



UNIVERSIDADE  
FEDERAL DO CEARÁ  
CAMPUS QUIXADÁ

# Fundamentos de Vue.js

QXD0020 - Desenvolvimento de Software para Web

Prof. Bruno Góis Mateus ([brunomateus@ufc.br](mailto:brunomateus@ufc.br))

# Agenda

- Introdução
- Introdução ao VueJs
- Principais aspectos de uma aplicação em VueJs
- Diretivas
- Componentes
- Prática

# Introdução



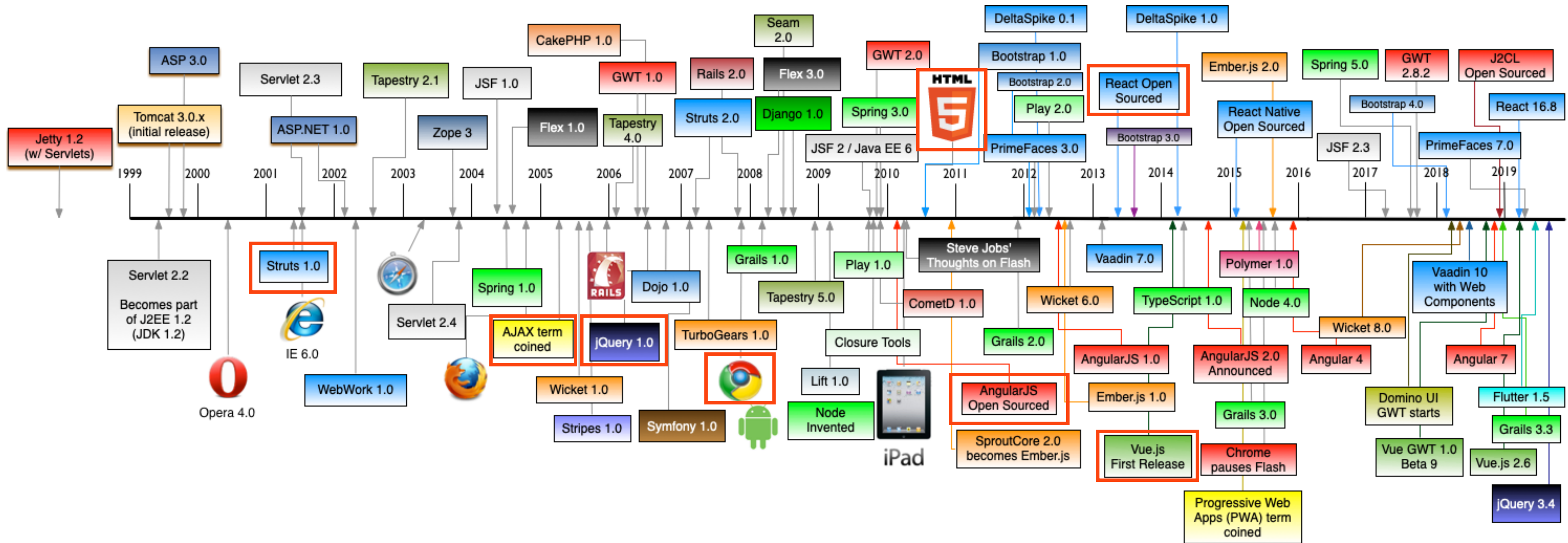
# Introdução

## Motivação

- A interatividade em websites sempre atraiu a atenção de desenvolvedores
- Com o desenvolvimento da Web 2.0, nos anos 2000, a interatividade e o engajamento do usuário passaram a receber um foco ainda maior
  - Companhias como Twitter, Facebook, and YouTube foram criadas nesse período
- Desenvolvedores precisaram se adaptar para permitir esse novo nível de interatividade
  - Bibliotecas e framework foram lançadas para permitir a criação desses sites
  - Em 2006, John Resig lançou o jQuery, que simplificou a escrita de JS no lado do cliente
  - Com o passar do tempo, bibliotecas e frameworks focando no server side também surgiram

# Introdução

## Motivação



# Introdução

## Motivação

**Gmail** by Google BETA

Search Mail Search the Web

Compose Mail « Back to Inbox Archive More actions... 1 of 25 Older »

**Inbox**

Starred ☆  
Sent Mail  
All Mail  
Spam  
Trash

▼ Labels  
quir  
Orkut  
Prius (18)  
Edit labels

**Screenshot** Inbox Apply label...

★ Jason Shellen Kevin, Please send me screensh... 3:31pm (7 minutes ago)  
★ Kevin Fox to Jason More options 3:37pm (1 minute ago)

Don't worry, I'm sure nobody'll find out that you're using work resources to serve these images from [shellen.com](#). Just in case they cut you off though, I'll post the screenshots on my blog at [fury.com](#) as well.

Seeya later,  
Kevin  
[fury.com](#)

- Show quoted text -

Reply Forward

Open in new window  
Print conversation  
Expand all

Related Pages  
Mac OS X Hosting - Apache Hosting - Macintosh Web Site Hosting ...  
Offers affordable Mac OS X Apache web server unix hosting - BSD unix ...  
www.inno-tech.com  
Rivercom - Web Hosting - Web Development - Multimedia Development ...  
We combine original graphic design, intuitive navigational controls, ...  
www.rivercom.com

About these links

« Back to Inbox Archive More actions... 1 of 25 Older »

You are currently using 0 MB (0%) of your 1000 MB.  
Shortcuts: o-open y-archive c-compose j-older k-newer [more](#) »

[Terms of Use](#) - [Privacy Policy](#) - [Program Policies](#) - [Google Home](#)  
©2004 Google

Google

Gmail ▾

COMPOSE

Primary Social 3 new Promotions 2 new Updates 2 new

Inbox (7)

Starred  
Drafts  
Sent Mail

Search people...

- Jenny Kang
- Peter H
- Jonathan Pelleg
- Brett C
- Max Stein
- Jen Hart
- Eric Lowery

Google+ new You were tagged in 3 photos on Google+ - Google+ You were tagged in three pl

YouTube new LauraBlack just uploaded a video. - Jess, have you seen the video LauraBlack u

Emily Million (Google+) new [Knitting Club] Are we knitting tonight? - [Knitting Club] Are we knitting tonight?

Sean Smith (Google+) Photos of the new pup - Sean Smith shared an album with you. View album be tho

Google+ Kate Baynham shared a post with you - Follow and share with Kate by adding her

Google+ Danielle Hoodhood added you on Google+ - Follow and share with Danielle by

YouTube Just for You From YouTube: Daily Update - Jun 19, 2013 - Check out the latest

Google+ You were tagged in 3 photos on Google+ - Google+ You were tagged in three pho

Hilary Jacobs (Google+) Check out photos of my new apt - Hilary Jacobs shared an album with you. View

Google+ Kate Baynham added you on Google+ - Follow and share with Kate by adding her

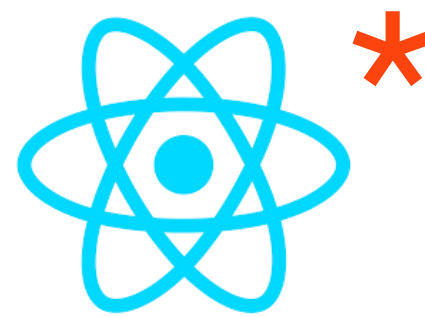
# Introdução

## MVC -> MVVM

- O suporte ao AJAX (2005) permitiu a atualização parcial de aplicações web
  - Requisições a página completa deixaram de ser necessárias
  - As atualizações passaram a ser mais rápidas
  - No entanto, algum esforço duplicado era necessário com o espelhamento da lógica de apresentação e a lógica de negócio
- Por volta de 2010 os primeiros framework focados em MVVM surgiram



- Atualmente os frameworks mais utilizados utilizam a arquitetura MVVM



# Introdução

## MVC -> MVVM

- MVVM (Model View ViewModel) é um padrão arquitetural baseado no MVC
  - Foca em separar mais claramente a UI da lógica de negócio da aplicação (*model*)
- Diversas implementações desse padrão utilizam *declarative data binding* para separar o a implementação das *Views* de outra camadas
- Foi projetado para "remover" todo o código relacionado UI por meio do uso de *data binding*
- A ideia é obter ambas as vantagens da separação do desenvolvimento funcional fornecido pelo MVC, enquanto aproveita as vantagens do *data binding*



# Introdução

## MVC -> MVVM

### MVVM - View

- Responsável por mostrar as informações para o usuário
- Representam o estado atual da aplicação fornecido pelo **ViewModel**
- São ativas já que possuem data-bindings, eventos and comportamentos que requerem conhecimento do **Model** e do **ViewModel**
- São responsáveis por tratar os eventos do usuário e enviá-los ao **ViewModel**
- É importante notar que a **View** não é responsável por manter o estado da aplicação

# Introdução

## MVC -> MVVM

### MVVM - ViewModel

- Pode ser considerado como um **Controller** especializado que age como um conversor de dados
  - Transforma as informações do **Model** para **View** passando comandos da View para o Model
- Armazena o estado atual da aplicação e gerencia boa parte da lógica de visualização
  - Mantém a sincronia do estado da aplicação entre com **View** por meio do **data binding (Binder)**

# Introdução

## MVC -> MVVM

### MVVM - Model

- Como em outros padrões da família **MV\***, o **Model** representa os dados específicos do domínio da aplicação
- Armazenam informações, mas não tratam o comportamento (lógica de negócio)
- É a camada responsável por persistir os dados da aplicação

# Introdução

## MVC -> MVVM

Uma view é responsável por mostrar as informações para o usuário

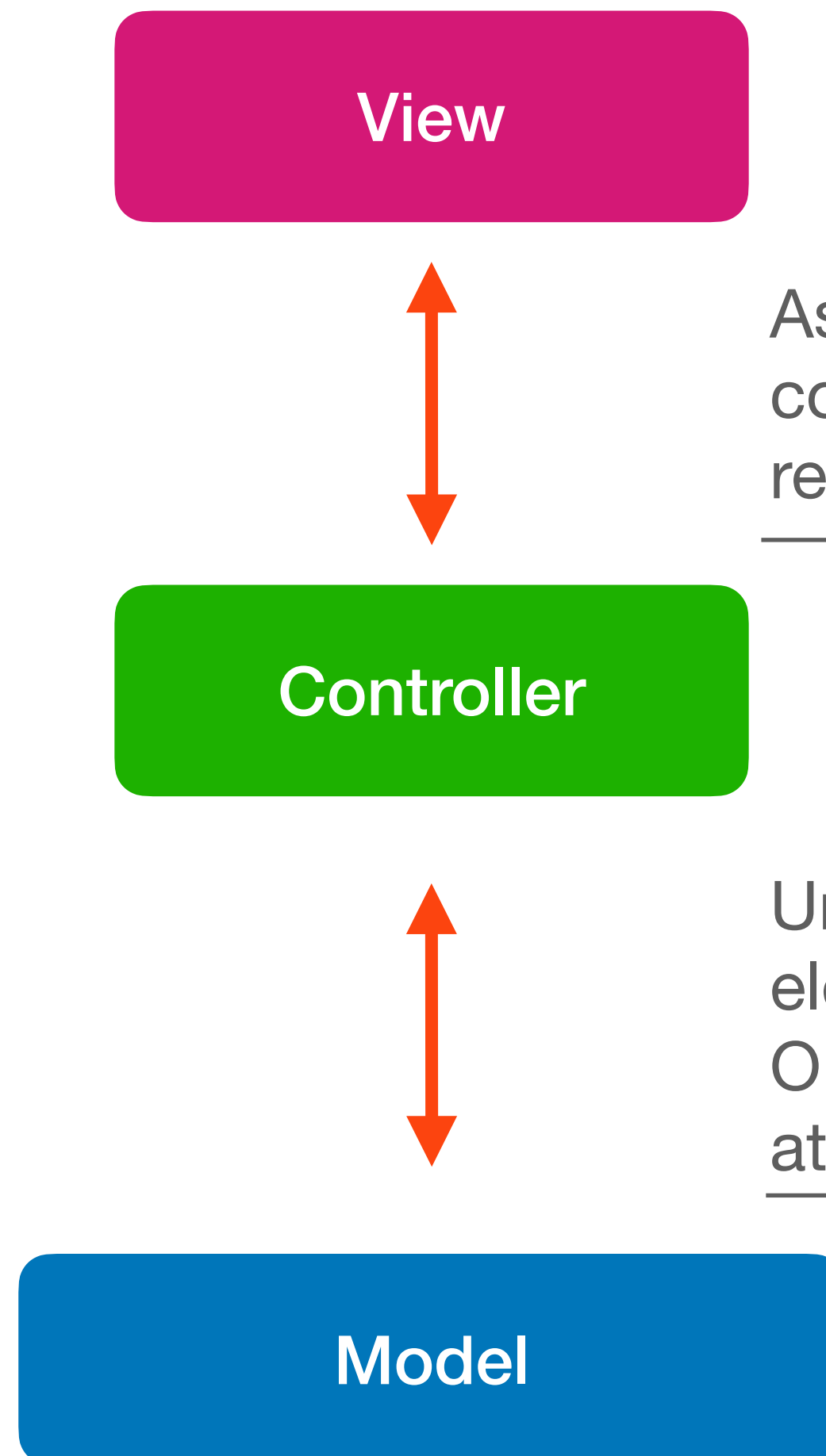
---

O controller age como um mediator. Ele ajuda na aquisição e transformação de dados a partir do model e as propaga para a View. Além disso, ele roteia os eventos ocorridos na View até o Model.

---

Em uma aplicação MVC, o modelo é apresentado pelo domínio dela e sua regra de negócio.

---



As ações do usuário nas Views são enviado ao controller. Ao final do processo de responder a requisição, a resposta é enviada para um nova view.

---

Uma vez que o Controller processa a informação, ele as envia ao modelo para que sejam persistidas. O modelo informa ao Controller quando essa atividade termina.

---

# Introdução

## MVC -> MVVM

Apresenta o estado atual da aplicação fornecido pelo ViewModel e lidam com eventos do usuário

ViewModel não apenas expõe atributos dos modelos, mas também dão acesso a métodos e funcionalidades como validação. Armazena o estado atual da aplicação e gerencia boa parte da lógica de visualização.

Model armazenam informações, mas não tratam o comportamento (lógica de negócio).



Views e ViewModel se comunicam via data-binding e eventos. O binder expõe os dados para View em forma de propriedades. Models e propriedades no ViewModel são sincronizados e atualizados via two-way data-binding.

O ViewModel, assim com o Controller, é o responsável enviar as informações que devem ser persistidas pelo Model.

# Introdução

## MVVM - Vantagens

- Facilita o desenvolvimento em paralelo da UI e de seus components
- Abstrai o funcionamento da View, reduzindo a quantidade da lógica de negócio na camada de View
- O **ViewModel** é mais fácil de ser testado com testes unitários se comparado com *event-driven code*
- O **ViewModel** pode ser testado sem preocupações com automatização da UI e interações
- Da um melhor suporte a aplicações **reativas**

# Introdução

## Aplicações Reativas

### Aplicações Reativas

- Não são um paradigma ou uma nova ideia
- Sua adoção no contexto web está intimamente ligada a framework JS como: Vue, React e Angular
- De forma simplificada podemos dizer que uma aplicação reativa:
  - **Observa** as modificações do estado da aplicação
  - **Propaga/Notifica** as mudanças em toda a aplicação
  - **Atualiza/Renderiza** as views automaticamente em resposta a mudanças
  - Forneça feedback oportuno para as interações do usuário

# Introdução ao VueJS





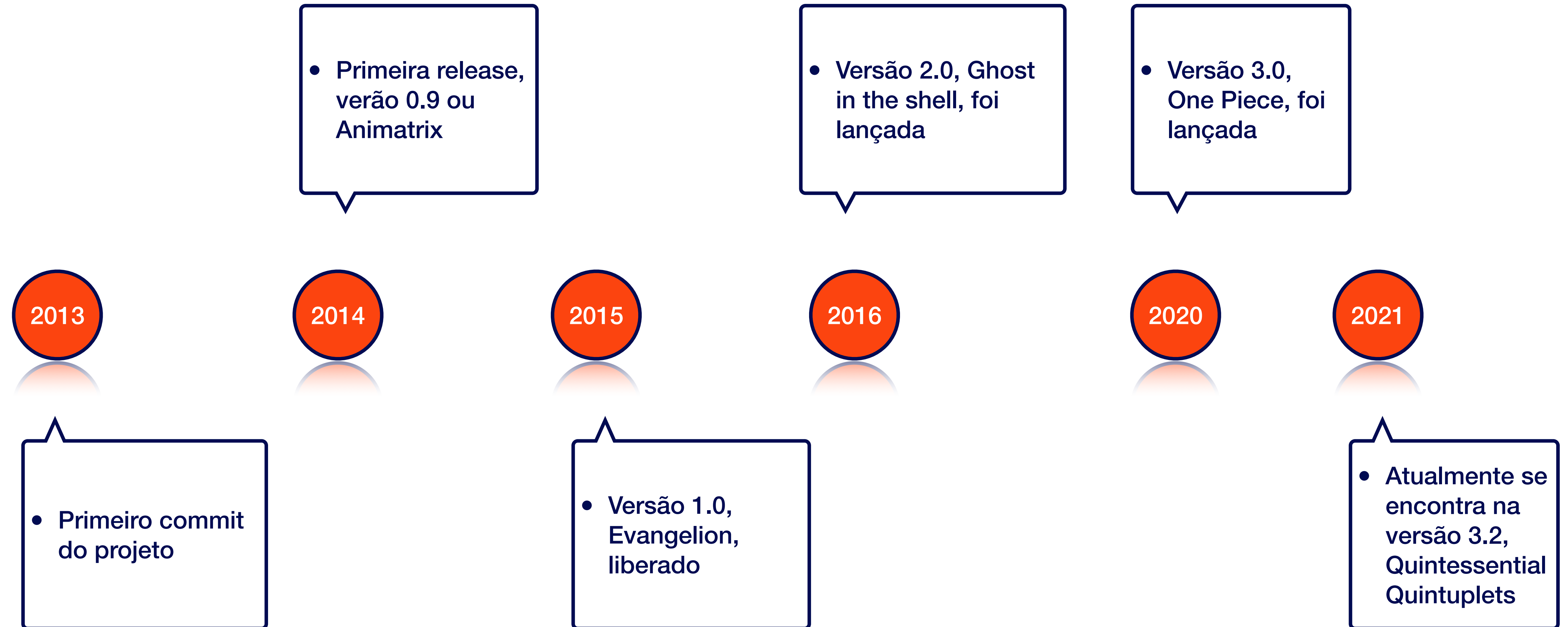
# Introdução ao VueJS

## VueJs

- Comumente conhecido como Vue, pronunciado “view”
- Framework **progressivo** do JavaScript de código aberto (*open source*) para a construção de interfaces de usuário
  - Projetado para ser adotado de forma incremental
- Também pode funcionar como uma estrutura de aplicativos web capaz de alimentar aplicativos avançados de uma única página
- Criado por **Evan You** depois de trabalhar para o **Google** no **AngularJS**

# Introdução

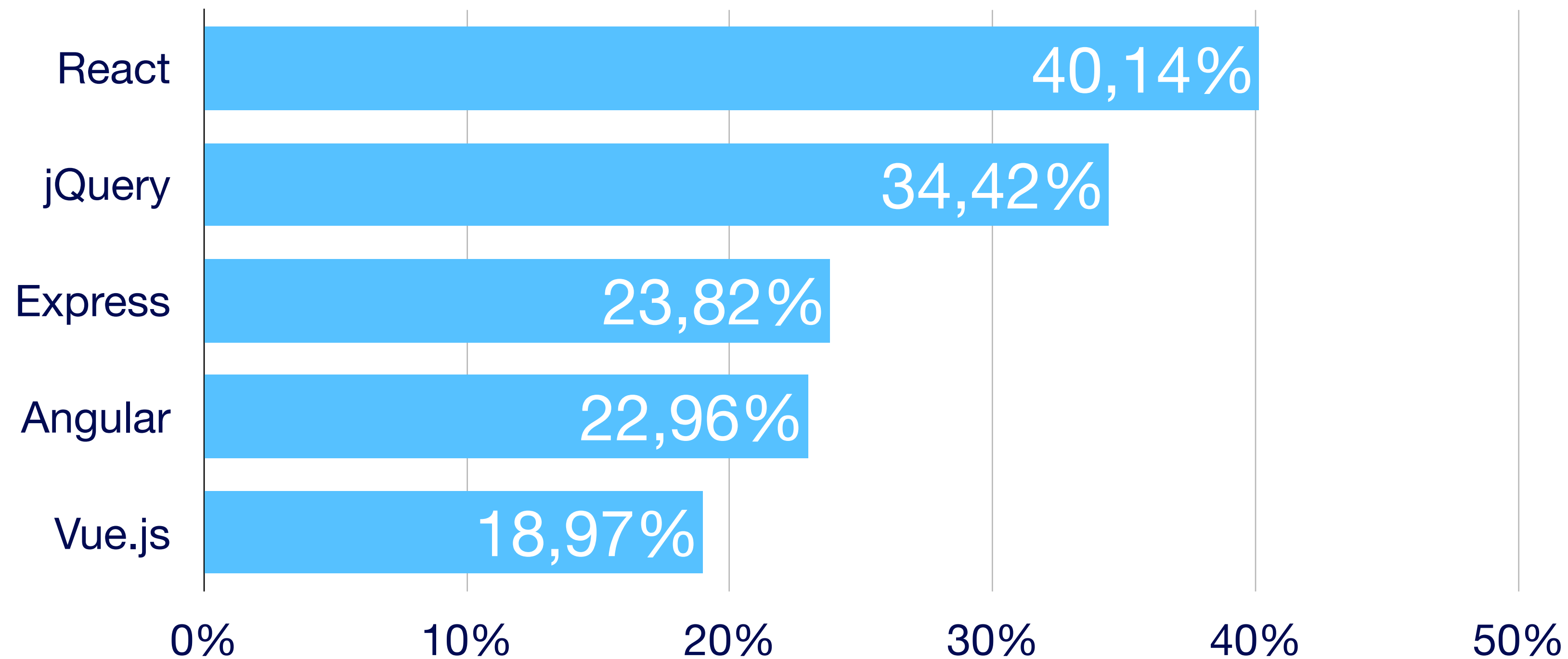
## História - Timeline



# Introdução ao VueJS

## Comunidade

Frameworks web mais utilizados

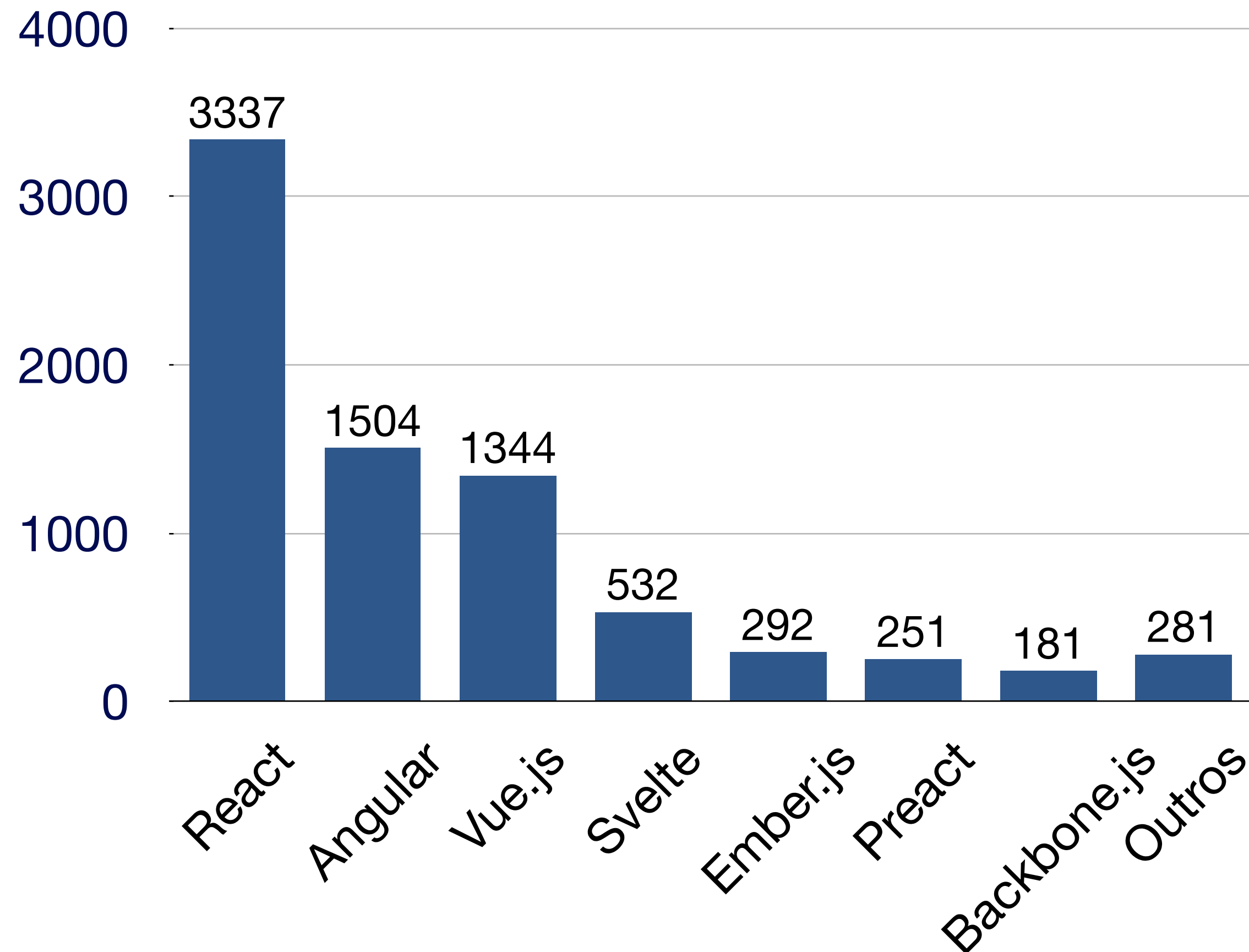


Fonte: <https://insights.stackoverflow.com/survey/2021#section-most-popular-technologies-web-frameworks>

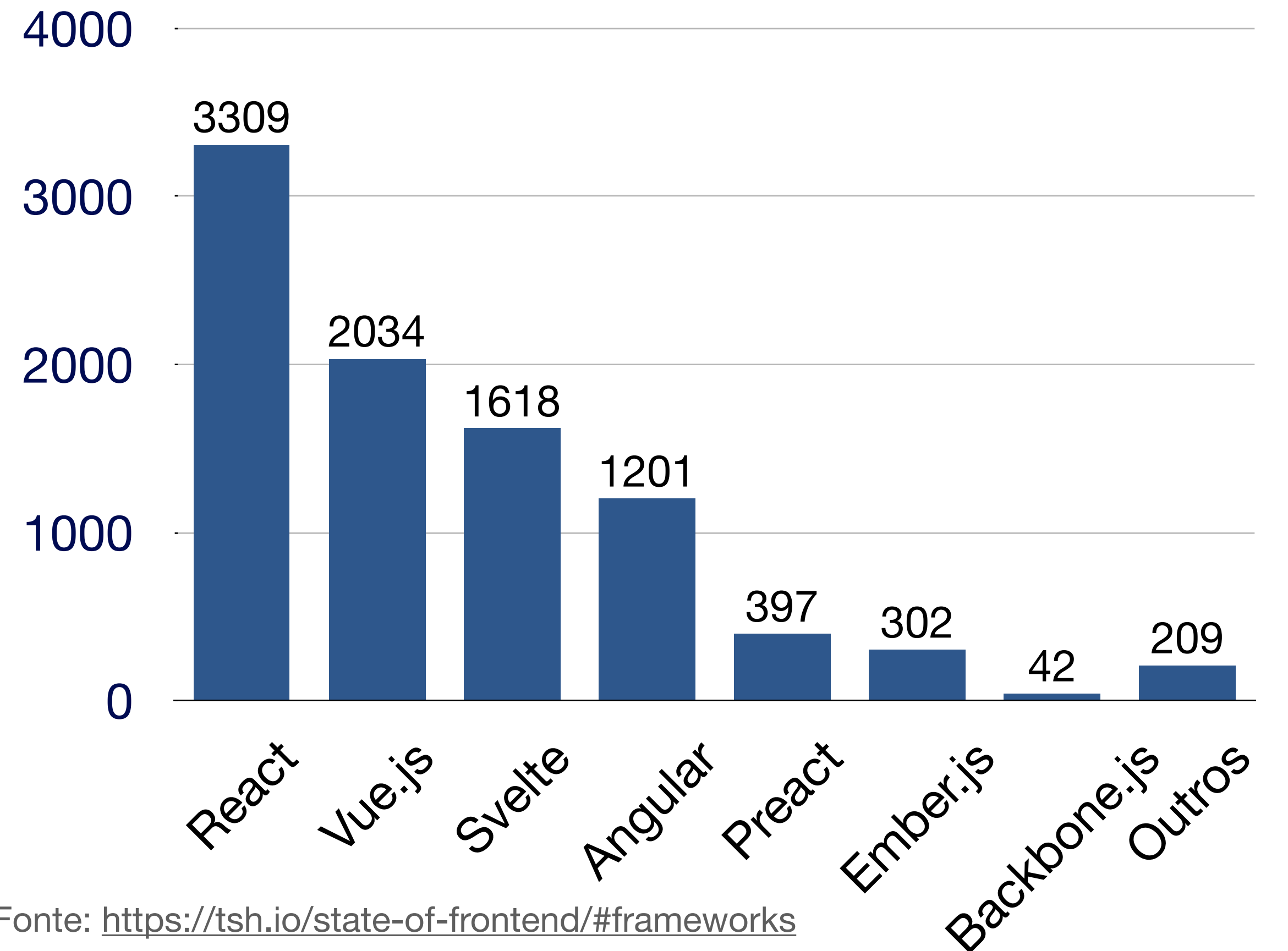
# Introdução ao VueJS

## Comunidade

Quais frameworks você utilizou no último ano?



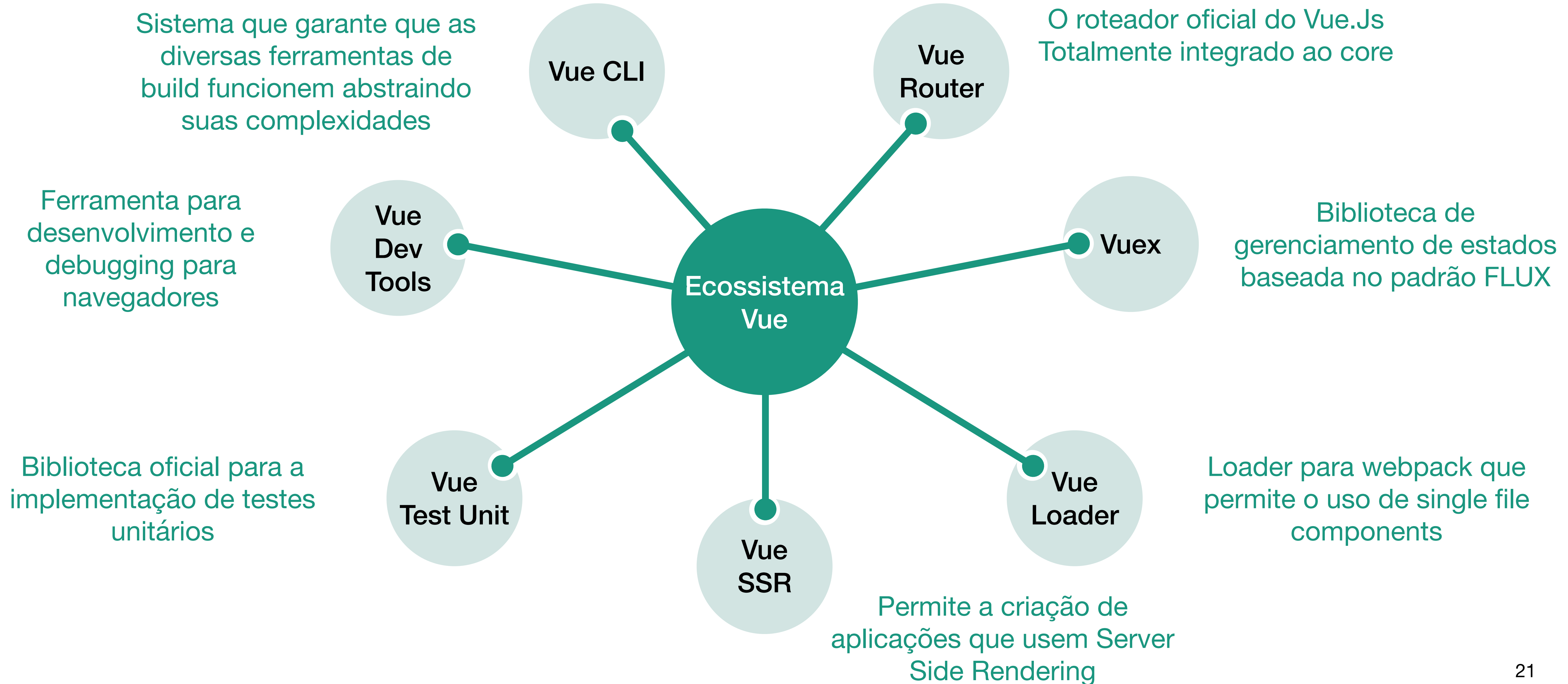
Qual framework você gostaria de continuar utilizando no futuro?



Fonte: <https://tsh.io/state-of-frontend/#frameworks>

# Introdução ao VueJS

## Ecosystema



# Introdução ao VueJS

## Vantagens

- Leve
  - Pouco mais de 18KB, consideravelmente menor que os concorrentes
- Performance e Virtual DOM
- Data binding reativo de duas mãos (*two way*)
- Legibilidade
  - Single File Components incentivam a separação em componentes com seus respectivos HTML, CSS e JS/TS

# Introdução ao VueJS

## Vantagens

- Ecossistema bem estabelecido
- Flexível
- Documentação concisa e atualizada
- Suporte da comunidade
- Fácil de usar

# Principais aspectos de uma aplicação em VueJs





# Principais aspectos de uma aplicação em VueJs

## Criando uma aplicação em Vue

```
const app = Vue.createApp({  
  /* options */  
}).mount('#app')
```

O pontapé inicial de toda aplicação em Vue é a criação de uma instância do objeto *application*

Recebe como primeiro parâmetro um objeto usado para configurar o componente raiz (root)

Uma aplicação precisa ser montando em um elemento da DOM

# Principais aspectos de uma aplicação em VueJs

## Criando uma aplicação em Vue

```
const RootComponent = { /* options */  
Vue.createApp(RootComponent)  
  .component('SearchInput', SearchInputComponent)  
  .directive('focus', FocusDirective)  
  .use(LocalePlugin)  
  .mount('#app')
```

O componente raiz é o ponto inicial de renderização após a montagem da aplicação

A instância de application é utilizada para registrar elementos “globais” que podem ser usados por outros componentes da aplicação

# Principais aspectos de uma aplicação em VueJs

## Criando uma aplicação em Vue

```
const app = Vue.createApp({
  data() {
    return { count: 4 }
  }
})
const vm = app.mount('#app')
console.log(vm.count) // => 4
```

- Cada componente pode expor dados por meio da função *data*
  - Deve retornar um objeto
  - Automaticamente são rastreados pelo sistema de reatividade do Vue
- Outros tipos de propriedades podem ser definidas: *methods*, *props*, *computed* ...

# Principais aspectos de uma aplicação em VueJs

## Data binding, interpolação e Diretivas

```
const app = Vue.createApp({
  data() {
    return { count: 4 }
  }
})
const vm = app.mount('#app')
vm.count++
```

```
<span>Message: {{ count }}</span>
```

Message: 4

- A forma mais simples de data binding é estabelecida usando um ‘bigode’ `{{ }}`
  - Também chamada de interpolação
  - Vincula o dado com o texto do elemento HTML
  - Não é aplicável em atributos HTML, para isso diretivas são utilizadas

# Principais aspectos de uma aplicação em VueJs

## Data binding, interpolação e Diretivas

```
const app = Vue.createApp({
  data() {
    return { seen: true }
  }
})
const vm = app.mount('#app')
vm.seen = false
```

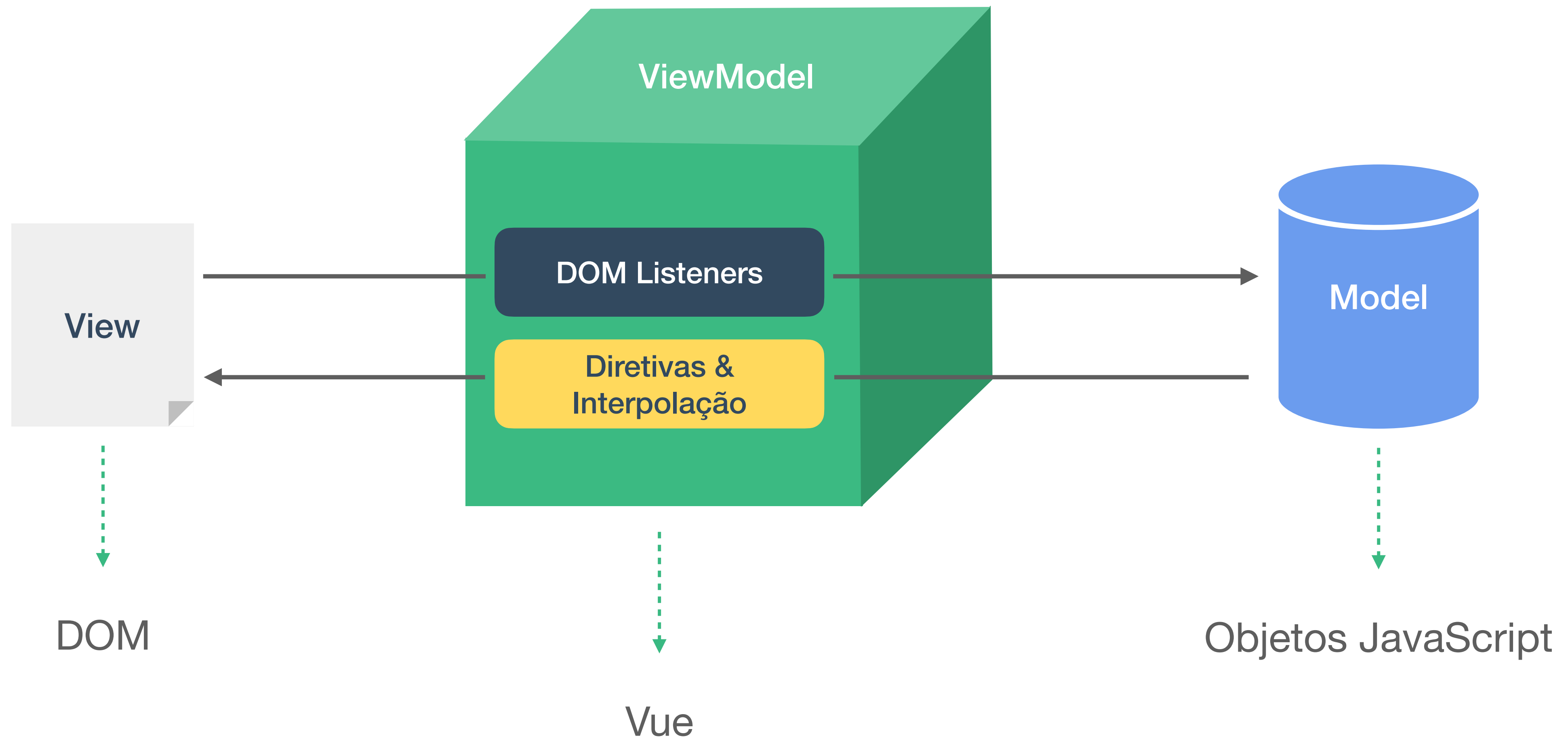
```
<p v-if="seen">Now you see me</p>
```

Now you see me

- **Diretivas** são atributos especiais começados com o prefixo **v-**
  - Utilizadas para realizar a vinculação de atributos
  - Tem como objetivo aplicar reativamente os efeitos colaterais da mudança dos valores de suas expressões na DOM

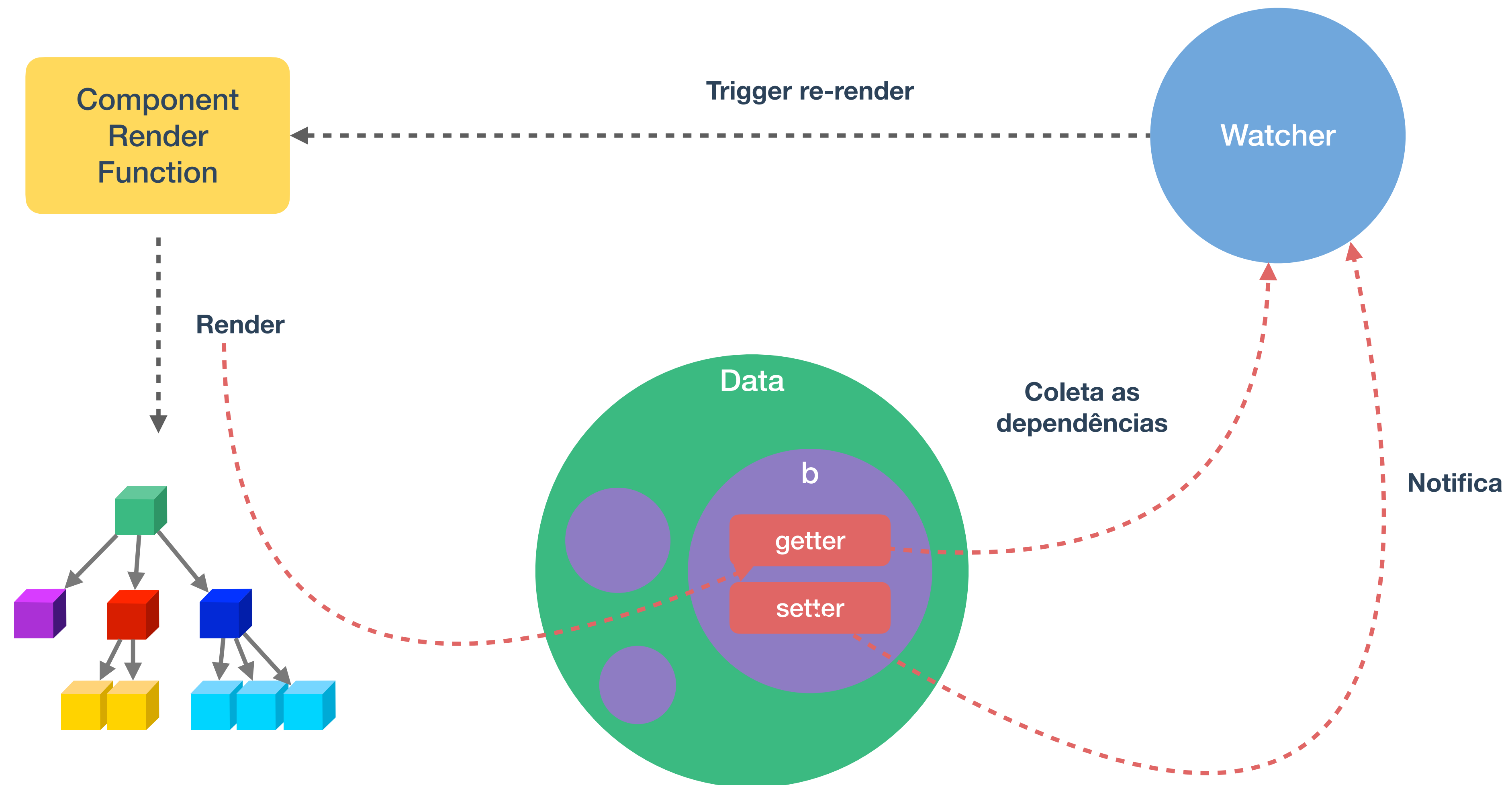
# Principais aspectos de uma aplicação em VueJs

## Data binding



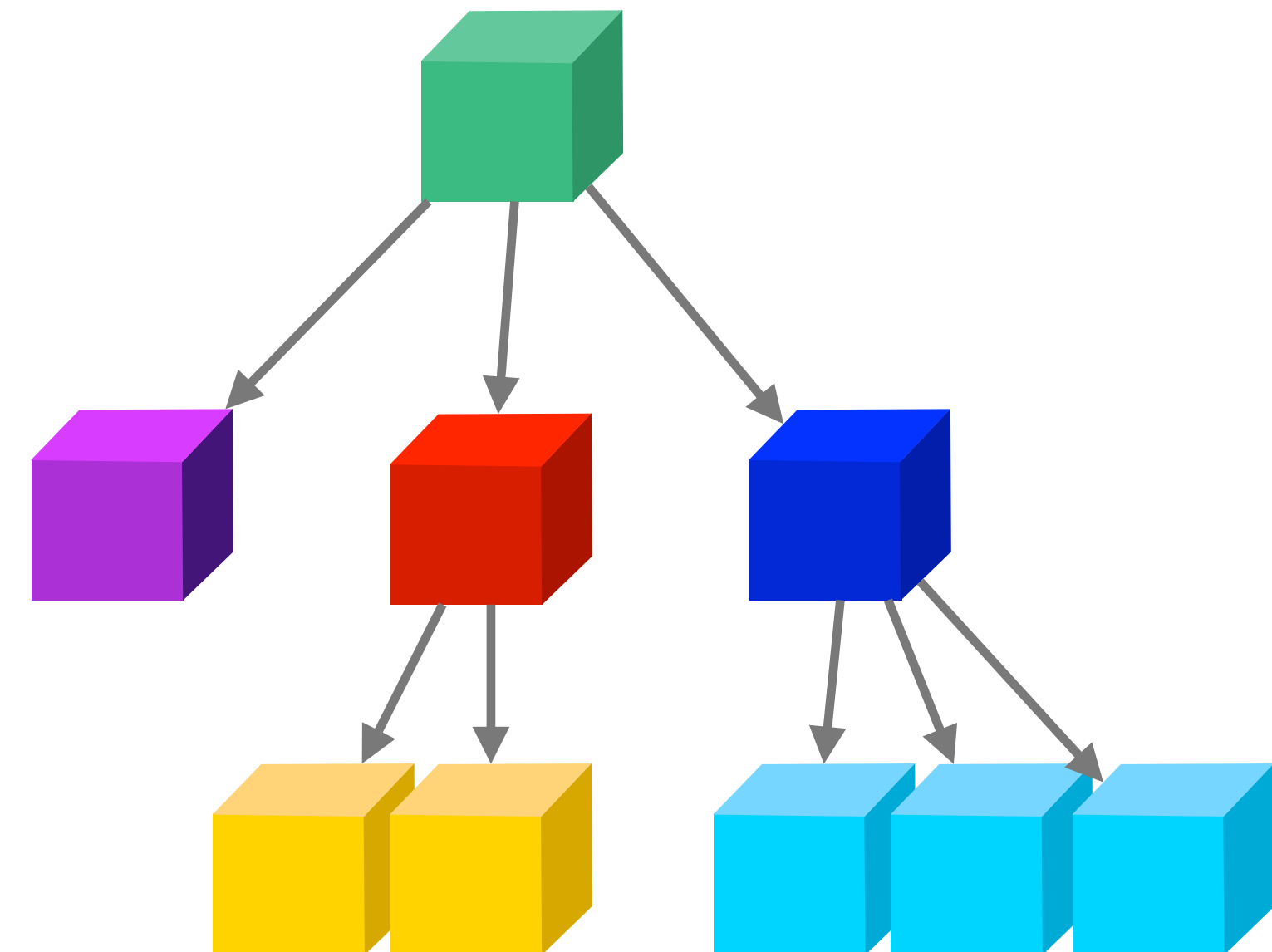
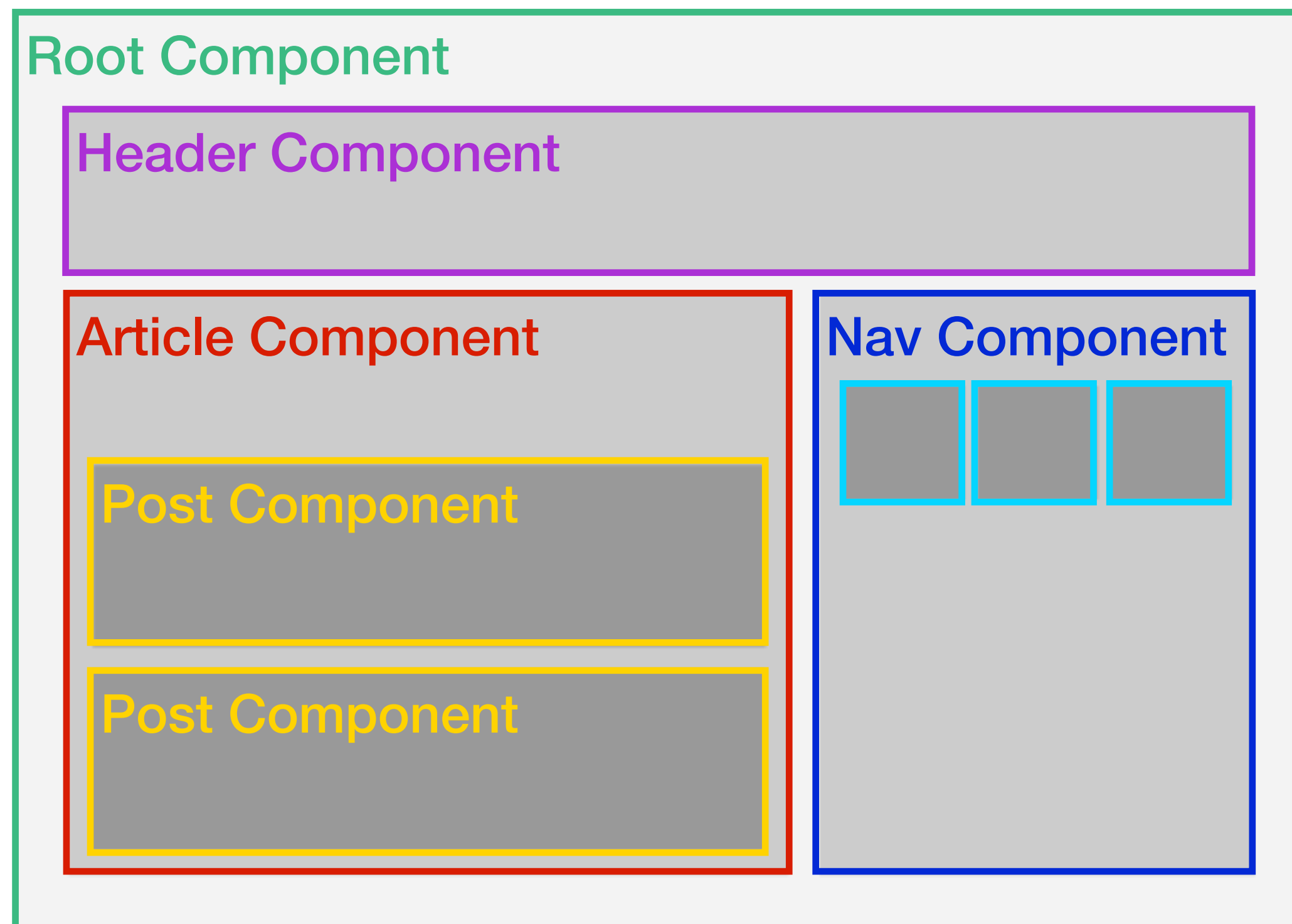
# Principais aspectos de uma aplicação em VueJs

## Data binding



# Principais aspectos de uma aplicação em VueJs

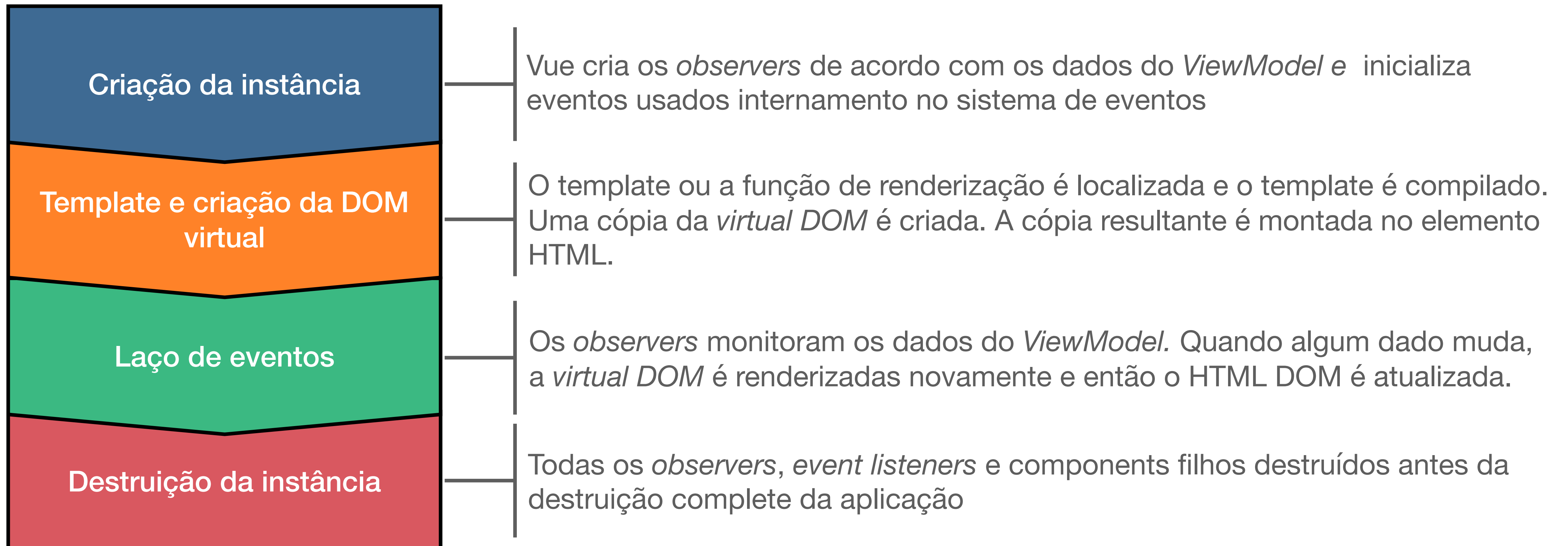
## Componentes





# Principais aspectos de uma aplicação em VueJs

## Ciclo de Vida



# Principais aspectos de uma aplicação em VueJs

## Ciclo de Vida

```
app = Vue.createApp(options)  
app.mount(el)
```

# Diretivas



# Diretivas

## Argumentos

- Diretivas são atributos especiais começados com o prefixo **v-**
- Algumas diretivas aceitam um argumento
  - Após o nome da diretiva e um sinal de :
  - Por exemplo: v-bind é utilizado para atualizar de forma reativa um atributo HTML

```
<a v-bind:href="url"> ... </a>  
<a v-on:click="doSomething"> ... </a>  
<a v-bind:[attributeName]="url"> ... </a>
```

# Diretivas

## Vinculação de classes (CSS)

```
Vue.createApp({
  data() {
    return {
      isActive: true,
      hasError: false
    }
  }
}).mount('#array-rendering')
```

```
<div :class="{ active: isActive }"></div>

<div
  class="static"
  :class="{
    active: isActive,
    'text-danger': hasError
  }"
></div>
```

# Diretivas

## Vinculação de classes (CSS)

```
Vue.createApp({  
  data() {  
    return {  
      activeClass: 'active',  
      errorClass: 'text-danger'  
    }  
  })  
).mount('#array-rendering')
```

```
<div :class="[activeClass, errorClass]"></div>
```

# Diretivas

## Renderização condicional

```
<h1 v-if="awesome">Vue is awesome!</h1>  
<h1 v-else>Oh no 😞</h1>
```

```
<template v-if="ok">  
  <h1>Title</h1>  
  <p>Paragraph 1</p>  
  <p>Paragraph 2</p>  
</template>
```

```
<div v-if="type === 'A'">  
  A  
</div>  
<div v-else-if="type === 'B'">  
  B  
</div>  
<div v-else-if="type === 'C'">  
  C  
</div>  
<div v-else>  
  Not A/B/C  
</div>
```

# Diretivas

## Renderização de listas

```
Vue.createApp({
  data() {
    return {
      items: [{ message: 'Foo' }, { message: 'Bar' }]
    }
  }
}).mount('#array-rendering')
```

```
<ul id="array-rendering">
  <li v-for="item in items">
    {{ item.message }}
  </li>
</ul>
```



# Diretivas

## Renderização de listas

```
Vue.createApp({
  data() {
    return {
      items: [{ message: 'Foo' }, { message: 'Bar' }]
    }
  }
}).mount('#array-rendering')
```

```
<ul id="array-with-index">
  <li v-for="(item, index) in items">
    {{ parentMessage }} - {{ index }} - {{ item.message }}
  </li>
</ul>
```

# Diretivas

## Tratando eventos

```
Vue.createApp({
  data() {
    return {
      counter: 0
    }
  }
}).mount('#basic-event')
```

```
<div id="basic-event">
  <button @click="counter += 1">Add 1</button>
  <p>The button above has been clicked {{ counter }} times.</p>
</div>
```

# Diretivas

## Tratando eventos

```
Vue.createApp({
  data() {
    return { name: 'Vue.js' }
  },
  methods: {
    greet(event) {
      alert('Hello ' + this.name + '!')
      if (event) { alert(event.target.tagName) }
    }
  }
}).mount('#event-with-method')
```

```
<div id="event-with-method">
  <button v-on:click="greet">Greet</button>
</div>
```

# Diretivas

## Tratando eventos

```
Vue.createApp({
  methods: {
    say(message) {
      alert(message)
    }
  }
}).mount('#inline-handler')
```

```
<div id="inline-handler">
  <button v-on:click="say('hi')">Say hi</button>
  <button v-on:click="say('what')">Say what</button>
</div>
```

# Diretivas

## Vinculação de formulários

- Utilizamos a diretiva *v-model* para criar *two data binding* com elementos do *input*, *textarea* e *select*
  - Atualiza corretamente o elemento baseado no tipo de *input*
  - O valor inicial deve ser declarado no objeto retornado pelo método *data*
    - *text* e *textarea* usam a propriedade *value* e o evento *input*
    - *checkboxes* e *radiobuttons* usam a propriedade *checked* e o evento *change*
    - *select* usam a propriedade *value* e o evento *change*

# Diretivas

## Vinculação de formulários: Input e Textarea

```
<input v-model="message" placeholder="edit me" />  
<p>Message is: {{ message }}</p>
```

```
<span>Multiline message is:</span>  
<p style="white-space: pre-line;">{{ message }}</p>  
<br />  
<textarea v-model="message" placeholder="add multiple lines"></textarea>
```

# Diretivas

## Vinculação de formulários: checkbox

```
<input type="checkbox" id="checkbox" v-model="checked" />
<label for="checkbox">{{ checked }}</label>
```

```
<div id="v-model-multiple-checkboxes">
  <input type="checkbox" id="jack" value="Jack" v-model="checkedNames" />
  <label for="jack">Jack</label>
  <input type="checkbox" id="john" value="John" v-model="checkedNames" />
  <label for="john">John</label>
  <input type="checkbox" id="mike" value="Mike" v-model="checkedNames" />
  <label for="mike">Mike</label>
  <br>
  <span>Checked names: {{ checkedNames }}</span>
</div>
```

# Diretivas

## Vinculação de formulários: radiobutton

```
<div id="v-model-radiobutton">
  <input type="radio" id="one" value="One" v-model="picked" />
  <label for="one">One</label>
  <br />
  <input type="radio" id="two" value="Two" v-model="picked" />
  <label for="two">Two</label>
  <br>
  <span>Picked: {{ picked }}</span>
</div>
```

```
Vue.createApp({
  data() {
    return {
      picked: ''
    }
  }
}).mount('#v-model-radiobutton')
```



# Diretivas

## Vinculação de formulários: select

```
<div id="v-model-select" class="demo">
  <select v-model="selected">
    <option disabled value="">Please select one</option>
    <option>A</option>
    <option>B</option>
    <option>C</option>
  </select>
  <span>Selected: {{ selected }}</span>
</div>
```

```
Vue.createApp({
  data() {
    return {
      selected: ''
    }
  }
}).mount('#v-model-select')
```

# Diretivas

## Atalhos

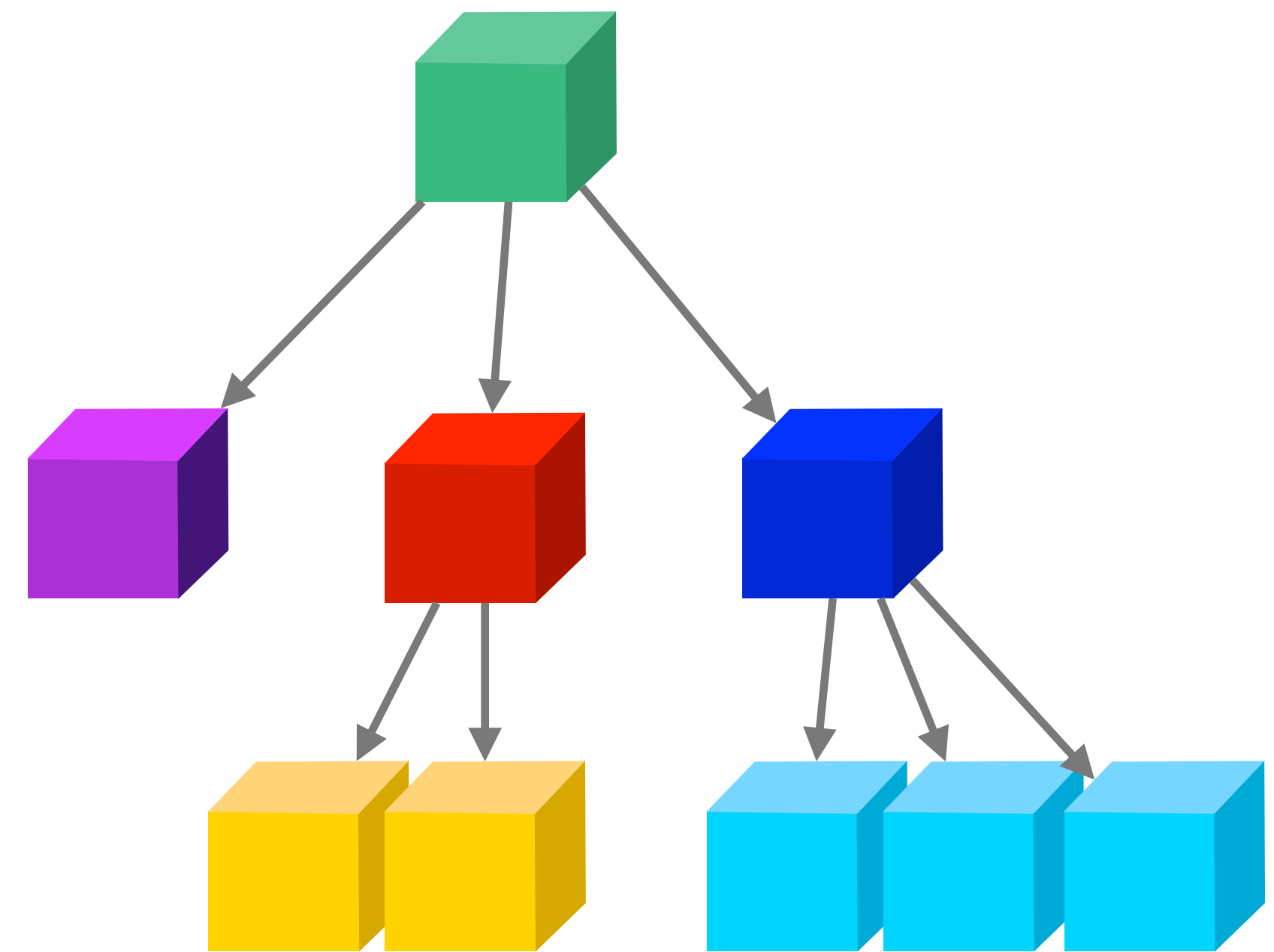
### Atalho para v-bind

```
<!-- full syntax -->  
<a v-bind:href="url"> ... </a>  
<!-- shorthand -->  
<a :href="url"> ... </a>  
<!-- shorthand with dynamic argument -->  
<a :[key]="url"> ... </a>
```

### Atalho para v-on

```
<!-- full syntax -->  
<a v-on:click="doSomething"> ... </a>  
<!-- shorthand -->  
<a @click="doSomething"> ... </a>  
<!-- shorthand with dynamic argument -->  
<a @[event]="doSomething"> ... </a>
```

# Componentes



# Componentes

## Componentes

- Assim como outros frameworks modernos, Vue permite que os usuários criem componentes isolados em suas aplicações
- São importantes dentre outros motivos por favorecer a **reusabilidade** e a **manutenabilidade**
- São auto-contidos agrupando HTML, JS e CSS
  - Facilitando a manutenção especialmente quando a aplicação escala

# Componentes

```
const app = Vue.createApp({})
app.component('button-counter', {
  data() {
    return {
      count: 0
    }
  },
  template: `
    <button @click="count++">
      You clicked me {{ count }} times.
    </button>`
})
```

Maneira padrão de criar um componente

Permite a definição do template do componente

```
<div id="components-demo">
  <button-counter></button-counter>
  <button-counter></button-counter>
</div>
```

Componente em uso

# Componentes

## Passando dados para um componente filho

### Props

- Permite o envio de dados aos components filhos
- São atributos customizáveis registrados por um componente
  - Dever ser explicitamente declarados no componente filho
  - Um valor deve ser dado pelo componente pai/mãe
- São unidirecionais, sempre no sentido pai/mãe -> filho

# Componentes

## Passando dados para um componente filho


```
const App = {
  data() {
    return {
      posts: [
        { id: 1, title: 'My journey with Vue' },
        { id: 2, title: 'Blogging with Vue' },
        { id: 3, title: 'Why Vue is so fun' }
      ]
    }
  }
}

const app = Vue.createApp(App)

app.component('blog-post', {
  props: ['title'],
  template: `<h4>{{ title }}</h4>`
})

app.mount('#blog-posts-demo')
```

```
<div id="blog-posts-demo">
  <blog-post
    v-for="post in posts"
    :key="post.id"
    :title="post.title"
  ></blog-post>
</div>
```



**My journey with Vue**  
**Blogging with Vue**  
**Why Vue is so fun**

# Componentes

## Escutando eventos do componentes filhos

### Eventos

- Em diversas situações é necessário que haja comunicação entre filho e pai
- Essa comunicação é feita por meio de eventos customizados
  - São iniciados quando um componente executa a instrução `$emit('nome do evento')`
  - Um componente que está escutando pelo é evento é notificado na instrução `$on('nome do evento')`
  - Dados podem ser enviados



# Componentes

## Escutando eventos do componentes filhos

```
const app = Vue.createApp({
  data() {
    return {
      posts: [
        { id: 1, title: 'My journey with Vue' },
        { id: 2, title: 'Blogging with Vue' },
        { id: 3, title: 'Why Vue is so fun' }
      ],
      postFontSize: 1
    }
  }
})

app.component('blog-post', {
  props: ['title'],
  emits: ['enlargeText'],
  template: `
    <div class="blog-post">
      <h4>{{ title }}</h4>
      <button @click="$emit('enlargeText')">
        Enlarge text
      </button>
    </div>
  `
})
```

```
<div id="blog-posts-events-demo" class="demo">
  <div :style="{ fontSize: postFontSize + 'em' }">
    <blog-post
      v-for="post in posts"
      :key="post.id"
      :title="post.title"
      @enlarge-text="postFontSize += 0.1"
    ></blog-post>
  </div>
</div>
```

# Componentes

## Escutando eventos do componentes filhos

```
const app = Vue.createApp({
  data() {
    return {
      posts: [ ... ],
      postFontSize: 1
    }
  },
  methods: {
    onEnlargeText(enlargeAmount) {
      this.postFontSize += enlargeAmount
    }
  }
})
app.component('blog-post', {
  props: ['title'],
  emits: ['enlargeText'],
  template: `
    <div class="blog-post">
      <h4>{{ title }}</h4>
      <button @click="$emit('enlargeText', 0.1)">
        Enlarge text
      </button>
    </div>
  `
})
```

```
<div id="blog-posts-events-demo" class="demo">
  <div :style="{ fontSize: postFontSize + 'em' }">
    <blog-post
      v-for="post in posts"
      :key="post.id"
      :title="post.title"
      @enlarge-text="onEnlargeText"
    ></blog-post>
  </div>
</div>
```

1. Ao clicar no botão o evento `enlargeText` é emitido com o valor `0.1` como argumento
2. O evento é tratado no `blog-post`
  - O valor `0.1`, é o **valor do parâmetro** recebido
3. o método `onEnlargeText` do **componente pai** é invocado
4. A propriedade do componente pai é atualizada
5. O `style` do `div` é atualizado como consequência



Valor enviado

# Componentes

## Distribuição de componentes com slots

### Slots

- Assim com elemento HTML, algumas vezes é útil passar o conteúdo para o componente da seguinte forma

```
<alert-box>  Nome do componente  
Content.  Conteúdo  
</alert-box>
```

- Isto pode ser feito ao utilizar o elemento `<slot>`

# Componentes

## Distribuição de componentes com slots

```
app.component('alert-box', {  
  template: `  
    <button class="btn-primary">  
      <slot></slot>  
    </button>  
  `,  
})
```

```
<todo-button>  
  Add todo  
</todo-button>  
  
<todo-button>  
  <i class="fas fa-plus"></i>  
  Add todo  
</todo-button>  
  
<todo-button>  
  <font-awesome-icon name="plus"></font-  
  awesome-icon>  
  Add todo  
</todo-button>
```

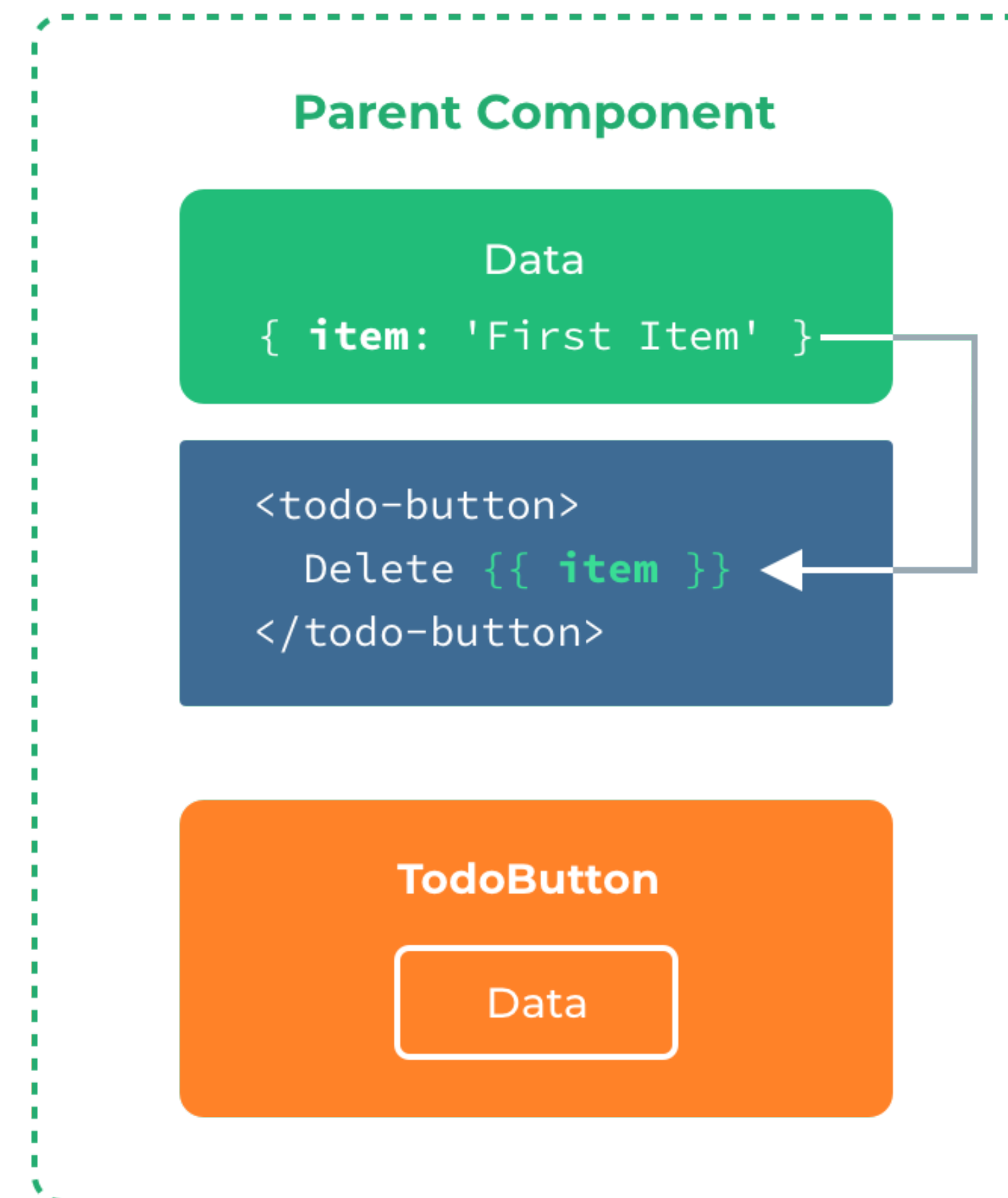
Outro componente

# Componentes

## Distribuição de componentes com slots

```
app.component('alert-box', {
  data() {
    return {
      item: 'First item'
    }
  }
  template: `
    <button class="btn-primary">
      <slot></slot>
    </button>
  `
})
```

```
<todo-button>
  Delete a {{ item }}
</todo-button>
```



Slots não tem acesso a dados do componente filho

# Prática



# Prática

Start Bootstrap Home About Shop Cart 0

## Root Component

### Shop in style

With this shop homepage template

### Product List Component

Produto Component

Incredible Metal Chicken  
★★★★  
R\$281  
Add to cart

Produto Component

Handmade Granite Bacon  
★★  
R\$166  
Add to cart

Produto Component

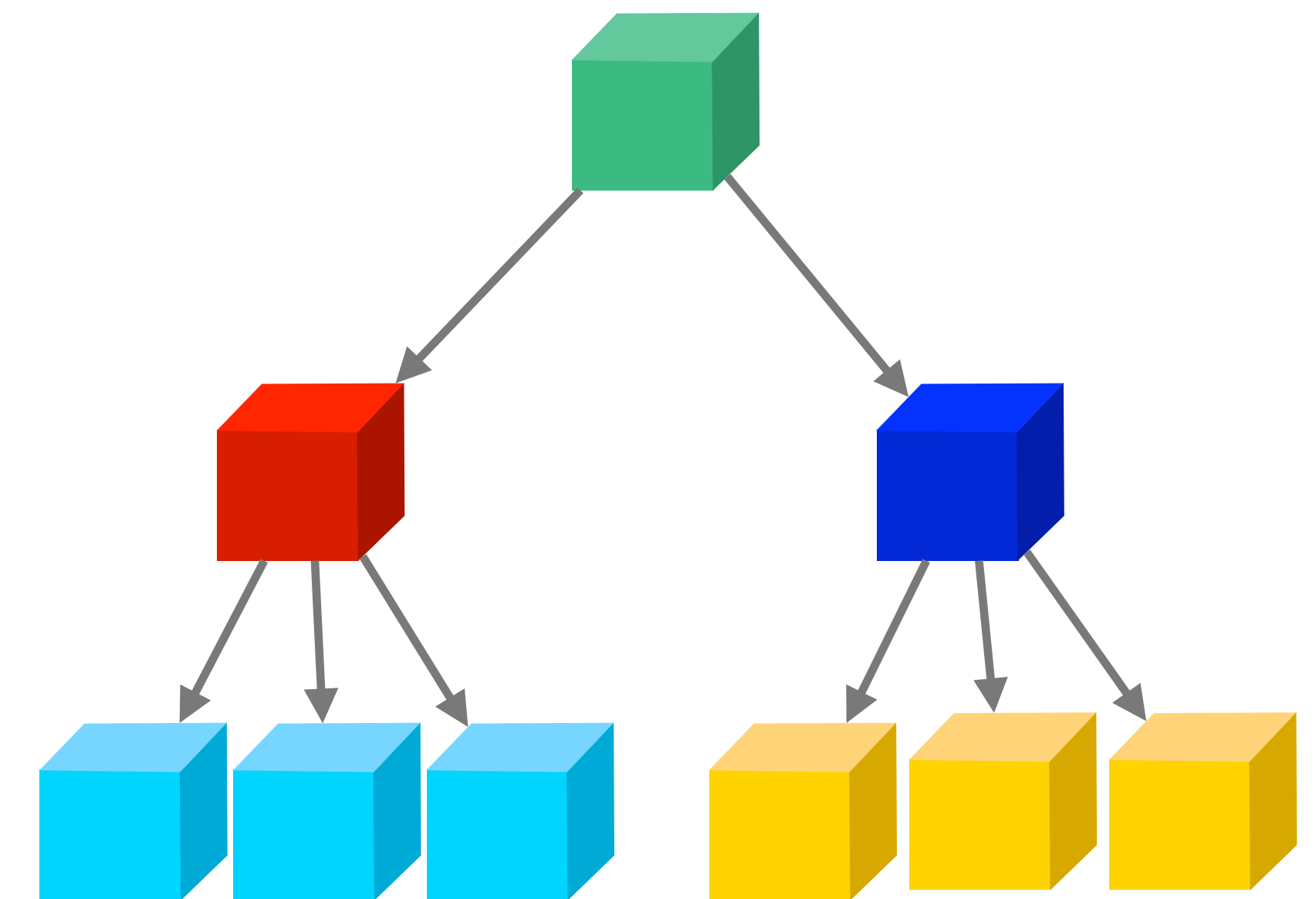
Sleek Concrete Ball  
★  
R\$394  
Add to cart

### Estoque

#	Name	Description	Qty	Price	Discount(%)	Action
1	Incredible Metal Chicken	Carbonite web goalkeeper gloves are ergonomically designed to give easy fit	3	281	0	Delete
2	Handmade Granite Bacon	The automobile layout consists of a front-engine design, with transaxle-type transmissions mounted at the rear of the engine and four wheel drive	3	166	0	Delete
3	Sleek Concrete Ball	New range of formal shirts are designed keeping you in mind. With fits and styling that will make you stand apart	3	394	0	Delete

### Stock Component

Item component



# Referências

- Vue.js in Action por Eric Hanchett com Benjamin Listwon
- Fullstack Vue: The Complete Guide to Vue.js por Hassan Djirdeh, Nate Murray e Ari Lerner
- [A brief history of web development](#)
- [FrontEnd Chronology](#)
- [Vue Presentation](#)
- [Understanding MVVM - A guide for JavaScript developers](#)





# Referências

- [MVVM - Learning JavaScript Design Patterns \[Book\]](#)
- [Vue.js : Documentação oficial](#)
- [The ultimate guid to Javascript frameworks](#)
- [JavaScript Technical Interview Question: is React a MVC or MVVM?](#)
- [VueJs OverView](#)

Por hoje é só