Vue.js

Introduction by Lateral Nord

Lateral Nord

Jerry Jäppinen

Product design consultant

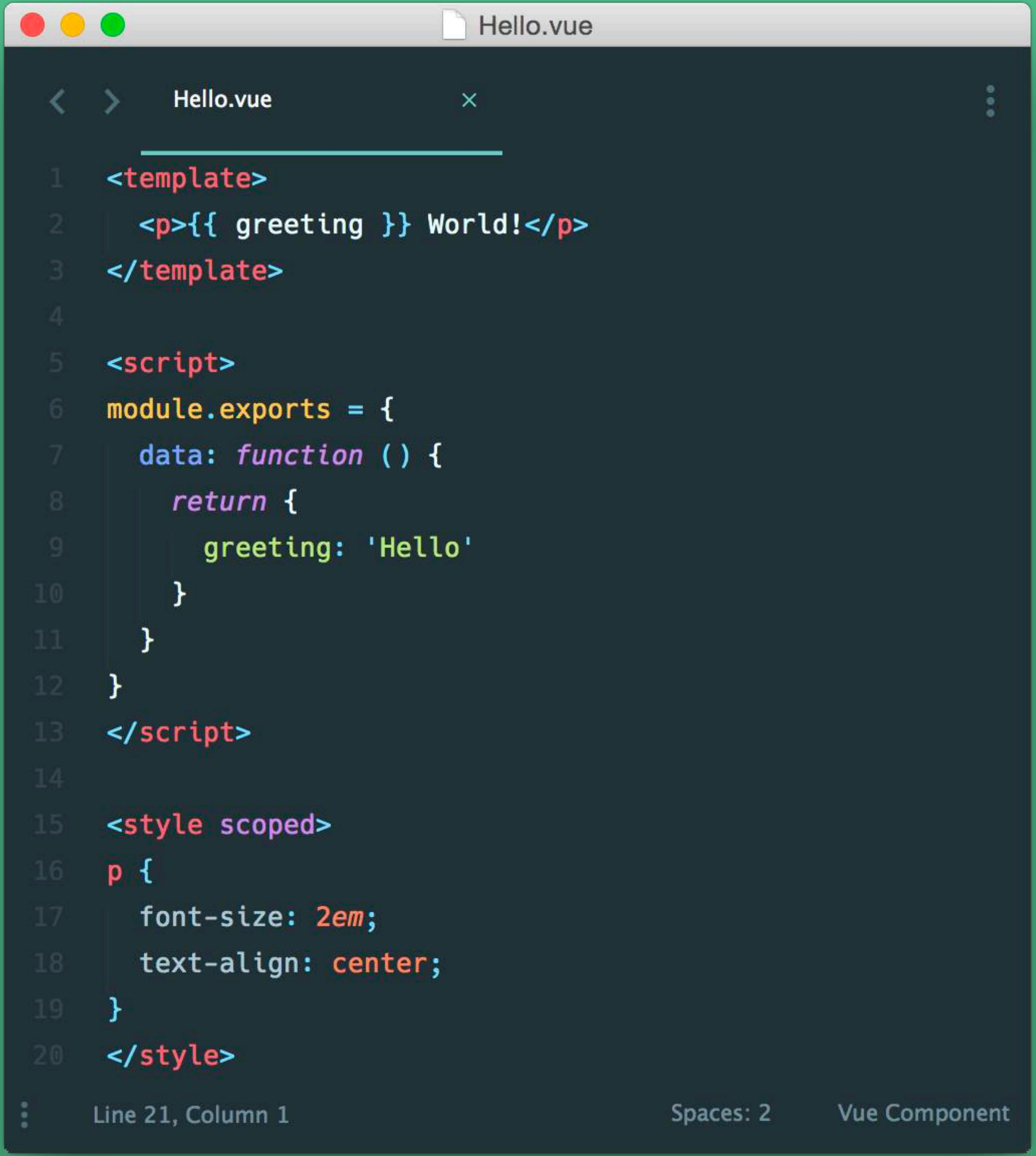
DCMN elisa visable OLX ProductBeat

jerryjappinen@lateralnord.com
+358 40 7188776
@jerryjappinen

Today's programme

Platform introduction: 30 min

Live coding: 30 min



Hello.vue

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

<script>
module.exports = {
  data: function () {
    return {
      greeting: 'Hello'
    }
  }
}
</script>

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

Line 21, Column 1   Spaces: 2   Vue Component

What is Vue.js?

A modern JS framework

that is progressive

and approachable

and versatile

and performant

Core vs platform

Vue is similar to React

Vue is more complete

Vue.js core: reactivity, change detection, event handling, templating, data-binding etc.

Rich ecosystem and development experience around the core

Extremely flexible

Can be used via a CDN on a static website without tooling

Supports both templating and render functions

Pluggable templating (JSX 🙄 support available)

Works on both server and browser

Custom libraries can easily be injected into Vue instances

Vue is extremely well architectured, extensible and progressive!

…but approachable

Users can get started with just a a static HTML page and get the JS file from a CDN

Strongly suggested canonical solutions with high level of abstraction exist

…but everything is pluggable and layered

Style guide is thought-out, cascading rules and suggestions

Official guide

Excellent guide:

https://vuejs.org/v2/guide/

(Much better than this presentation)

A word on completeness

A word on completeness

I’ve observed Vue to have a certain vibe and philosophy of completeness and orientation towards practicality.

Vue and its supporting libraries should work in real-life scenarios.

They should be easy to learn and understand.

If people write bad code with good libraries, then maybe the libraries could be better.

If you provide a solution for components with no word on how to implement styling, it’s not a solution for components.

This doesn’t mean that I haven’t had issues closed with “doesn’t belong in core” - the core team can still be strict about what to accept. But they have also rolled back changes that users found were not benefitting their practical development experience.

Common misconception: Vue.js is only for simple projects

Vue.js today is extremely flexible and powerful

has a mature ecosystem

and a large, enthusiastic and talented community

Example: Vue.js Server-Side Rendering Guide

End-to-end guide to the topic of SSR with Vue

https://ssr.vuejs.org/en/

Background

Background

1.0 out in 2014, 2.0 out in 2016

2.5 out 13 Oct 2017

Main developer Evan You (Google background), X core devs

Very community-driven, Patreon-funded

Major backers:

Popularity over time

Vue.js is growing, growing, growing...

Release schedule

Satisfying release cycle

Been more stable lately

Roadmap

Core and API is very stable

Currently most work goes towards the ecosystem

https://github.com/vuejs/roadmap

Evan You

State of Vue.js 2018

https://www.youtube.com/watch?v=TRJMT9yjONQ

Let’s dive in

https://vuejs.org/v2/guide/

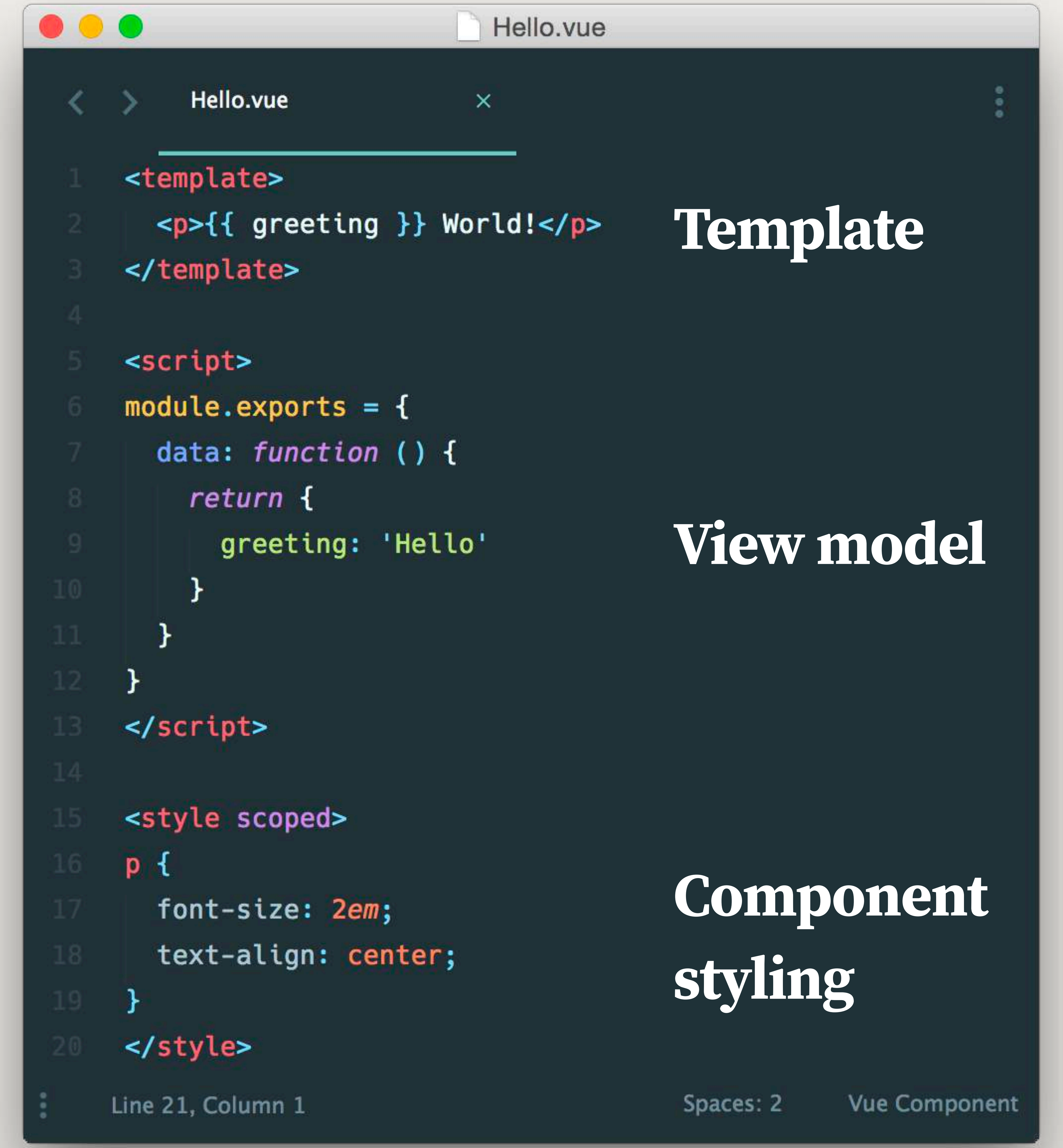
Structure of Vue apps

Official webpack template: Jest + Nightwatch

Official test-utils library available

Not much different from other JS frameworks

Single-file components



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

<script>
module.exports = {
  data: function () {
    return {
      greeting: 'Hello'
    }
  }
}
</script>

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

Template

View model

Component styling

Single-file components

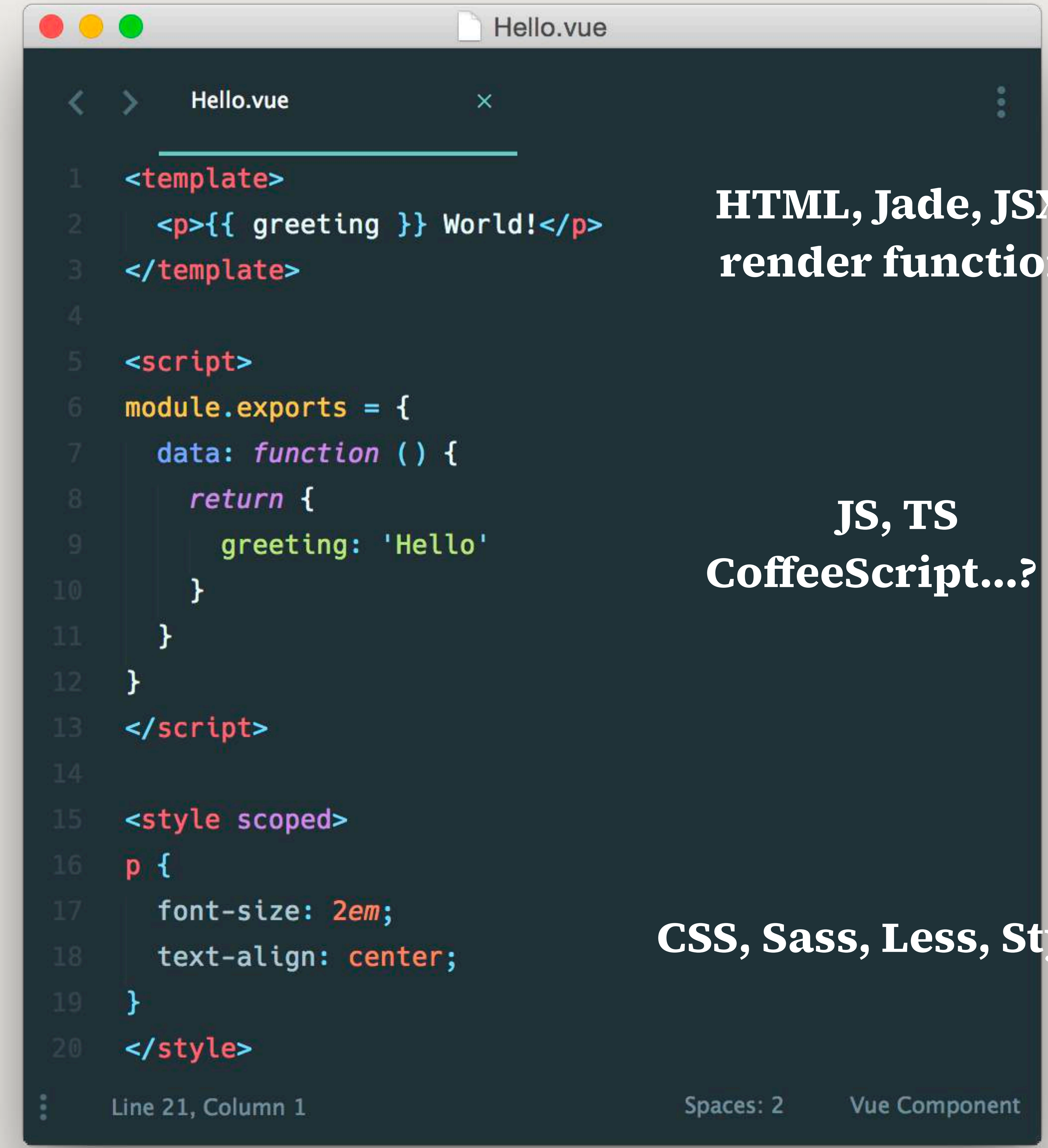
Canonical solution for authoring components in regular application development

Combines VM, template and styling into one file

Leverages existing web technologies

https://vuejs.org/v2/guide/single-file-components.html

Great multi-language support



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

<script>
module.exports = {
  data: function () {
    return {
      greeting: 'Hello'
    }
  }
}
</script>

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

HTML, Jade, JSX, render function

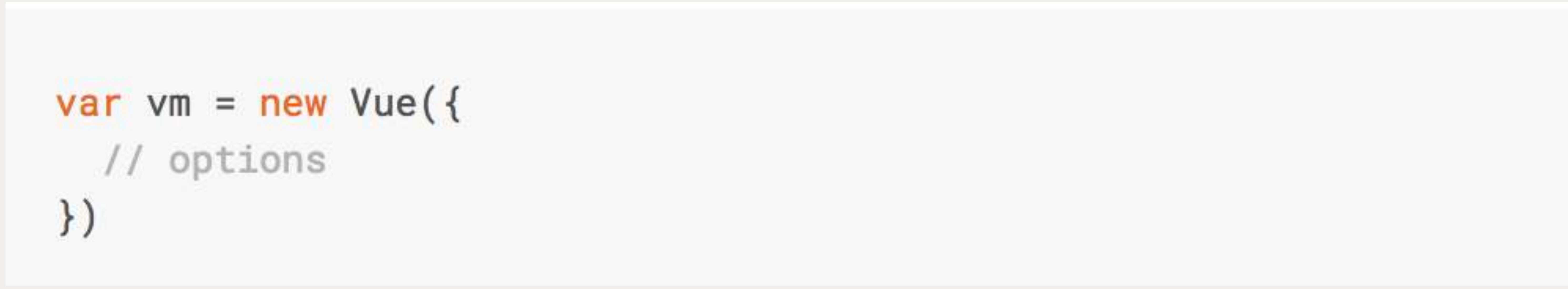
JS, TS
CoffeeScript...? 😳

CSS, Sass, Less, Stylus

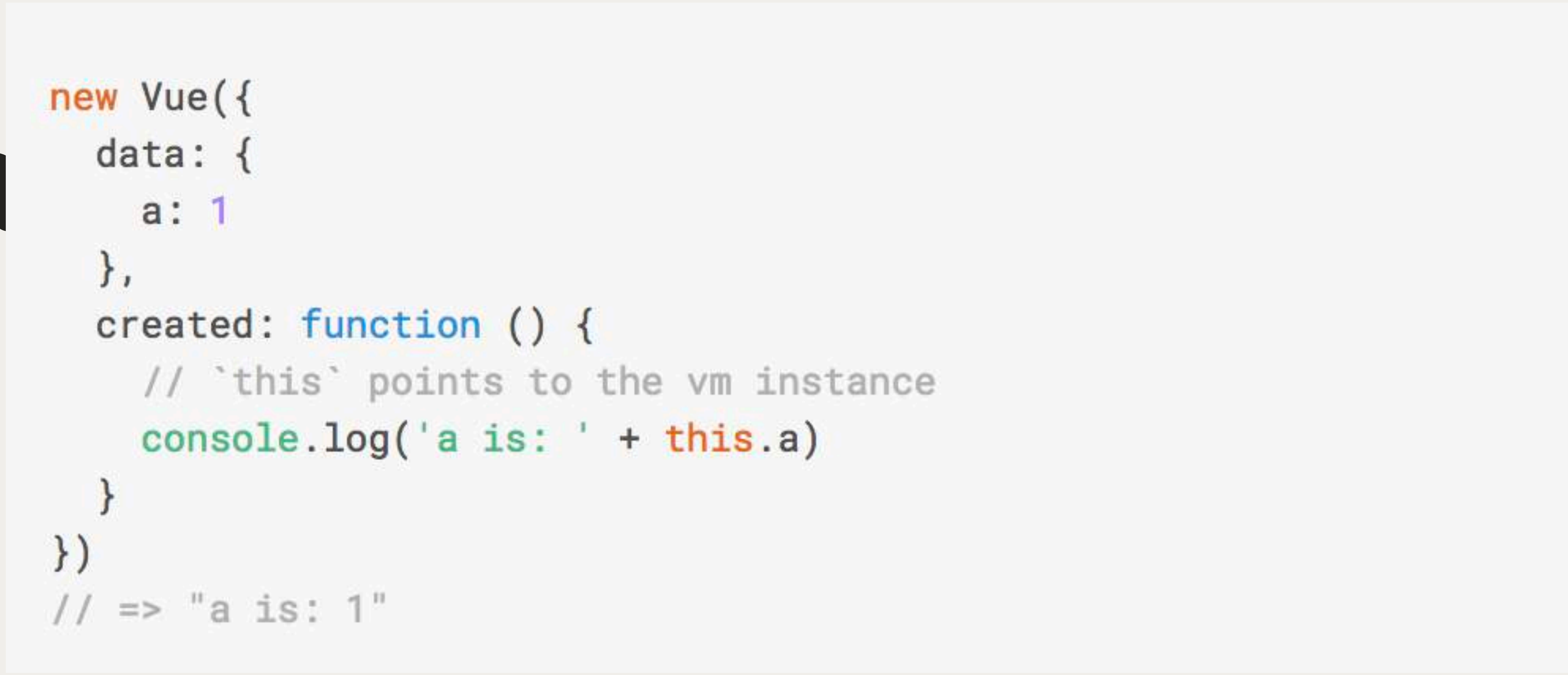
The Vue instance

https://vuejs.org/v2/guide/instance.html

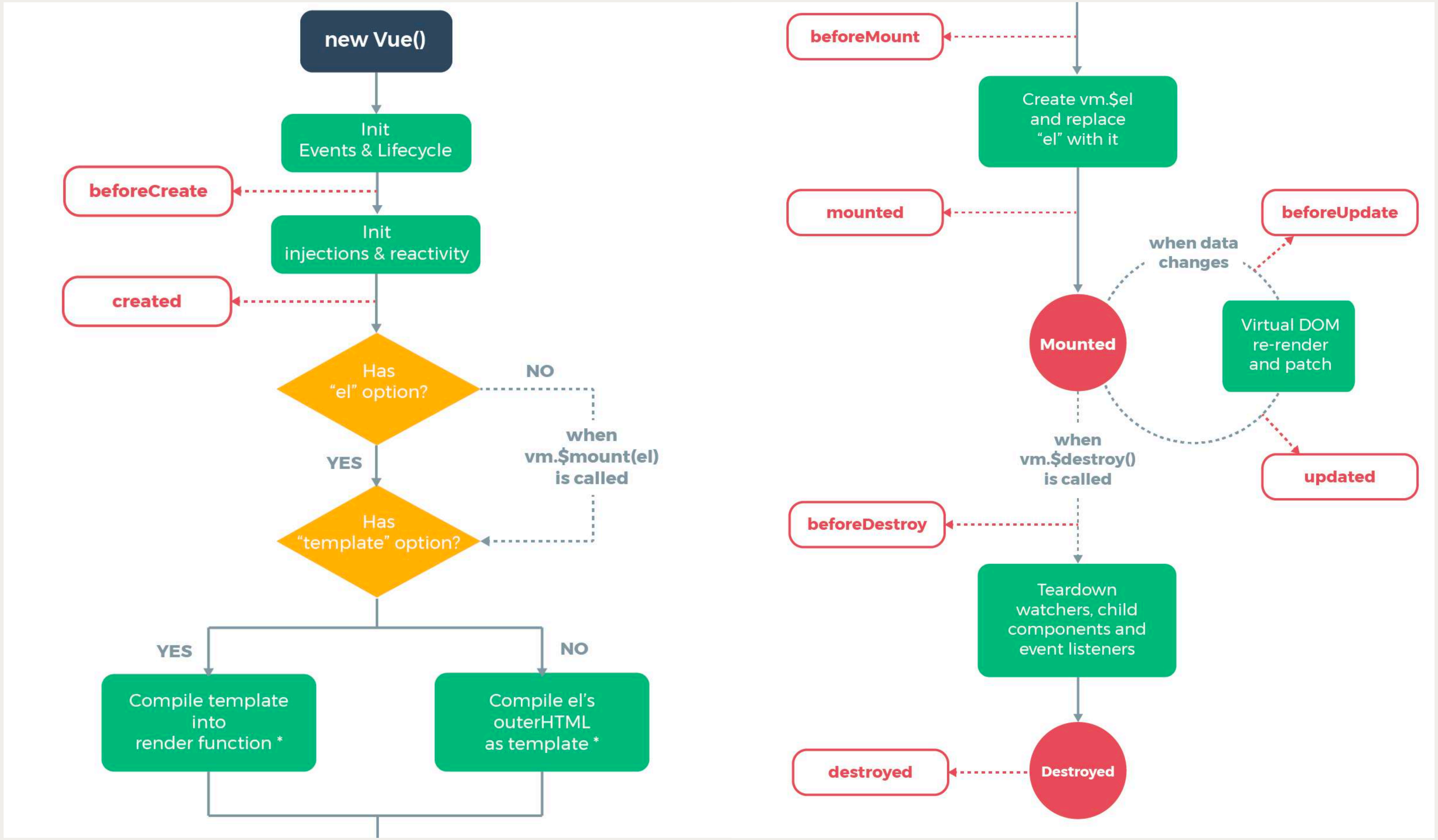
The Vue insta



```
var vm = new Vue({
  // options
})
```



```
new Vue({
  data: {
    a: 1
  },
  created: function () {
    // `this` points to the vm instance
    console.log('a is: ' + this.a)
  }
})
// => "a is: 1"
```


new Vue()
Init Events & Lifecycle
beforeCreate
Init injections & reactivity
created
Has "el" option?
NO
YES
when vm.$mount(el) is called
Has "template" option?
YES
NO
Compile template into render function *
Compile el's outerHTML as template *
beforeMount
Create vm.$el and replace "el" with it
mounted
when data changes
beforeUpdate
Mounted
Virtual DOM re-render and patch
updated
when vm.$destroy() is called
beforeDestroy
Teardown watchers, child components and event listeners
destroyed
Destroyed

Vue object

Vue object lifecycle can be used without rendering anything

Vue objects as models

Vue objects as services

Components

Scoped, reusable interface elements with a JS view-model, HTML template and component styles (CSS, Sass, Less…)

Authored as single-file components (most of the time)

Can be written as render functions, composed from multiple files etc. if desired

Vuex

Official Flux-like implementation

Canonical solution for larger projects

...but not the only one suggested by the devs or the community

https://vuejs.org/v2/guide/state-management.html

Transitions

Extremely useful, intuitive and easy

Works perfectly with built-in conditional logic

Can be wrapped to create a more customised transition code flow

Extensible via parameters, JS hooks etc.

Transitions guide

Extending Vue

Custom directives

Mixins

Filters

Plugins

Injecting any library into main instance (e.g. axios as $http)

The render function

All templates are compiled into render functions

Hence, templating is pluggable:

HTML
JSX
Render functions

Nuxt.js

Commonly used, fairly full-featured solution for universal Vue.js sites and applications.

Boilerplates available via vue-cli

nuxt is used as a dependency.

https://nuxtjs.org/

Easy to get started, works relatively well. Universal JS is still not perfect in practice, regardless of choice of platform… but it's pretty good, worth using over many other solutions.

Server-side rendering

https://vuejs.org/v2/guide/ssr.html

Styles

Global styles vs component styles

Scoped styles

CSS, Sass, Less etc.

Testing

Official webpack template: Jest + Nightwatch

Official test-utils library available

Not much different from other JS frameworks

Tooling

Webpack is the standard for SPAs and universal apps

Several templates available

Lots of standard tooling-related solutions available (Webpack template, vue-loader, vue-test-utils etc.)

Resources

Can be used one a static website (no tooling integration required)

Render functions

JSX 🙄 support available

Server or browser

Custom code can easily be injected into Vue instances

Everything I'm about to tell you is amazingly well architecture and extensible.

Vue.js on GitHub

https://github.com/vuejs

Articles and examples

https://alligator.io/vuejs/

Working demos

https://vuejs.org/v2/examples/

View model cheat sheet

https://github.com/LeCoupa/awesome-cheatsheets/blob/master/frontend/vue.js

What my code looks like

https://github.com/Eiskis/bellevue/blob/master/src/components/pages/PageDemo.vue

https://github.com/Eiskis/bellevue/blob/master/src/components/snippets/Bitmap.vue

https://bitbucket.org/Eiskis/cv-nuxt/src/master/pages/cv.vue

Higher-order components

https://github.com/Eiskis/bellevue/blob/master/src/components/animations/Animation.vue

Vuex

https://github.com/vuejs/vuex

Mature ecosystem

Vue

Vuex

vue-router

Weex

vue-nativescript

vue-i18n

Mature ecosystem

Vue

Vuex

vue-router

Weex

vue-nativescript

vue-i18n

vue-loader

Vue Server Renderer

vue-test-utils

Vue Devtools

Vetur

eslint-plugin-vue

https://github.com/vuejs - 81 repositories :O

Documentation & resources

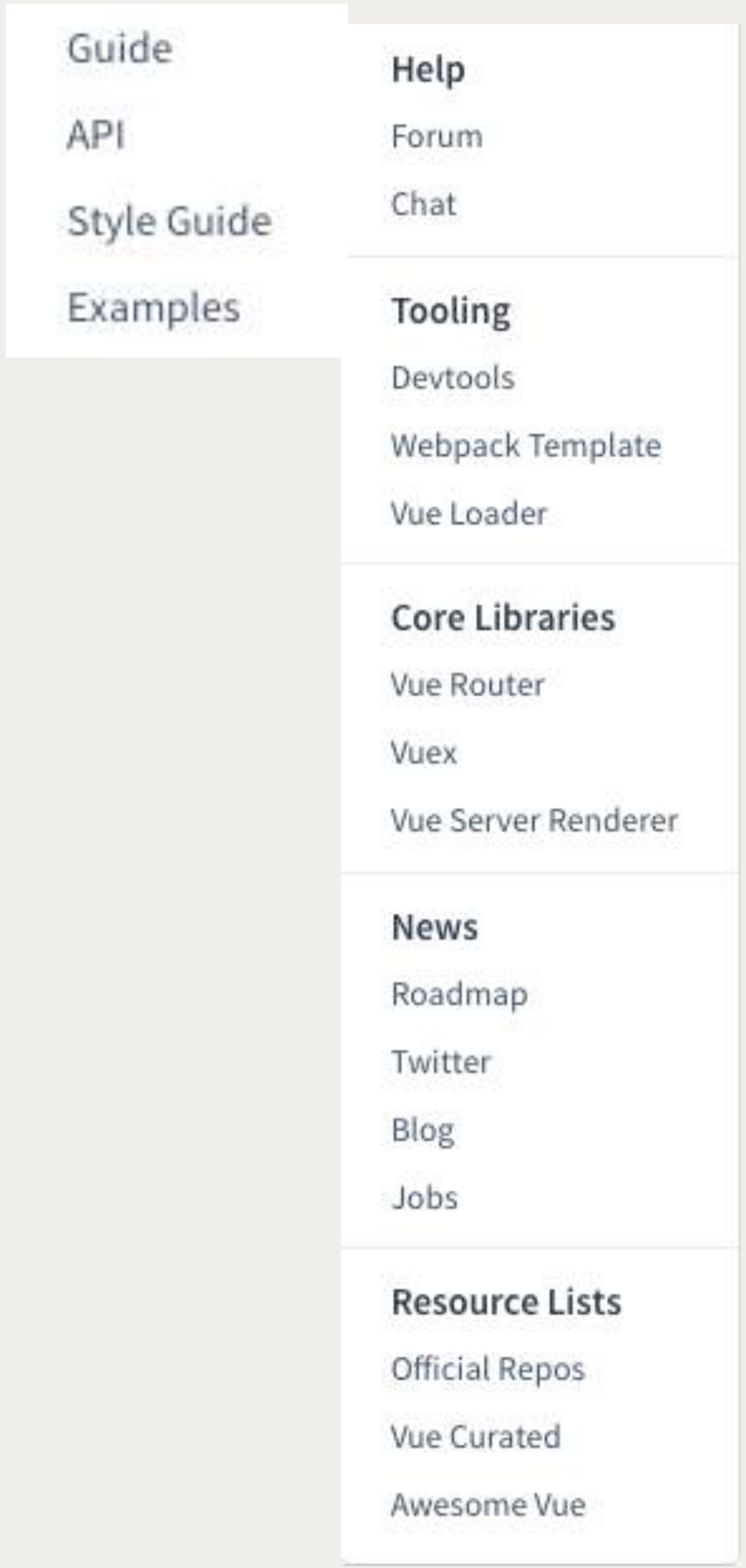
Great documentation: https://vuejs.org/

Very good guidance for newcomers

Full API reference

Same can be said for additional materials and sister projects

Practicality seems to always have been a major design goal



Community support

Awesome Vue

https://github.com/vuejs/awesome-vue

Templates

Several officially authored and maintained project templates

An official command line tool exists for this: vue-cli

https://github.com/vuejs/vue-cli

Mostly Hello Worlds for different setups

More full-featured community-driven solutions are available

Vue CLI 3

Next level of tooling coming this year

Maintainable project scaffolding: no templates, no eject pattern

Built-in dev server for rapid prototyping

vue serve App.vue

https://vuejsdevelopers.com/2018/03/26/vue-cli-3

Vue and me

Switched to Vue about 2 years ago

Previously preferred Knockout over Angular

MVVM similar to Knockout

Picked it up very fast, moved a Knockout project to Vue without much hassle

Used it on several projects (either with plenty of custom tooling or on server-rendered sites to spice up frontend)

Moved to Webpack about a year ago, never looked back

Personal development experience

Switched to Vue about 2 years ago

Previously selected Knockout over Angular

Picked up Vue fast, moved a Knockout project to Vue easily

Used Vue for several different projects (custom tooling, canonical SPA, Nuxt...)

Moved to Webpack about a year ago, never looked back

Worked with and set up a few React projects

React sucks

Bellevue

A more full-featured project template for real-life projects

Objective: Learn, set up and document tooling only once (linting, IDE integration, tests, global styling, SVG pipeline, etc. etc.)

One rewrite already done

Looking forward to Webpack 4… 😓

Bellevue

- Demo: bellevue.netlify.com
- Documentation: eiskis.gitbooks.io/bellevue
- Source and issues: github.com/Eiskis/bellevue

Comparison

Switched to Vue about 2 years ago

Previously preferred Knockout over Angular

MVVM similar to Knockout

Picked it up very fast, moved a Knockout project to Vue without much hassle

Used it on several projects (either with plenty of custom tooling or on server-rendered sites to spice up frontend)

Moved to Webpack about a year ago, never looked back

Official comparison

Vue's official stance on things:

https://vuejs.org/v2/guide/comparison.html

This has actually been run through React team as well!

Comparison

“Vue is great when you need something simple”

This is true. But Vue also scales and works for complex projects

“React is more flexible”

Well, maybe. But in what terms is Vue not flexible enough?

“React has more support and plugins”

Never found this to be an issue even remotely

Better to count the number of quality plugins vs any plugins

Beyond simple web

Vue is built in a way that supports this:

Weex: https://github.com/apache/incubator-weex

Nativecript-Vue: https://nativescript-vue.org/

https://alligator.io/vuejs/getting-started-vue-nativescript/

Production-ready? Probably depends

AR? VR?

Next steps

Set up a new project with vue-cli: vue init webpack mytestapp

Video tutorials: https://www.youtube.com/results?search_query=vue

Go through Vue guide: https://vuejs.org/v2/guide/

Get familiar with projects: https://github.com/vuejs/awesome-vue

That's it!

Jerry Jäppinen

Product design consultant

DCMN elisa visable OLX ProductBeat

jerryjappinen@lateralnord.com
+358 40 7188776
@jerryjappinen

Lateral Nord