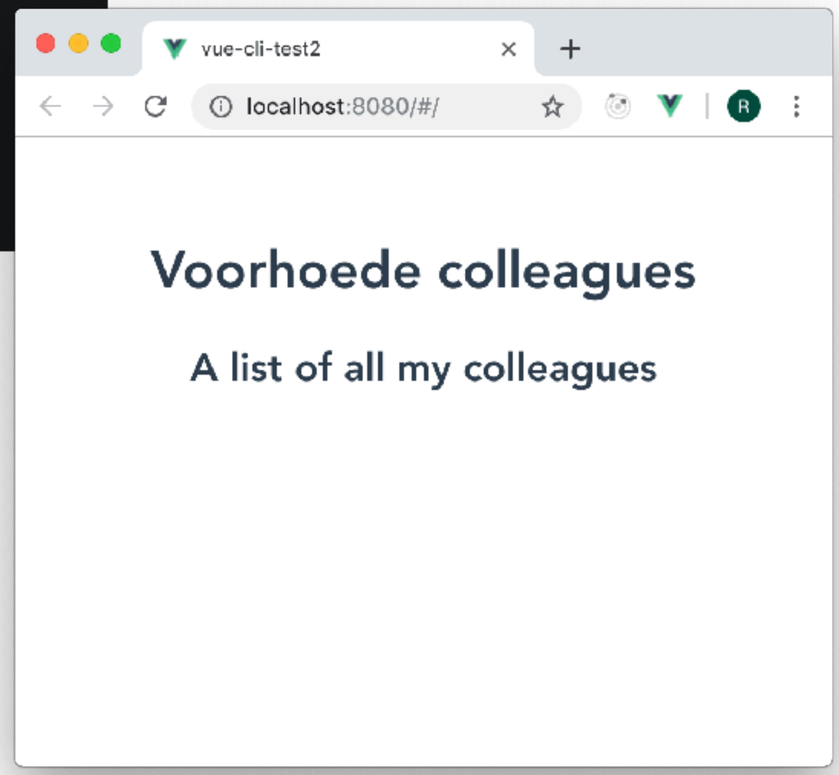
ReferenceError: event is not defined

https://rollbar.com/blog/top-10-javascript-errors/

THESE OCCUR BECAUSE

- Page refreshes are absent (no reset)
- More code runs in the client
- Bad connections or low memory devices
- JavaScript is a not so great language

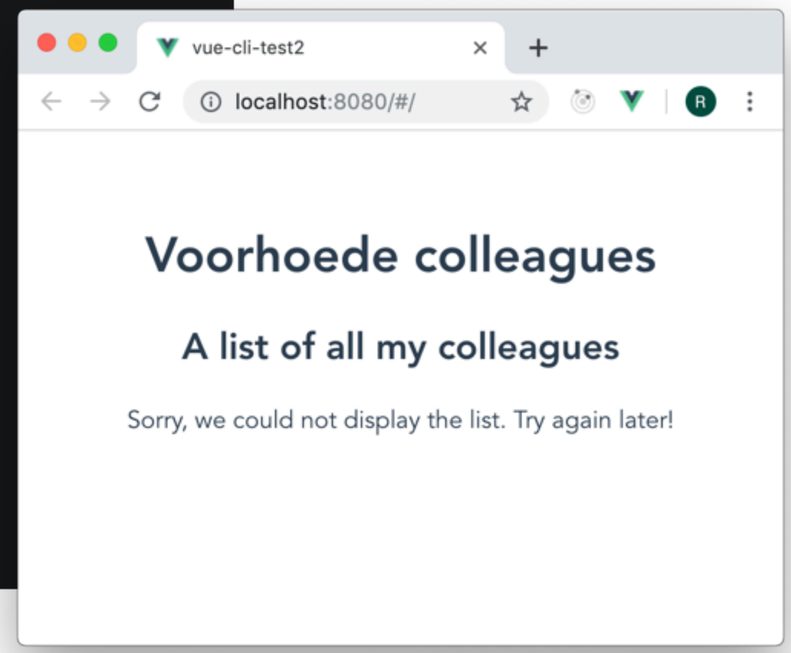
```vue
<template>
  <div id="app">
    <h1>Voorhoede colleagues</h1>

    <h2>A list of all my colleagues</h2>
    <ol>
      <li v-for="item of items" :key="item.id">
        {{ item.name }} is {{ item.info.age }} years old
      </li>
    </ol>
  </div>
</template>
```



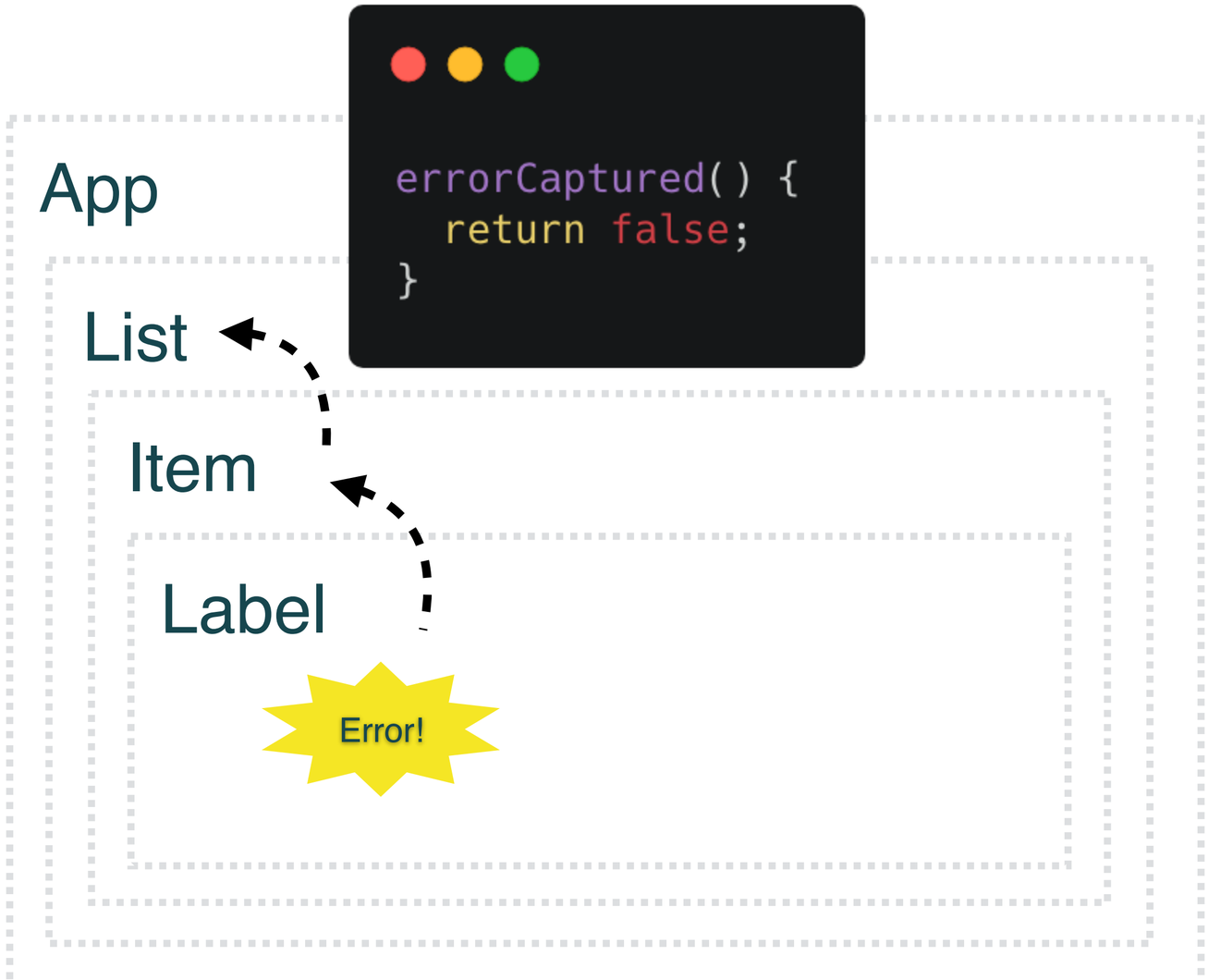
CATCH ERRORS

```
// 1 catch an error in a component
errorCaptured(error, vm, info) {
    // say sorry to the user :-(
}

// 2 catch global errors
Vue.config.errorHandler = (error) => {
    // log the errror?
}
```

```vue
<template>
  <div id="app">
    <h1>Voorhoede colleagues</h1>
    <h2>A list of all my colleagues</h2>

    <p v-if="hasError">Sorry, we could not display the list. Try again later!</p>
    <List v-else />
  </div>
</template>

<script>
  import List from '@/components/List.vue';

  export default {
    components: {
      List,
    },
    data() {
      return {
        hasError: false,
      }
    },
    errorCaptured() {
      this.hasError = true;
    },
  }
</script>
```



BUBBLING ERRORS



ERRORS IN EVENT HANDLERS

```
// errors in event handlers also trigger errorCaptured
<List @itemClicked="callUndefinedFunction()" />
```

Since vue 2.6 errors can be catched in event handlers

ERRORS IN EVENT HANDLERS

```vue
<template>
  <List @itemClicked="onItemClicked" />
</template>

<script>
  export default {
    methods: {
      onItemClicked() {
        return Promise.reject(
          new Error('Catching async errors!')
        );
      }
    }
  }
</script>
```

Also works for **async** errors!

ERROR-BOUNDARY

error-boundary.vue

```vue
<template>
  <div>
    <slot v-if="noError" />
    <p v-else>Something went wrong!</p>
  </div>
</template>

<script>
  export default {
    data() {
      return {
        noError: true,
      }
    },
    errorCaptured() {
      this.noError = false;
      return false;
    }
  }
</script>
```

A generic way to handle errors in your application

🏋️ EXERCISE TIME! 🏋️

Catch the 2 errors in the application and display a user friendly message.

branch: exercise8

🏋️ EXERCISE TIME! 🏋️

solution

branch: exercise8-solution

RECAP

WHAT WE LEARNED

1. You understand what the purpose of **vue-cli** is
2. You know different patterns for **structuring your state**
3. you know what it takes to make **testable components**
4. you know how to **test your components**
5. you know all about **client side routing**
6. you know why you should think about **error handling** and how to approach it

GO FORTH AND MAKE AWESOME APPLICATIONS!

Documentations

- https://jestjs.io
- https://nuxtjs.org/
- https://vue-test-utils.vuejs.org
- https://vuejs.org
- https://www.cypress.io/
- https://vuex.vuejs.org/guide/
- https://router.vuejs.org/

Made with Vue by the Voorhoede

- https://pathe-thuis.nl
- https://www.geldmaat.nl/
- https://startmail.com
- https://bouwenismacht.versbeton.nl
- https://pickup10.org
- https://www.worldwateratlas.org
- https://quantum-inspire.com

DE VOORHOEDE

front-end developers