# ADVANCED VUE JS WORKSHOP_III_




ABOUT ME

# @Rafael_Casuso

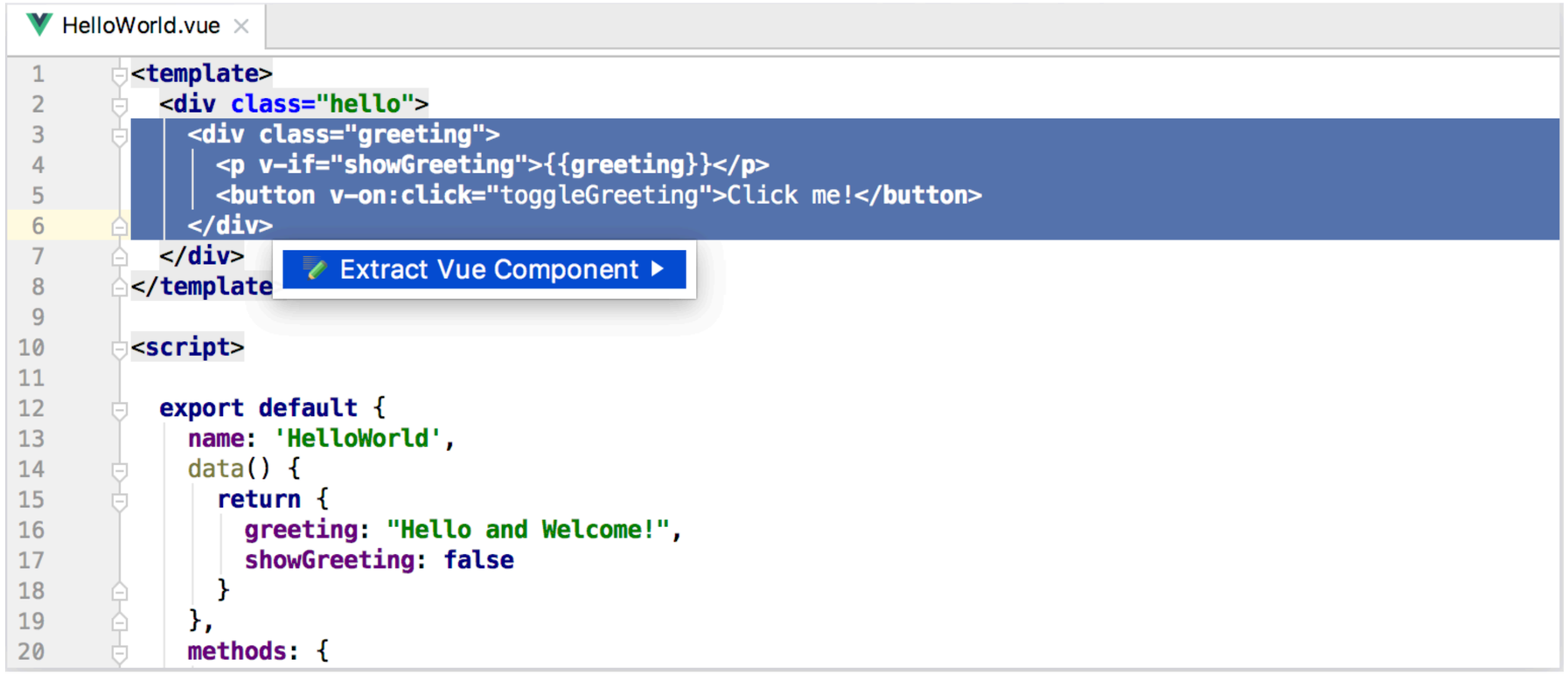
- **CTO @StayApp**
- **CEO @SnowStormIO**
- **Organizer @VueJsMadrid, @BotDevMad**
- **Software Engineer** with +10 years of experience leading teams and developing.
- **Software Architect** looking for revolutionary ways to change the world.
- **Specialties**: JavaScript, NodeJS, Conversational Intelligences.

+ ADVANCED

# REFACTORING

# COMPONENT REFACTORING_

- EXTRACTING SMALLER COMPONENTS IS THE BASE OF VUE REFACTORING
- LARGE COMPONENTS ARE A SMELL OF REFACTORING NEED



# REUSABLE COMPONENTS_

- REFACTOR INTO REUSABLE COMPONENTS ALLOW SUCINT CODE
- COMPONENT API INCLUDE PROPS, EVENTS & SLOTS
- ACCESSING CHILD COMPONENTS NEEDS REF DECLARATION

```
<my-component
  :foo="baz"
  :bar="qux"
  @event-a="doThis"
  @event-b="doThat"
>
  <img slot="icon" src="...">
  <p slot="main-text">Hello!</p>
</my-component>
```

```
<div id="parent">
  <user-profile ref="profile"></user-profile>
</div>
```

```
var parent = new Vue({ el: '#parent' })
// access child component instance
var child = parent.$refs.profile
```

# MIXINS_

- OBJECT THAT CAN CONTAIN ANY COMPONENT FUNCTIONS SUCH AS COMPUTED PROPERTIES, METHODS, HOOKS OR DATA PROPERTIES
- OPTIONS MERGE:
  - DATA PROPERTIES FOLLOW ONE-LEVEL DEEP MERGE, WITH COMPONENT PROPERTIES PRIORITY
  - HOOK FUNCTIONS ARE MERGED INTO AN ARRAY AND MIXINS' EXECUTE FIRST
  - OBJECTS LIKE METHODS, DIRECTIVES OR COMPONENTS HAVE COMPONENT PRIORITY

# MIXINS_

```
// define a mixin object
var myMixin = {
  created: function () {
    this.hello()
  },
  methods: {
    hello: function () {
      console.log('hello from mixin!')
    }
  }
}

// define a component that uses this mixin
var Component = Vue.extend({
  mixins: [myMixin]
})

var component = new Component() // => "hello from mixin!"
```

```
var mixin = {
  methods: {
    foo: function () {
      console.log('foo')
    },
    conflicting: function () {
      console.log('from mixin')
    }
  }
}

var vm = new Vue({
  mixins: [mixin],
  methods: {
    bar: function () {
      console.log('bar')
    },
    conflicting: function () {
      console.log('from self')
    }
  }
})

vm.foo() // => "foo"
vm.bar() // => "bar"
vm.conflicting() // => "from self"
```

# CUSTOM DIRECTIVES_

- PRIMARY WAY OF REUSABILITY IN VUE IS COMPONENTS
- WHEN YOU NEED LOW-LEVEL ACCESS ON ELEMENTS
- CUSTOM DIRECTIVES CAN BE GLOBAL OR COMPONENT-BASED
- HOOKS:
  - Bind (Only once, directive first bound to element)
  - Inserted (Bound element inserted into its parent node)
  - Update (After containing component VNode has been updated)
  - ComponentUpdated (After cont. component and children have been updated)
  - Unbind (Only once, when directive is unbound from element)

# CUSTOM DIRECTIVES_

- DIRECTIVE HOOK ARGUMENTS:
  - el: element the directive is bound to
  - binding:
    - name: name of the directive
    - value: value of the resolved javascript expression
    - oldValue: only in update or componentUpdate
    - expression: javascript expression as a string
    - arg: argument if passed
    - modifiers: as an object

# CUSTOM DIRECTIVES_

```
directives: {
  focus: {
    // directive definition
    inserted: function (el) {
      el.focus()
    }
  }
}
```

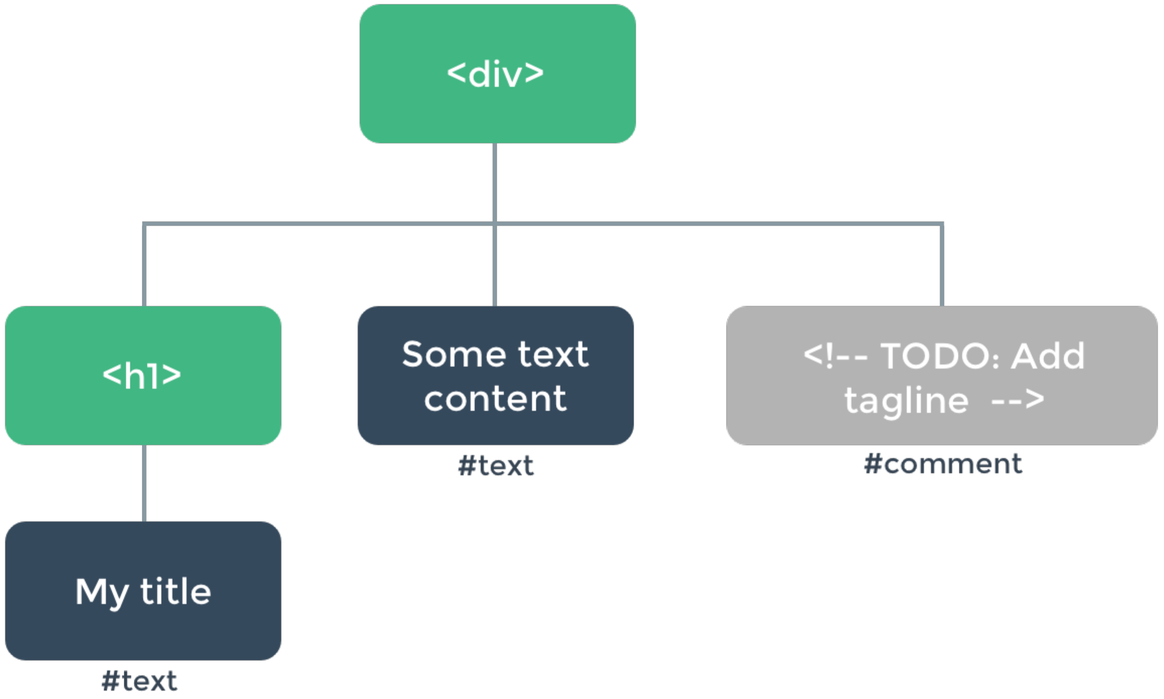
```
Vue.directive('color-swatch', function (el, binding) {
  el.style.backgroundColor = binding.value
})
```

```
Vue.directive('demo', {
  bind: function (el, binding, vnode) {
    var s = JSON.stringify
    el.innerHTML =
      'name: '       + s(binding.name) + '<br>' +
      'value: '      + s(binding.value) + '<br>' +
      'expression: ' + s(binding.expression) + '<br>' +
      'argument: '   + s(binding.arg) + '<br>' +
      'modifiers: '  + s(binding.modifiers) + '<br>' +
      'vnode keys: ' + Object.keys(vnode).join(', ')
  }
})

new Vue({
  el: '#hook-arguments-example',
  data: {
    message: 'hello!'
  }
})
```

# RENDER FUNCTION_

- TEMPLATES ARE THE RECOMMENDED DECLARATIVE WAY TO RENDER CONTENT IN VUE
- SOME CASES NEED RENDER FUNCTION POWER



# RENDER FUNCTIONS: FUNDAMENTALS_

- VUE AUTOMATICALLY KEEPS REACTIVE PROPERTIES UPDATED AS USUAL
- CREATE-ELEMENT IS AN HTML/ COMPONENT DESCRIPTION THAT GENERATES A VIRTUAL NODE

```
Vue.component('anchored-heading', {
  render: function (createElement) {
    return createElement(
      'h' + this.level,   // tag name
      this.$slots.default // array of children
    )
  },
  props: {
    level: {
      type: Number,
      required: true
    }
  }
})
```

# RENDER FUNCTIONS: CREATE ELEMENT_

- VUE AUTOMATICALLY KEEPS REACTIVE PROPERTIES UPDATED AS USUAL
- CREATE-ELEMENT IS AN HTML/COMPONENT DESCRIPTION THAT GENERATES A VIRTUAL NODE
- DATA CAN BE CLASSES, DIRECTIVES, PROPS OR EVENT HANDLERS

```
// @returns {VNode}
createElement(
  // {String | Object | Function}
  // An HTML tag name, component options, or function
  // returning one of these. Required.
  'div',

  // {Object}
  // A data object corresponding to the attributes
  // you would use in a template. Optional.
  {
    // (see details in the next section below)
  },

  // {String | Array}
  // Children VNodes, built using `createElement()`,
  // or using strings to get 'text VNodes'. Optional.
  [
    'Some text comes first.',
    createElement('h1', 'A headline'),
    createElement(MyComponent, {
      props: {
        someProp: 'foobar'
      }
    })
  ]
)
```

# FUNCTIONAL COMPONENTS_

- FUNCTIONAL COMPONENTS ARE:
  - STATELESS: THEY HAVE NO DATA
  - INSTANCELESS: THEY HAVE NO ‘THIS’ CONTEXT
- EVERYTHING IS PASSED THROUGH CONTEXT (PROPS, DATA, PARENT, ETC)

```
Vue.component('my-component', {
  functional: true,
  // To compensate for the lack of an instance,
  // we are now provided a 2nd context argument.
  render: function (createElement, context) {
    // ...
  },
  // Props are optional
  props: {
    // ...
  }
})
```

# PLUGINS_

- USUALLY ADD GLOBAL LEVEL FUNCTIONALITY
- THEY CAN ADD GLOBAL METHODS, PROPERTIES, DIRECTIVES, FILTERS, TRANSITIONS
- THEY CAN ADD COMPONENT OPTIONS THROUGH GLOBAL MIXINS
- EXTEND VUE.PROTOTYPE WITH NEW METHODS
- THEY HAVE TO EXPOSE THE INSTALL METHOD
- MULTIPLE USES OF A PLUGIN ARE CACHED

# PLUGINS_

```
MyPlugin.install = function (Vue, options) {
  // 1. add global method or property
  Vue.myGlobalMethod = function () {
    // something logic ...
  }

  // 2. add a global asset
  Vue.directive('my-directive', {
    bind (el, binding, vnode, oldVnode) {
      // something logic ...
    }
    ...
  })

  // 3. inject some component options
  Vue.mixin({
    created: function () {
      // something logic ...
    }
    ...
  })

  // 4. add an instance method
  Vue.prototype.$myMethod = function (methodOptions) {
    // something logic ...
  }
}
```

```
Vue.use(MyPlugin, { someOption: true })
```

# REFACTORING DATA ACCESS_

- WHEN DATA HAS TO BE SHARED BETWEEN COMPONENTS IT CAN BECOME STATE PROPERTIES
  - DERIVED COMPUTED PROPERTIES MUST BECOME THEN GETTERS
  - CHANGES OVER THOSE PROPERTIES MUST BECOME MUTATIONS
- WHEN ASYNC DATA CALLS ARE NEEDED ACROSS THE APPLICATION THEY CAN BECOME ACTIONS

+ ADVANCED

# TESTING

# SETUP_

- ANY TEST RUNNER WITH MODULE-TEST CAPABILITIES WILL WORK
- KARMA AND JEST ARE RECOMMENDED
- JEST IS CURRENTLY THE OFFICIAL TEST RUNNER
- NIGHTWATCHJS FOR E2E IF YOU WANT A SELENIUM SERVER
- CYPRESS IF YOU DON’T NEED SELENIUM SERVER
- TEST DIRECTORY STRUCTURE IS CLASSIC

# ASSERTIONS_

- USUALLY YOU WANT TO EVALUATE COMPONENT MOUNTED ON V-DOM
- THEN YOU CAN USE COMPONENT INSTANCE API PROPERTIES AND METHODS

```
import Vue from 'vue'
import MyComponent from './MyComponent.vue'

// helper function that mounts and returns the rendered text
function getRenderedText (Component, propsData) {
  const Constructor = Vue.extend(Component)
  const vm = new Constructor({ propsData: propsData }).$mount()
  return vm.$el.textContent
}

describe('MyComponent', () => {
  it('renders correctly with different props', () => {
    expect(getRenderedText(MyComponent, {
      msg: 'Hello'
    })).toBe('Hello')

    expect(getRenderedText(MyComponent, {
      msg: 'Bye'
    })).toBe('Bye')
  })
})
```

# ASSERTING ASYNCHRONOUS UPDATES_

- ASSERTIONS ON DOM UPDATES RESULTING FROM STATE UPDATES
- NEXT TICK IS NEEDED TO ASSURE THE DOM UPDATE LOOP
- THAT'S BECAUSE DOM UPDATE IS ASYNCHRONOUS

```
// Inspect the generated HTML after a state update
it('updates the rendered message when vm.message updates', done => {
  const vm = new Vue(MyComponent).$mount()
  vm.message = 'foo'

  // wait a "tick" after state change before asserting DOM updates
  Vue.nextTick(() => {
    expect(vm.$el.textContent).toBe('foo')
    done()
  })
})
```

# VUE TEST UTILS (BETA): SETUP_

- COMMON HELPERS FOR UNIT TESTING EXPOSED THROUGH VUE-TEST-UTILS PACKAGE
- https://github.com/vuejs/vue-test-utils
- git clone https://github.com/vuejs/vue-test-utils-getting-started
- cd vue-test-utils-getting-started
- npm install

# VUE TEST UTILS (BETA): FUNDAMENTALS_

- COMPONENTS ARE MOUNTED IN ISOLATION
- INPUTS ARE MOCKED (PROPS, INJECTION, DATA)
- ASSERTIONS ON OUTPUTS (RENDERED RESULT, EMITTED EVENTS)
- MOUNTED COMPONENTS ARE RETURNED INSIDE A ‘WRAPPER’
- WRAPPER OBJECT EXPOSES METHODS FOR MANIPULATING, TRAVERSING AND QUERYING THE UNDERLYING COMPONENT INSTANCE

# VUE TEST UTILS (BETA)_

- HTML OUTPUT CAN BE TRAVERSED
- USER INTERACTION CAN BE EMULATED

```
it('button click should increment the count', () => {
  expect(wrapper.vm.count).toBe(0)
  const button = wrapper.find('button')
  button.trigger('click')
  expect(wrapper.vm.count).toBe(1)
})
```

```
import { mount } from '@vue/test-utils'
import Counter from './counter'

describe('Counter', () => {
  // Now mount the component and you have the wrapper
  const wrapper = mount(Counter)

  it('renders the correct markup', () => {
    expect(wrapper.html()).toContain('<span class="count">0</span>')
  })

  // it's also easy to check for the existence of elements
  it('has a button', () => {
    expect(wrapper.contains('button')).toBe(true)
  })
})
```

+ ADVANCED

# THIRD PARTY RESOURCES

# COLLECTIONS OF COMPONENTS & PLUGINS_

- VUETIFY
  - MOST RECOMMENDED MATERIAL-DESIGN COMPONENT LIBRARY
- AWESOME VUE
  - https://github.com/vuejs/awesome-vue
  - PROJECTS, COMPONENTS AND LIBRARIES
- CURATED VUE
  - https://curated.vuejs.org
  - HIGH QUALITY PACKAGES INCLUDING VUEXFIRE OR VUE-METEOR

# FRAMEWORKS_

- QUASAR
  - WEB
  - MOBILE WEBAPP
- WEEX
  - WEB
  - NATIVE MOBILE APPS
- NATIVESCRIPT
  - NATIVE MOBILE APPS

# THANK YOU