

Vue.js | Angular | React

Veja diferenças de cada framework!

José Barbosa | [@kidchenko](https://twitter.com/kidchenko)

José Barbosa | @kidchenko

- Programador na **Lambda3**;
- Cloud Computing <3 **Azure**;
- Ecossistema de **Startups**;
- .Net, JavaScript e Front-end **Developer**;
- Docker <3;



Podcast 9 – O programador poliglota



Giovanni Bassi

26 de agosto de 2016

Podcast

linguagens, podcast

Nenhum comentário

Editar

Tempo de leitura: 2 minuto(s)

Nesse episódio nós te contamos porque você precisa saber mais de uma linguagem de programação. Aliás, muito mais que uma, ou duas ou três. Discutimos como aprender novas linguagens, os tipos de linguagem que existem, por onde começar essa caminhada, e contamos quais linguagens diferentes nós já usamos, e quais estão na nossa pauta. Inspire-se e venha com a gente aprender linguagens novas!

[Continue lendo](#)

Acompanhe nosso podcast:

**WE'RE
HIRING!**

m.br/category/p



www.lambda3.com.br



Vue.js

EVAN YOU



Oct 25, 2015

VUE.JS: A (RE)INTRODUCTION



Vue.js is a library for building web interfaces. Together with some other tools you can also call it a “framework”, although it’s more like a set of optional tools that work together really well. Now, if you’ve never heard of or used Vue before, you are probably thinking: great, yet another JavaScript framework! I get it. Turns out Vue isn’t particularly new—I first started working on its prototype almost two years ago, and the first public release was in **February 2014**. Over the time it has been evolving, and today **many are using it in production**.

<http://blog.evanyou.me/2015/10/25/vuejs-re-introduction/>

REACTIVITY

Keeping the state and the view in sync is hard. Or is it?

index.js

```
new Vue({  
  el: '#app',  
  data: {  
    message: 'Vue.js SP!'  
  }  
})
```

index.html

```
<script src="https://npmcdn.com/vue/dist/vue.js"></script>  
<div id="app">  
  {{ message }}  
</div>  
<script src="index.js"></script>
```



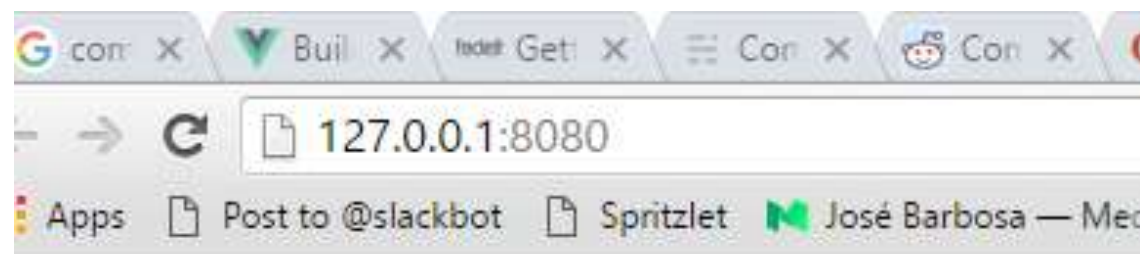

Vue.js SP!

index.js

```
var example = new Vue({  
  el: '#app',  
  data: {  
    count: 1  
  },  
  computed: {  
    comp: function () {  
      return this.count + 1  
    }  
  }  
})
```

index.html

```
<script src="https://npmcdn.com/vue/dist/vue.js"></script>
<div id="app">
  <ul>
    <li>Count: {{ count }}</li>
    <li>Computed: {{ comp }}</li>
  </ul>
</div>
<script src="index.js"></script>
```



- Count: 1
- Computed: 2

COMPONENTS

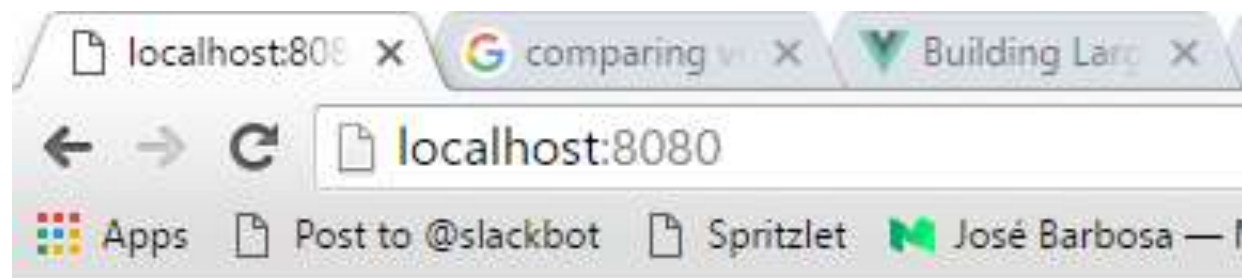
Ok the data binding is neat for small demos. What about big apps?

index.js

```
// define
var MeetupComponent = Vue.extend({
  template: '<div>Meetup Vue SP!</div>'
})
// register
Vue.component('meetup', MeetupComponent)
// create a root instance
new Vue({
  el: '#example'
})
```

index.html

```
<script src="https://npmcdn.com/vue/dist/vue.js"></script>  
<div id="example">  
  <meetup></meetup>  
</div>  
<script src="index.js"></script>
```



Meetup Vue SP!

MODULARITY

It's 2015 and we shouldn't put everything in the global scope!

ComponentA.js

```
// ComponentA.js
export default {
  template: '<div>{{ message }}</div>',
  data () {
    return {
      message: 'Hello Vue.js!'
    }
  }
}
```

App.js

```
// App.js
import ComponentA from './ComponentA'

export default {
  // use another component, in this scope only.
  // ComponentA maps to the tag <component-a>
  components: { ComponentA },
  template: `
    <div>
      <p>Now I'm using another component.</p>
      <component-a></component-a>
    </div>
  `
}
```

MyComponent.vue

```
<!-- MyComponent.vue -->

<!-- css -->
<style>
.message {
  color: red;
}
</style>

<!-- template -->
<template>
  <div class="message">{{ message }}</div>
</template>

<!-- js -->
<script>
export default {
  props: ['message'],
  created() {
    console.log('MyComponent created!')
  }
}
</script>
```

example.vue

```
< > example.vue x ...
1 <template lang="jade">
2   div.my-component
3     h1 {{ msg }}
4     other-component
5 </template>
6
7 <script>
8   import OtherComponent from './other-component.vue'
9
10  export default {
11    components: { OtherComponent },
12    data() {
13      return {
14        msg: 'Hello Vue.js!'
15      }
16    }
17  }
18 </script>
19
20 <style lang="sass" scoped>
21   $font-stack: Helvetica, sans-serif;
22   $primary-color: #333;
23
24   .my-component {
25     font: 100% $font-stack;
26     color: $primary-color;
27   }
28 </style>

Line 1, Column 23      Spaces: 2    Vue Component
```

QuickTime PlayerFileEditViewWindowHelp

localhost:8080/bundle

localhost:8080/bundle

Counter Hot

Current count: 4

counter.js

```
1 module.exports = {
2
3   template:
4     `

5       <h1>Counter Hot</h1>
6       <p>Current count: {{count}}</p>
7     </div>`,
8
9   data () {
10     return { count: 0 }
11   },
12
13   ready () {
14     this.handle = setInterval(() => {
15       this.count++
16     }, 1000)
17   },
18
19   destroyed () {
20     clearInterval(this.handle)
21   }
22 }


```

Line 5, Column 22Spaces: 2JavaScript Next

ROUTING

So I want to build an app, but where's the router?

index.js

```
import Vue from 'vue'
import VueRouter from 'vue-router'
import App from './app.vue'
import ViewA from './view-a.vue'
import ViewB from './view-b.vue'
```

```
Vue.use(VueRouter)
```

```
const router = new VueRouter()
```

```
router.map({
  '/a': { component: ViewA },
  '/b': { component: ViewB }
})
```

```
router.start(App, '#app')
```


app.vue

```
<div>
  <h1>This is the layout that won't change</h1>
  <router-view><!-- matched component renders here --></router-view>
</div>
```

Two Way binding

Here where the magic happens?

index.js

```
new Vue({  
  el: '#app',  
  data: {  
    message: 'Oi meetup Vuejs SP!'  
  }  
})
```

index.html

```
<script src="https://npmcdn.com/vue/dist/vue.js"></script>  
<div id="app">  
  <p>{{ message }}</p>  
  <input v-model="message">  
</div>  
<script src="index.js"></script>
```

Render a List

```
todos: [  
  { text: 'Learn JavaScript' },  
  { text: 'Learn Vue.js' },  
  { text: 'Build Something Awesome' }  
]
```

index.js

```
new Vue({  
  el: '#app',  
  data: {  
    todos: [  
      { text: 'Aprender Vue.js' },  
      { text: 'Usar Vue.js em um projeto' },  
      { text: 'Fazer curso de Vue.js' }  
    ]  
  }  
})
```

index.html

```
<script src="https://npmcdn.com/vue/dist/vue.js"></script>
<div id="app">
  <ul>
    <li v-for="todo in todos">
      {{ todo.text }}
    </li>
  </ul>
</div>
<script src="index.js"></script>
```

Handle User Input

Your app do somethig?

index.js

```
new Vue({  
  el: '#app',  
  data: {  
    message: 'Meetup Vue.js SP!'  
  },  
  methods: {  
    reverseMessage: function () {  
      this.message = this.message.split('').reverse().join('')  
    }  
  }  
})
```

index.html

```
<script src="https://npmcdn.com/vue/dist/vue.js"></script>
<div id="app">
  <p>{{ message }}</p>
  <button v-on:click="reverseMessage">Reverter</button>
</div>
<script src="index.js"></script>
```

Todo Example

Show me the code!

index.js

```
new Vue({
  el: '#app',
  data: {
    newTodo: '',
    todos: [
      { text: 'Adicionar todo' }
    ]
  },
  methods: {
    addTodo: function () {
      var text = this.newTodo.trim()
      if (text) {
        this.todos.push({ text: text })
        this.newTodo = ''
      }
    },
    removeTodo: function (index) {
      this.todos.splice(index, 1)
    }
  }
})
```

index.html

```
<script src="https://npmcdn.com/vue/dist/vue.js"></script>
<div id="app">
  <input v-model="newTodo" v-on:keyup.enter="addTodo">
  <ul>
    <li v-for="todo in todos">
      <span>{{ todo.text }}</span>
      <button v-on:click="removeTodo($index)">X</button>
    </li>
  </ul>
</div>
<script src="index.js"></script>
```



About Vue.js and AngularJS comparison

Angular Modules

So I want to build an app, but where's the router?


```
angular.module('myModule', [])
```

```
Vue.extend({  
  data: function(){ return {} },  
  created: function() {},  
  ready: function() {},  
  components: {},  
  methods: {},  
  watch: {}  
  //(other props excluded)  
});
```

Directives

```
myModule.directive('directiveName', function (injectables) {  
  return {  
    restrict: 'A',  
    template: '<div></div>',  
    controller: function() { },  
    compile: function() {},  
    link: function() { }  
    //(other props excluded)  
  };  
});
```

```
Vue.directive('my-directive', {  
  bind: function () {},  
  update: function (newValue, oldValue) {},  
  unbind: function () {}  
});
```

Filters

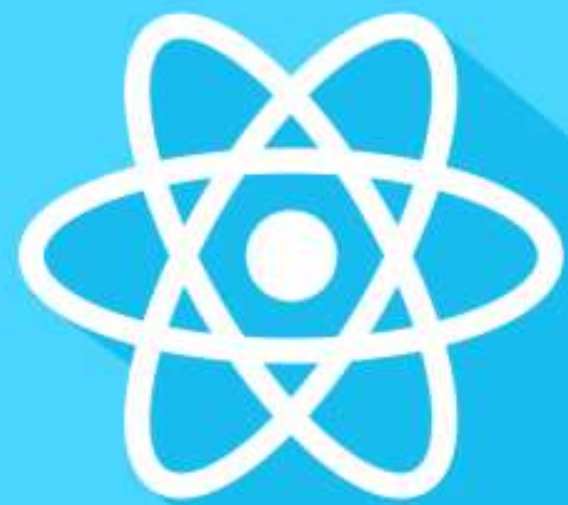
```
myModule.angular.module('filterName', [])  
  .filter('reverse', function() {  
    return function(input) {};  
  });
```

```
Vue.filter('reverse', function (value) {  
  return function(value){};  
});
```

Vue.js vs Angular

- Vue.js do francês; visão;
- Vue.js é **beeeem** mais simples que o angular ele é apenas a view;
- Ele é micro, mais fino, um framework apenas para sua view;
- Angular usa 2 way data binding; Vue.js default é one-way. Com fluxo de pai para filho entre os componentes;
- Baseado em componentes diretivas apenas encaspulam o DOM;
- Vue.js tem melhor performance, Angular é baseado em dirty checking. Vue.js é baseado em Sistema Assíncrono de Filas;


Vue.js or React.js, which should I choose to make frontend?



React

vue.js or react.js, which should i choose to make frontend?



ParaDevil  — 11 months ago

I'm a new, and which one to choose? Please give me some advice.



 malfait.robin — 11 months ago

Depends on what you want to do, but I vote React, no doubt!

ltrain

ParaDevil

michaeljcalkins





🇺🇸 opheliadesign — 11 months ago

I've been happily using **Vue**JS in production ever since completing the tutorials on here. Found it extremely easy to start using and I personally like how a lot of the view logic is directly in my HTML, similar to Blade.

The **Vue**JS FAQ has a fair and honest breakdown of **Vue**JS vs whatever, might want to check it out. Different strokes for different folks. :)

Bokdeal

vitorarjol

strategicsdemexico

Fethi

amarsyla

thomasbabuj

arnabrahman



colourmill — 11 months ago

I like the **Vue** API and syntax a bit more than React's, producing more readable code and making it easy to sink your teeth in if you're new. Function wise I haven't run into any shortcomings, but I haven't used them on gigantic projects yet.

opheliadesign





haugen — 2 months ago

Kind of a late reply, but the notes that came for **Vue.js** 2.0 alpha makes the choice pretty easy.

With the new architecture, there are even more possibilities to explore - for example, rendering to native interfaces on mobile. Currently, we are exploring a port of **Vue.js** 2.0 that uses weex as a native rendering backend, a project maintained by engineers at Alibaba Group, the biggest tech enterprise of China. It is also technically feasible to adapt **Vue** 2.0's virtual-DOM to run inside ReactNative. We are excited to see how it goes!

Vue.js would be my choice.



Iterating

```
var HelloMessage = React.createClass({  
  render: function() {  
    return <div>Hello {this.props.name}</div>;  
  }  
});
```

```
ReactDOM.render(<HelloMessage name="John" />, mountNode);
```

```
var Timer = React.createClass({
  getInitialState: function() {
    return {secondsElapsed: 0};
  },
  tick: function() {
    this.setState({secondsElapsed: this.state.secondsElapsed + 1});
  },
  componentDidMount: function() {
    this.interval = setInterval(this.tick, 1000);
  },
  componentWillUnmount: function() {
    clearInterval(this.interval);
  },
  render: function() {
    return (
      <div>Seconds Elapsed: {this.state.secondsElapsed}</div>
    );
  }
});

ReactDOM.render(<Timer />, mountNode);
```

Props and Components

```
// React.js (.jsx)

var Iteration = React.createClass({
  getInitialState() {
    return {
      array: [1,2,3,4]
    }
  },

  render() {
    this.state.array.map( function(value) {
      return (
        <span>{value}</span>
      )
    });
  }
});

ReactDOM.render(<Iteration />, document.getElementById('array'));
```



```
// Vue.js (js)
```

```
var Iteration = new Vue({  
  el: '#array',  
  data: {  
    array: [1,2,3,4]  
  }  
});
```

Vue.js vs React

- Ambos possuem composição de View components reativos;
- React é baseado em Virtual DOM, quando o estado muda, Virtual DOM aplica o patch no real DOM;
- Virtual DOM++;
- Vue.js usa o DOM como template e mantém uma referência para a atual os bindings;
- Vue.js possui dependência do DOM; Virtual DOM permite ao React ser isomórfico.

Vue.js vs React

- JSX;
- React.render pode ficar macarrônico;
- Vue.js possui um data-binding leve;
- O time do react quer fazer do React uma plataforma agnostica à UI (React Native), Vue.js é focado em uma solução para web.
- React é usado com functional programming patterns.
- Arquiteturas Flux/Redux com Vuex e Revue
- CSS-in-JS solutions vs Vue single file component;

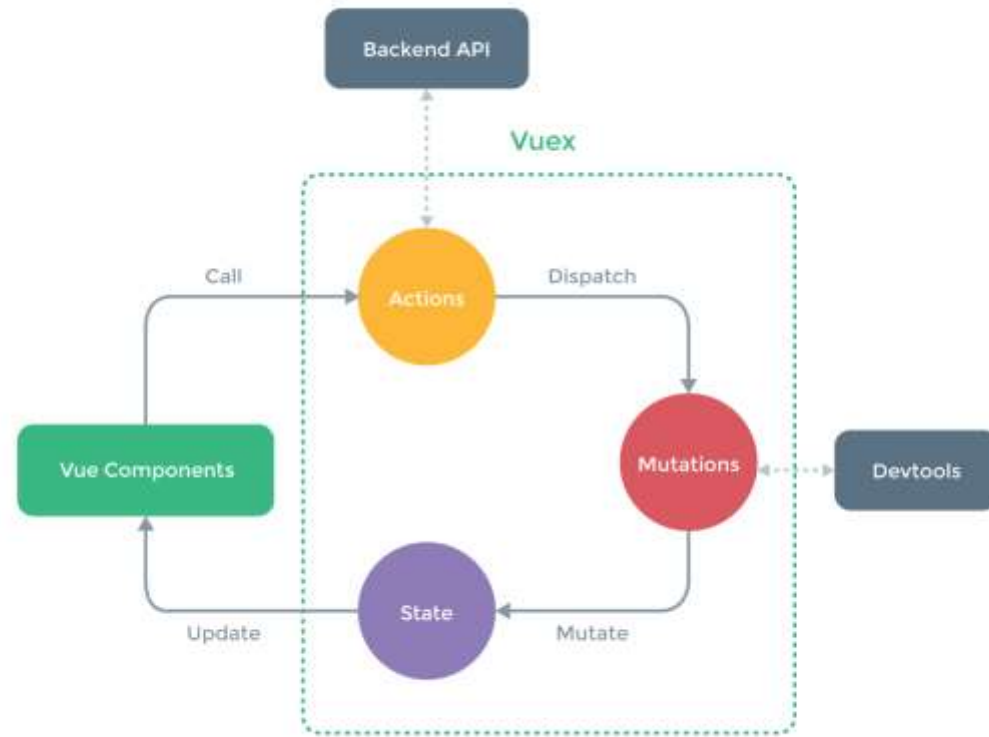
Single File Components

```
< > my-component.vue x
1 <style>
2 .my-component h2 {
3   color: red;
4 }
5 </style>
6
7 <template>
8   <div class="my-component">
9     <h2>{{msg}}</h2>
10   </div>
11 </template>
12
13 <script>
14 module.exports = {
15   data: function () {
16     return {
17       msg: 'hello!'
18     }
19   }
20 }
21 </script>
```

Line 22, Column 1 Spaces: 2 Vue Component

Flux/Redux para Vue.js

Vuex



Revue

store.js

```
import Vue from 'vue'
import Revue from 'revue'
import {createStore} from 'redux'
// create the logic how you would update the todos
import todos from './reducers/todos'
// create some redux actions
import actions from './actions'

// create a redux store
const reduxStore = createStore(todos)
// binding the store to Vue instance, actions are optional
const store = new Revue(Vue, reduxStore, actions)
// expose the store for your component to use
export default store
```

Revue

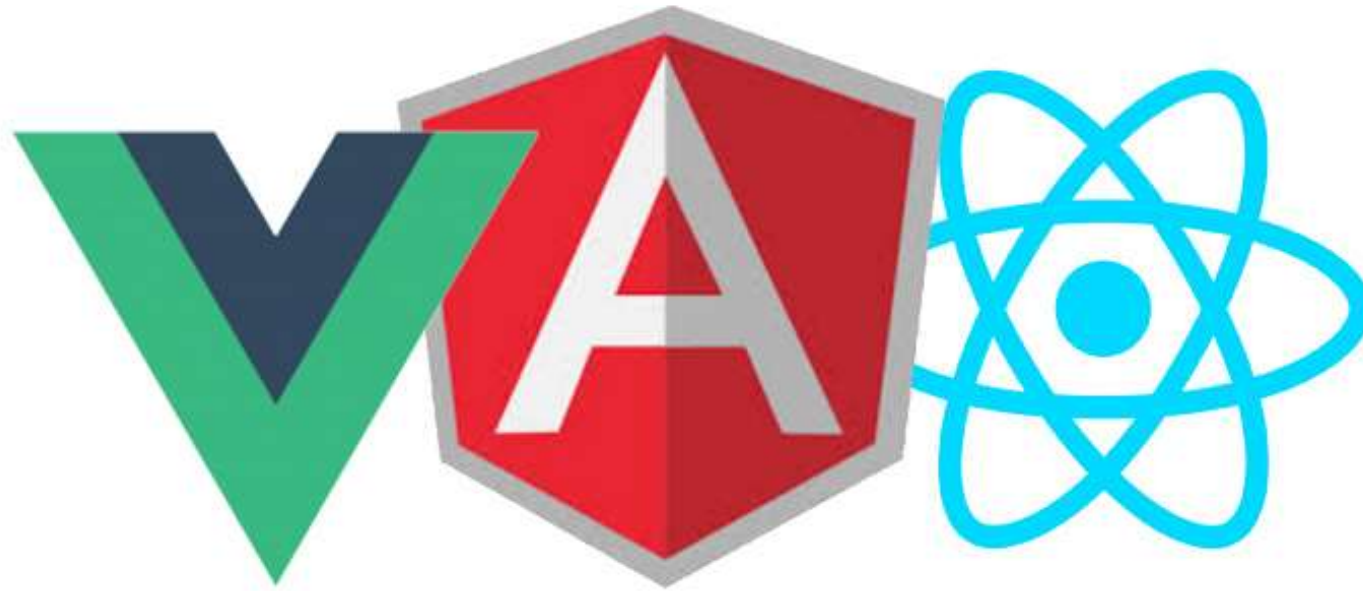
component.js

```
import store from './store'
import * as todoActions from './actions/todo'

export default {
  data() {
    return {
      todo: '',
      todos: this.$select('todos')
      //=> subscribe state.todos to vm.todos
      // if prop is not in top level
      // do this.$select('todos as path.to.todos')
    }
  },
  methods: {
    addTo() {
      store.dispatch({type: 'ADD_TODO', this.todo})
      // or use the actionCreator
      store.dispatch(todoActions.addTo(this.todo))
      // also equal to: (if you binded actions when creating the store)
      const {addTo} = store.actions
      store.dispatch(addTo(this.todo))
    }
  }
}
```




awesome



Obrigado!

Dúvidas?

José Barbosa | [@kidchenko](#)