



UNIVERSIDADE  
FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

# Fundamentos de Vue.js

**QXD0279 - Desenvolvimento de Software para Web 2**

**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



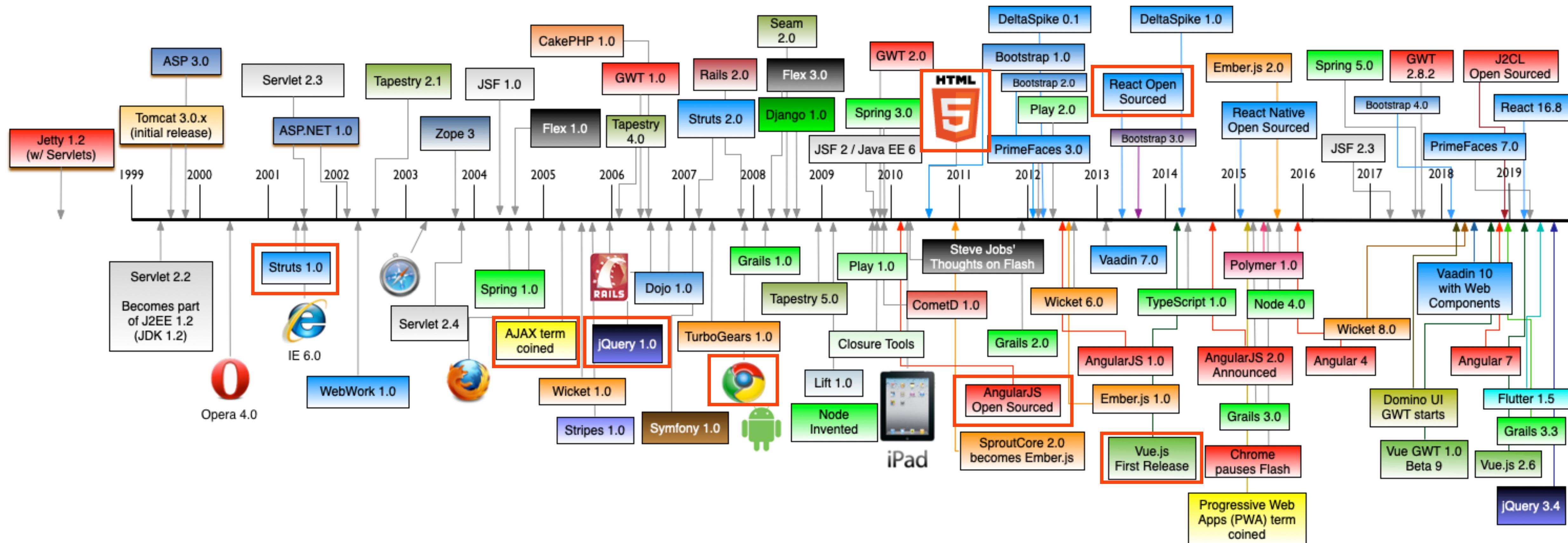
# Introdução

## 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 lançadass 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

The screenshot shows the original Gmail interface from 2004. At the top, it says "Gmail by Google BETA". Below the header are search bars for "Search Mail" and "Search the Web", and links for "Report bug", "Contacts", "Settings", "Help", and "Sign out". A "Compose Mail" button is on the left. The main area shows an inbox with two messages: one from Jason Shellen and one from Kevin Fox. The message from Kevin Fox contains a screenshot and a link to fury.com. On the left sidebar, there are links for "Starred", "Sent Mail", "All Mail", "Spam", "Trash", and a "Labels" section which includes "quirk", "Orkut", and "Prius (18)". A green box highlights the "Prius (18)" label. At the bottom, it says "You are currently using 0 MB (0%) of your 1000 MB." and provides keyboard shortcuts: o - open, y - archive, c - compose, j - older, k - newer, and a link to "more". It also includes links for "Terms of Use", "Privacy Policy", "Program Policies", and "Google Home".

The screenshot shows the modern Gmail interface. At the top, it says "Gmail". Below the header are search bars and a search button. A "COMPOSE" button is visible. The main area shows an inbox with several messages from Google+ users like Jenny Kang, Peter H, and Brett C. There are sections for "Primary", "Social" (3 new), "Promotions" (2 new), and "Updates" (2 new). The "Social" section includes links for "You were tagged in 3 photos on Google+", "LauraBlack just uploaded a video.", "[Knitting Club] Are we knitting tonight?", and "Danielle Hoodhood added you on Google+". The "Promotions" section includes "Google Offers, Zagat". The "Updates" section includes "Shoehop, Blitz Air". On the left sidebar, there are links for "Inbox", "Starred", "Drafts", and "Sent Mail". Below the sidebar is a "Search people..." field and a list of contacts: Jenny Kang, Peter H, Jonathan Pelleg, Brett C, Max Stein, Jen Hart, and Eric Lowery.

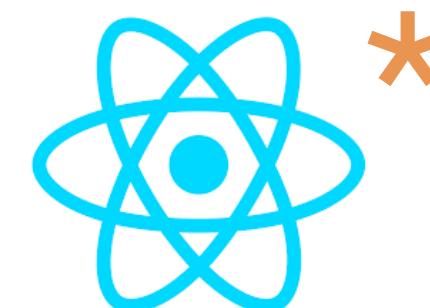
# Introdução

## 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 negócios
- 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

## 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

- 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



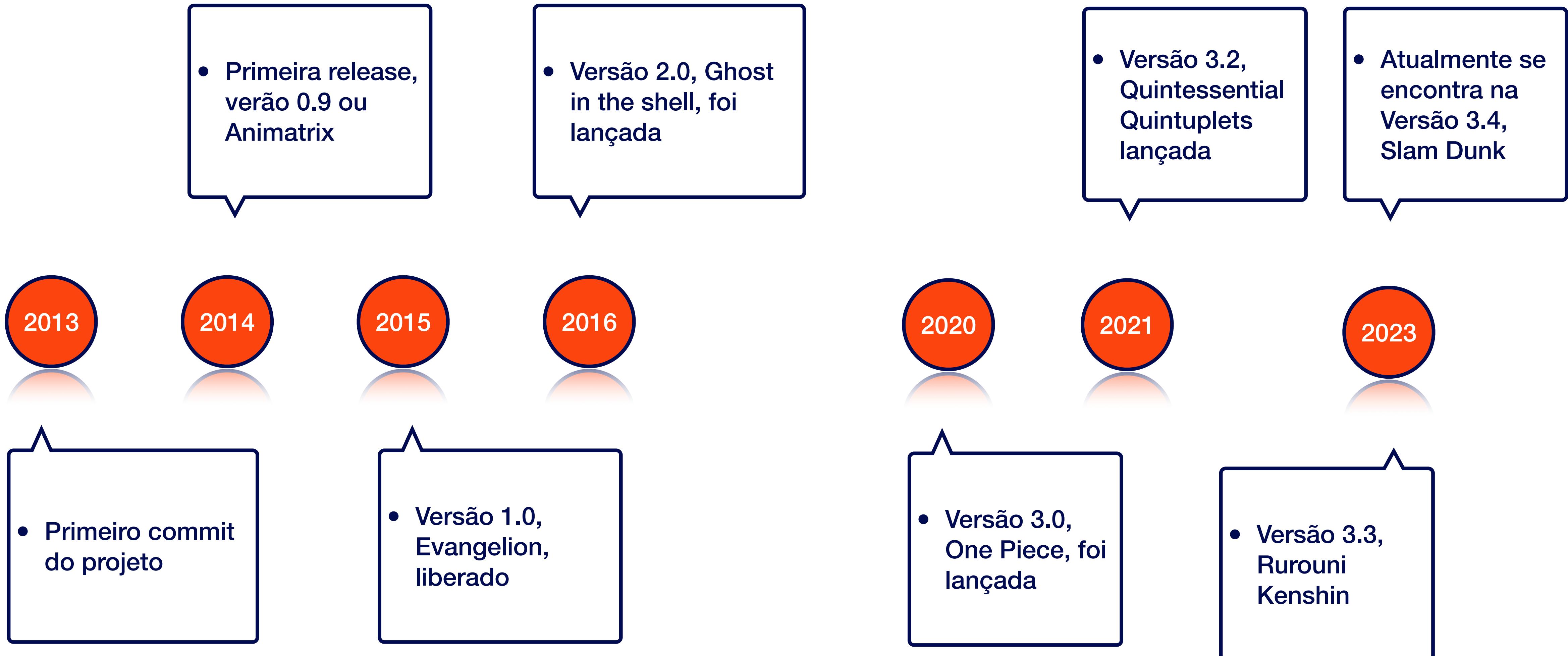
# Introdução ao VueJS

## 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**

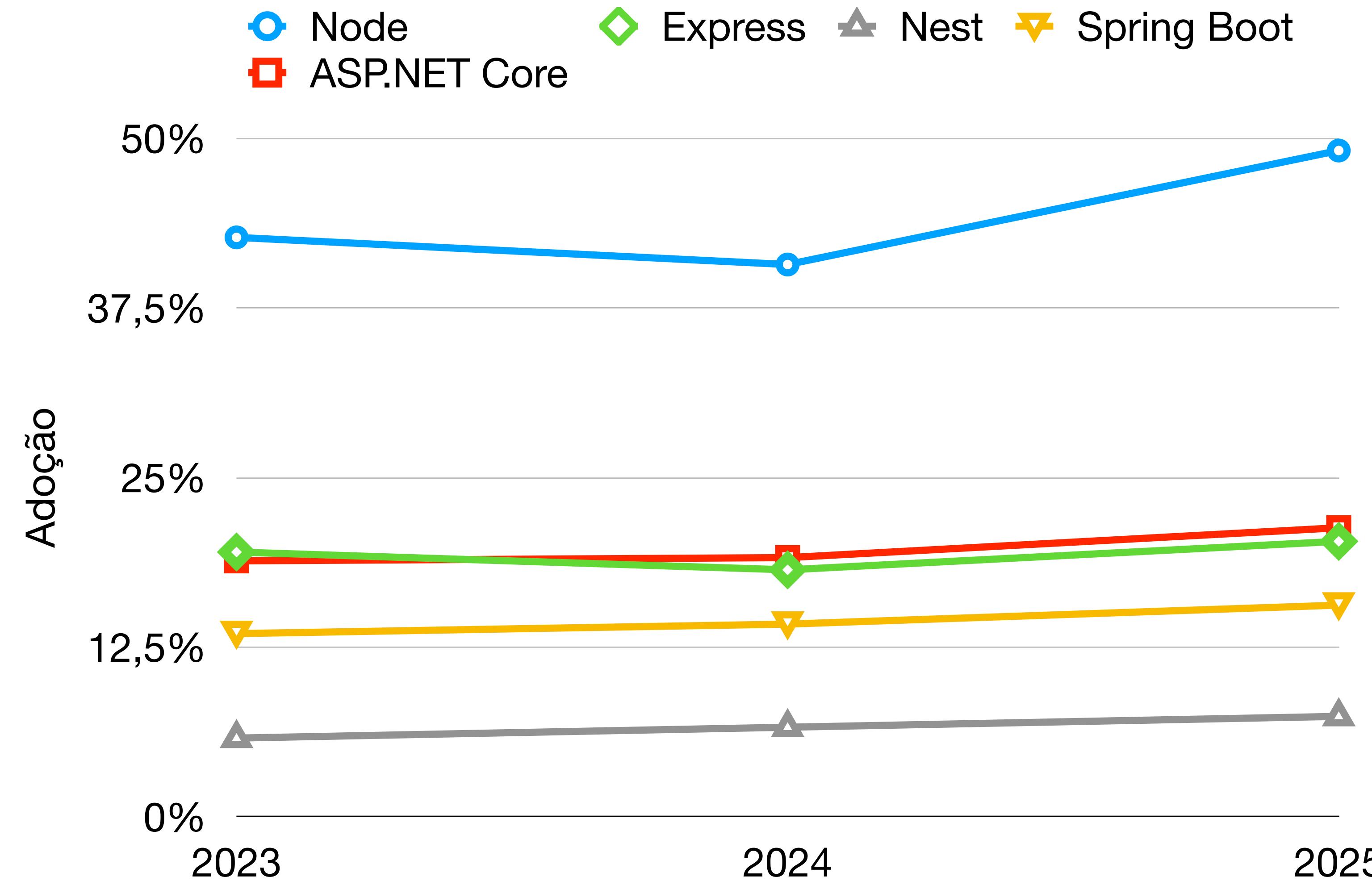
# Introdução

## História - Timeline



# Introdução ao VueJS

## Comunidade

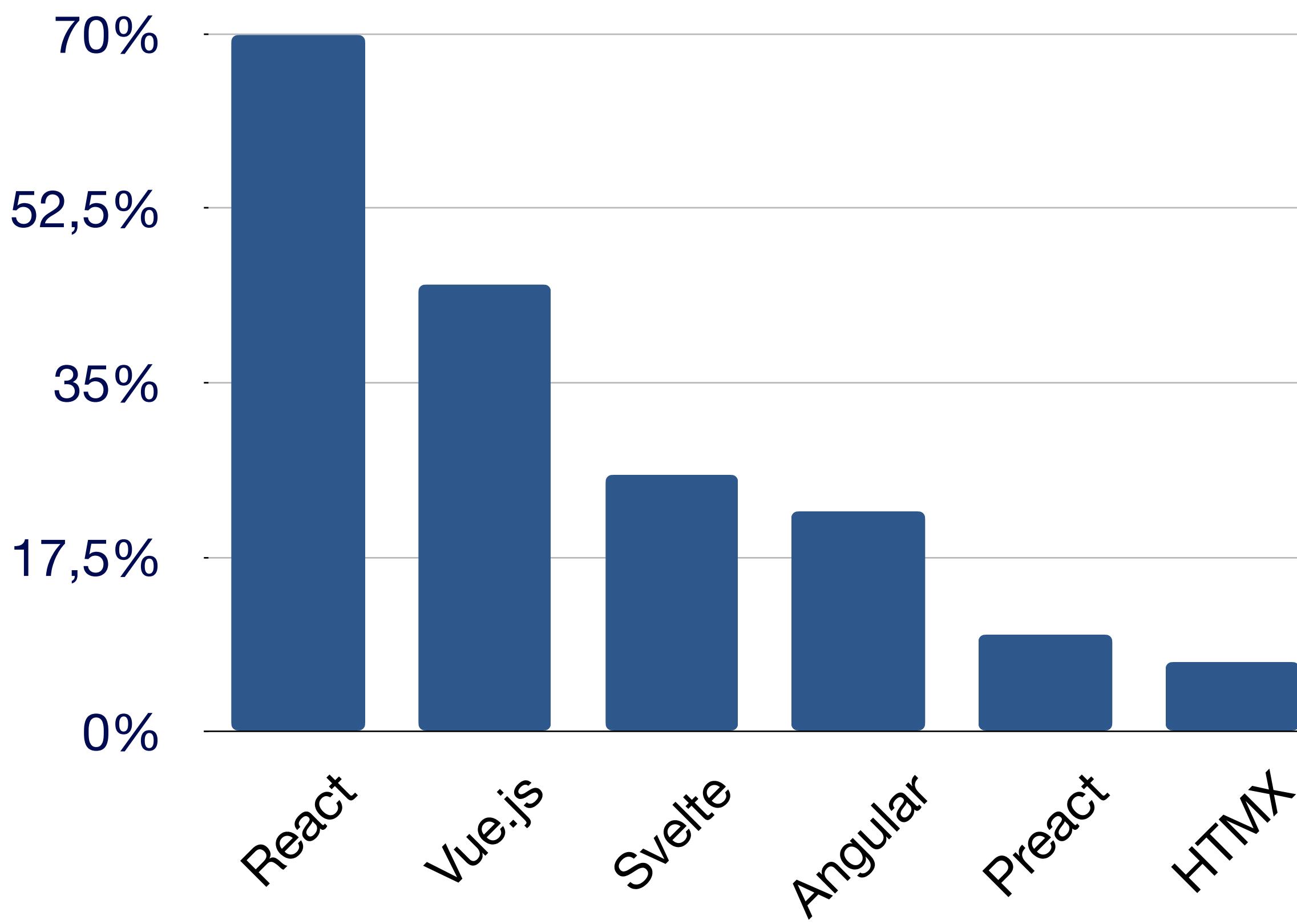


Fonte: <https://survey.stackoverflow.co>

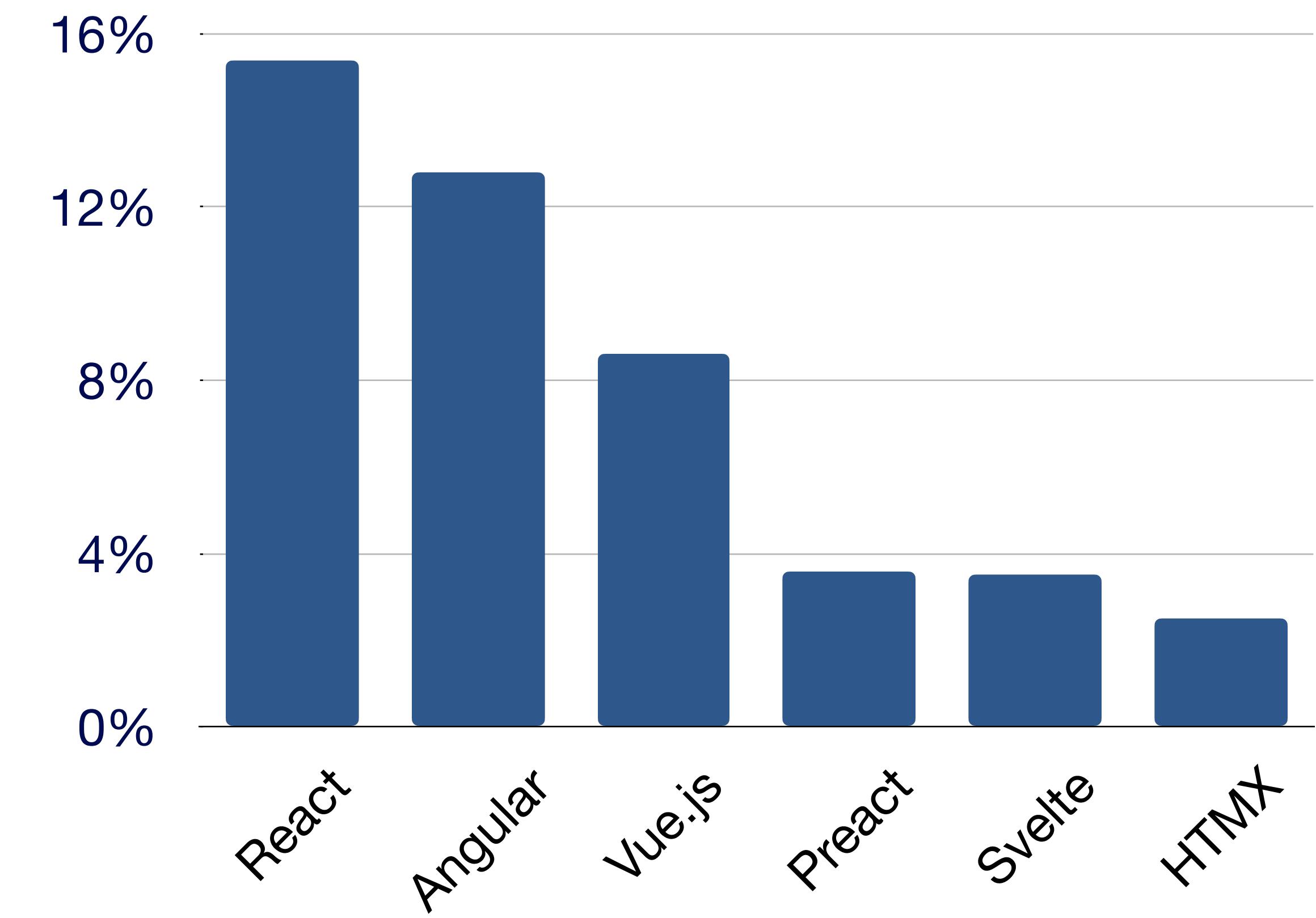
# Introdução ao VueJS

## Comunidade

Quais frameworks você utilizou no último ano e gostou?



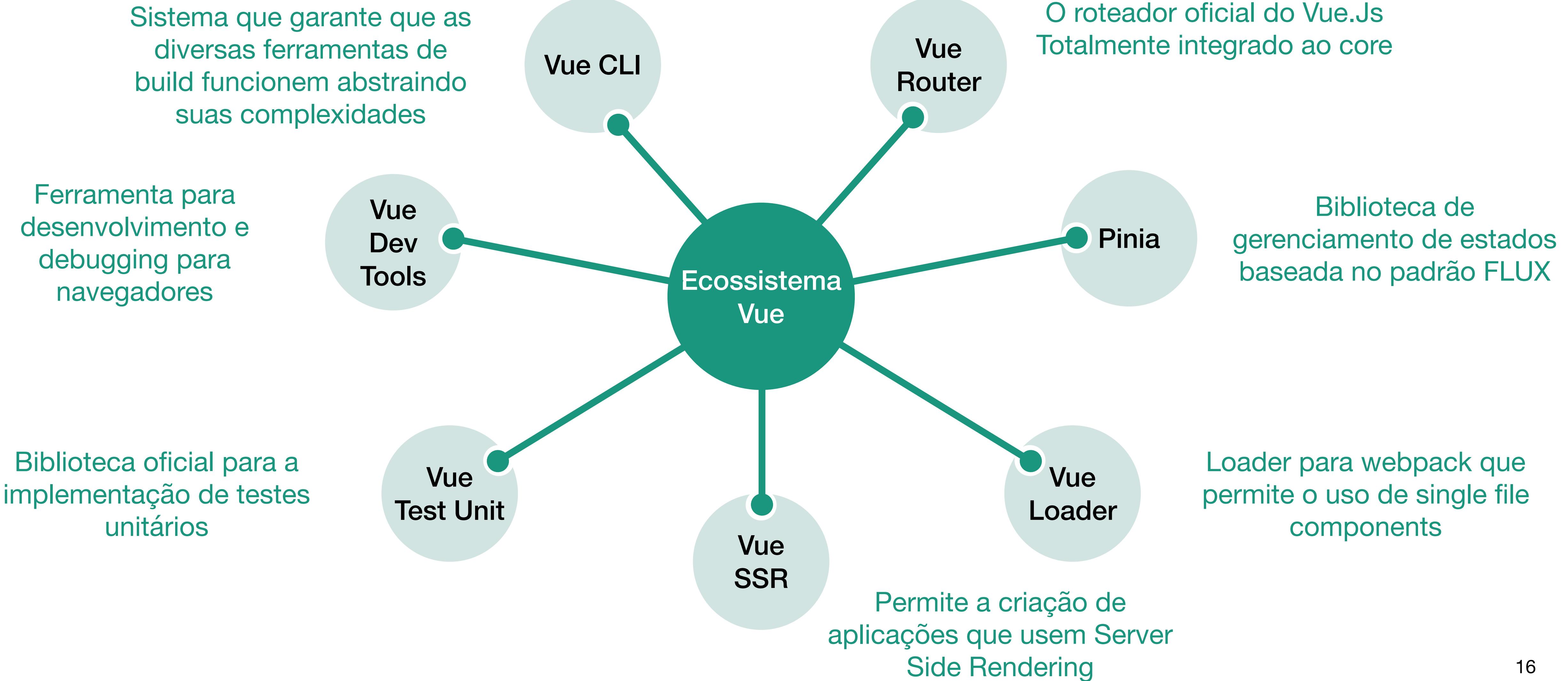
Qual framework você usou e não gostou?



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

# Introdução ao VueJS

## Ecossistema



# Introdução ao VueJS

## Vantagens

- Leve
  - Pouco mais de 18KB, consideravelmente menor que os concorrentes
- Performance e Virtual DOM
- Data binding reativo de duas mão (*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

# Criando uma aplicação Vue



# Criando uma aplicação Vue

```
const { createApp } = Vue  
const app = 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

# Criando uma aplicação Vue

```
const { createApp } = Vue
const RootComponent = {/* options */}
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

# Criando uma aplicação Vue

```
const { ref, createApp } = Vue
const vm = createApp({
  setup() {
    const count = ref(4)
    return { count }
  }
}).mount('#app')
console.log(vm.count) // -> 4
```

- Cada componente pode expor por meio do retorno da função *setup*
  - Deve retornar um objeto
  - Automaticamente são rastreados pelo sistema de reatividade do Vue

# Criando uma aplicação Vue

## Data binding, interpolação e Diretivas

```
const vm = createApp({  
  setup() {  
    const count = ref(4)  
    return { count }  
  }  
}).mount('#app')  
vm.count++
```

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

Message: 5

- 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

# Diretivas



# Diretivas

```
const vm = createApp({  
  setup() {  
    const seen = ref(true)  
    return { seen }  
  }  
}).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

# 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

```
createApp( {  
    setup() {  
        const items = ref([ { message: 'Foo' } , { message: 'Bar' } ] )  
        return { items }  
    }  
} ).mount('#app')
```

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

# Diretivas

## Renderização de listas

```
createApp({  
  setup() {  
    const items = ref([ { message: 'Foo' } , { message: 'Bar' } ])  
    return { items }  
  }  
}).mount('#app')
```

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

# Diretivas

## Argumentos

- 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

## Tratando eventos

```
createApp({  
  setup() {  
    const count = ref(0)  
    return { count }  
  }  
}).mount('#app')
```

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

# Diretivas

## Tratando eventos

```
createApp({  
  setup() {  
    const name = ref('Bruno')  
    function greet(event) {  
      alert('Hello ' + name.value + '!')  
      if (event) { alert(event.target.tagName) }  
    }  
    return { name, greet }  
  }  
}).mount('#app')
```

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

# Diretivas

## Tratando eventos

```
createApp( {  
    setup() {  
        function say(something) {  
            alert(something)  
        }  
        return { say }  
    }  
} ).mount('#app')
```

```
<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 classes (CSS)

```
createApp({  
  setup() {  
    const isActive = ref(true)  
    const hasError = ref(false)  
    return { isActive, hasError }  
  }  
}).mount('#app')
```

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

# Diretivas

## Vinculação de classes (CSS)

```
createApp( {  
    setup() {  
        const activeClass = ref("active")  
        const errorClass = ref("danger")  
        return { activeClass, errorClass }  
    }  
} ).mount('#app')
```

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

# Diretivas

## Vinculação de formulários

- Ao utilizamos formulários, frequentemente precisamos sincronizar o dados de entradas a elementos que JavaScript que representem esse estado
- Usando as diretivas básicas que vimos podemos fazer isso assim:

```
<input  
  :value="text"  
  @input="event => text = event.target.value">
```

- Ou simplesmente assim:

```
<input v-model="text">
```

# Diretivas

## V-model

- A diretiva *v-model* é usada para criar *two-way binding*
- Pode ser usadas com diferentes tipos de elementos de entrada
- Atualiza corretamente o elemento baseado no tipo de *input*
  - *text* e *textarea* usam a propriedade *value* e o evento *input*
  - *checkbox* e *radio buttons* 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>
```

```
createApp( {
  setup() {
    const picked = ref('')
    return { picked }
  }
}) .mount('#app')
```

# 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>
```

```
createApp( {
  setup() {
    const selected = ref('')
    return { selected }
  }
}).mount('#app')
```

# 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

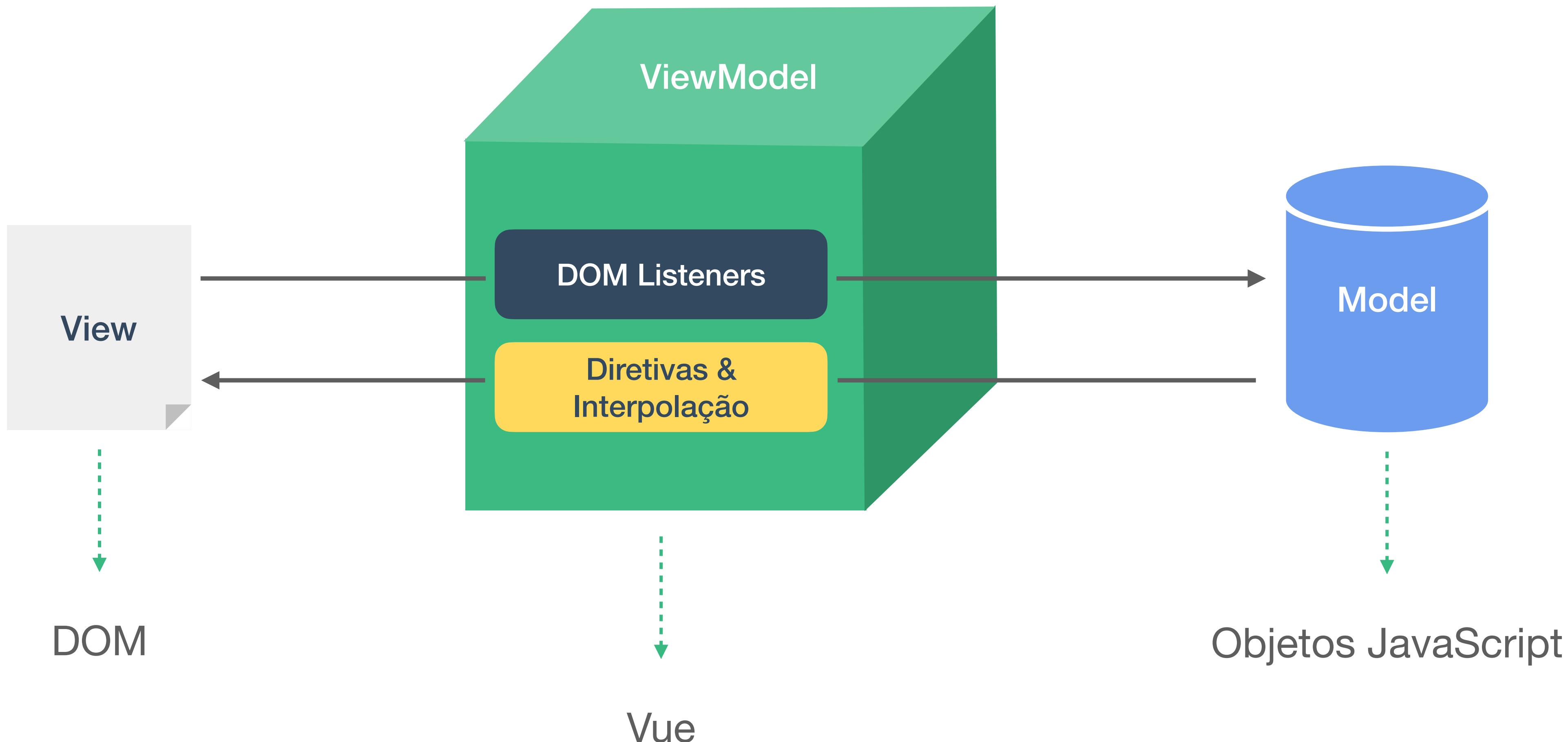
```
<!-- full syntax -->
<a v-on:click="doSomething"> ... </a>
<!-- shorthand -->
<a @click="doSomething"> ... </a>
<!-- shorthand with dynamic argument -->
<a @*[event] ="doSomething"> ... </a>
```

# Principais aspectos de uma aplicação em VueJs



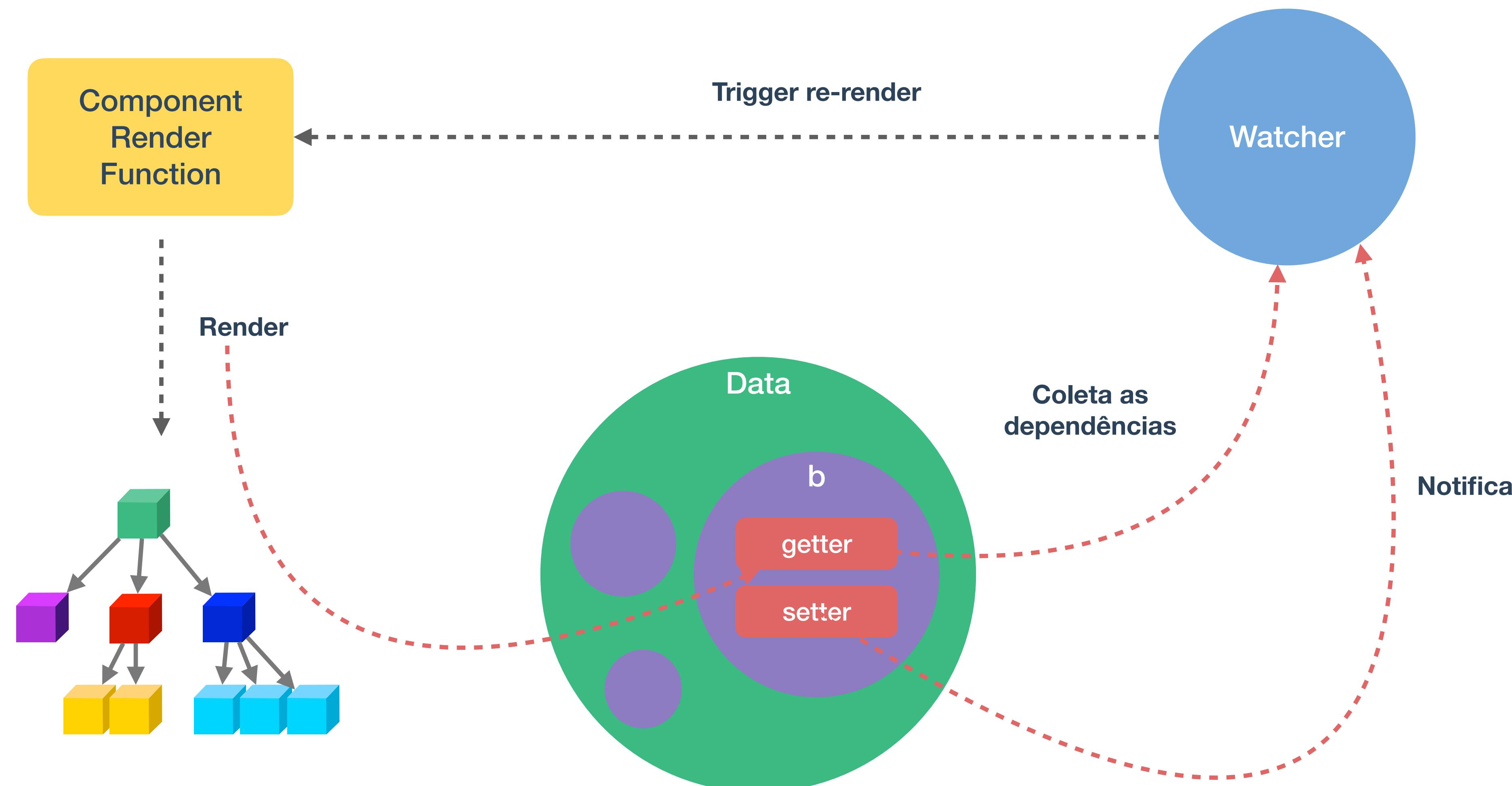
# Principais aspectos de uma aplicação em Vue.js

## Data binding



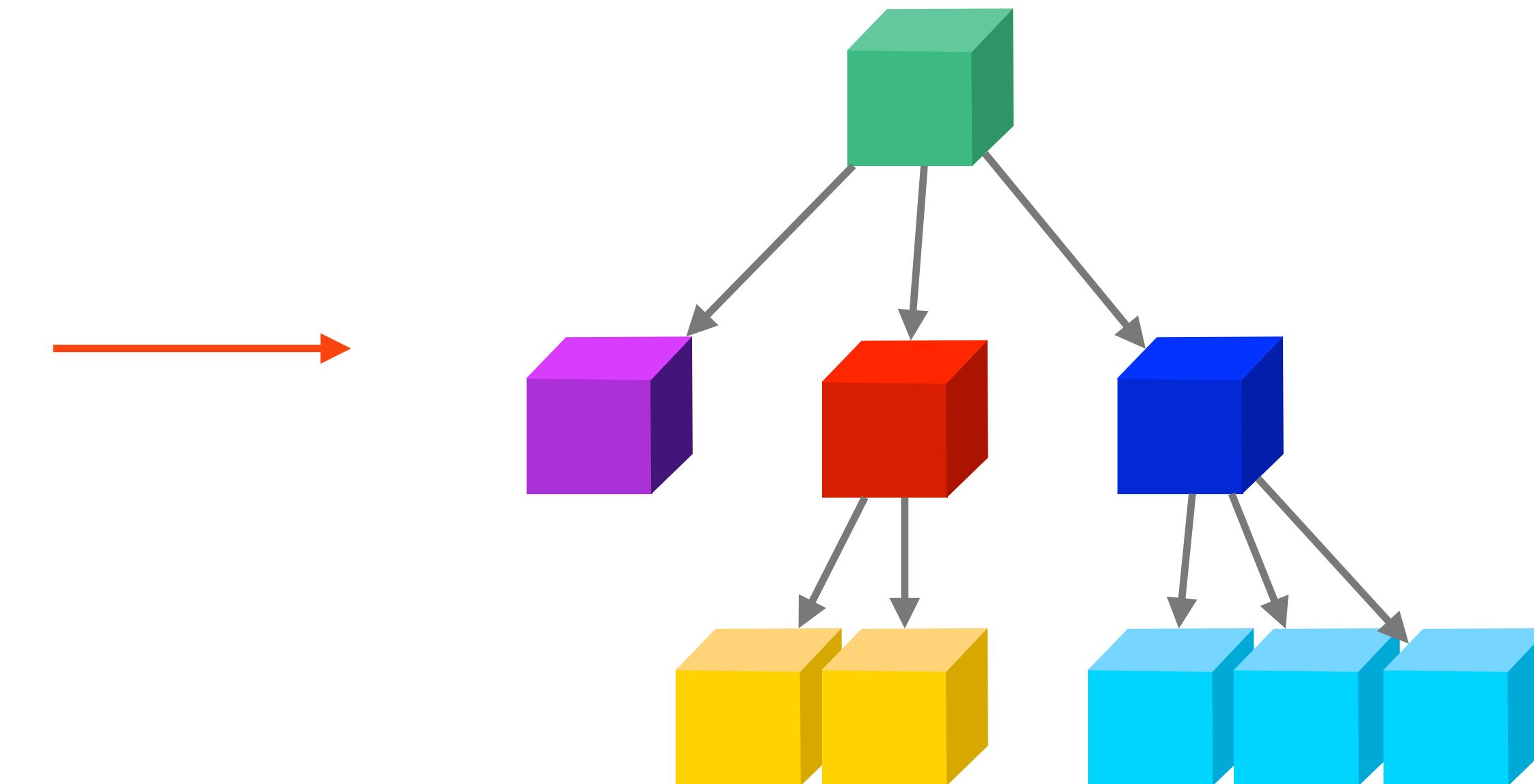
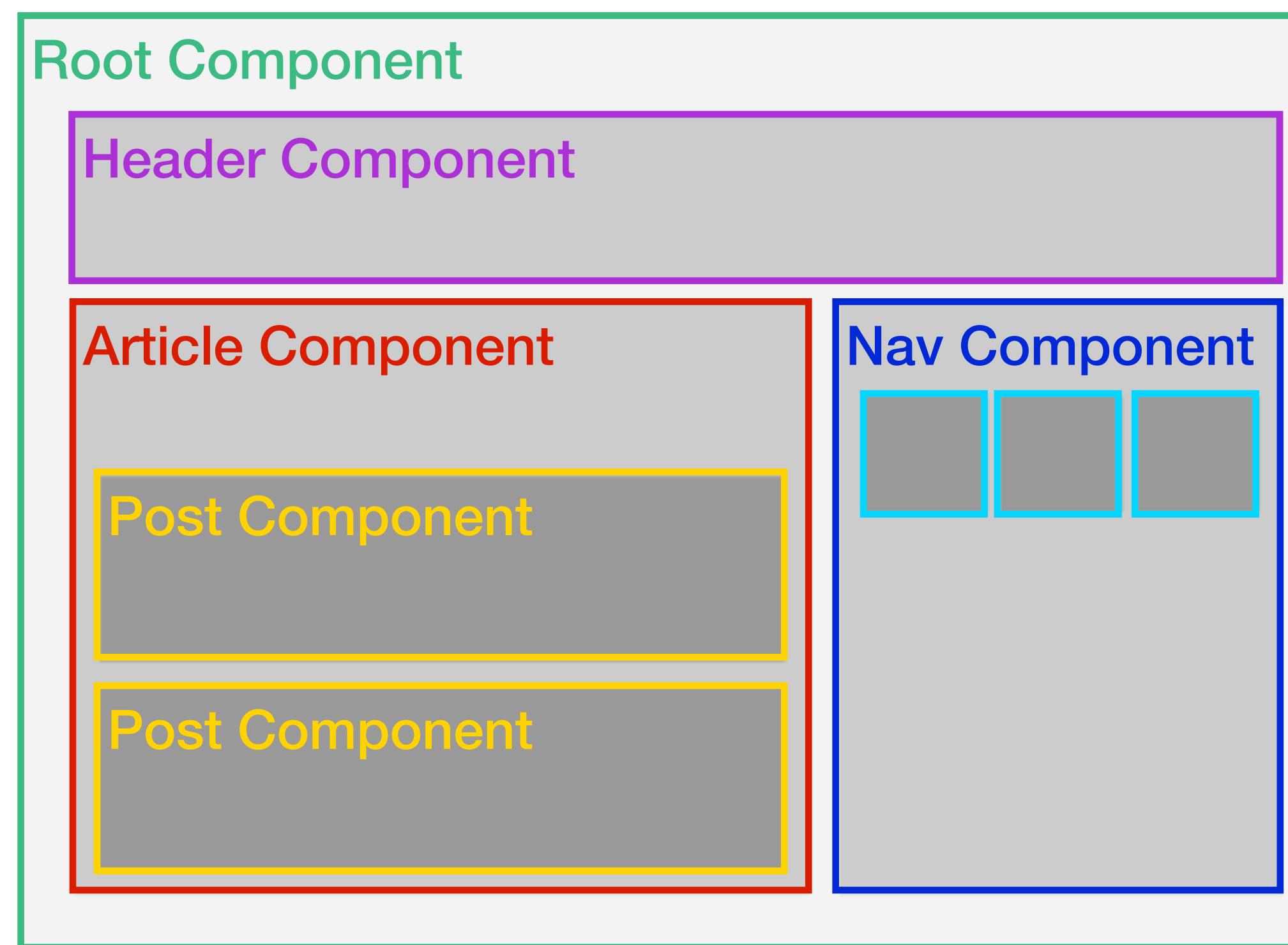
# Principais aspectos de uma aplicação em VueJs

## Data binding



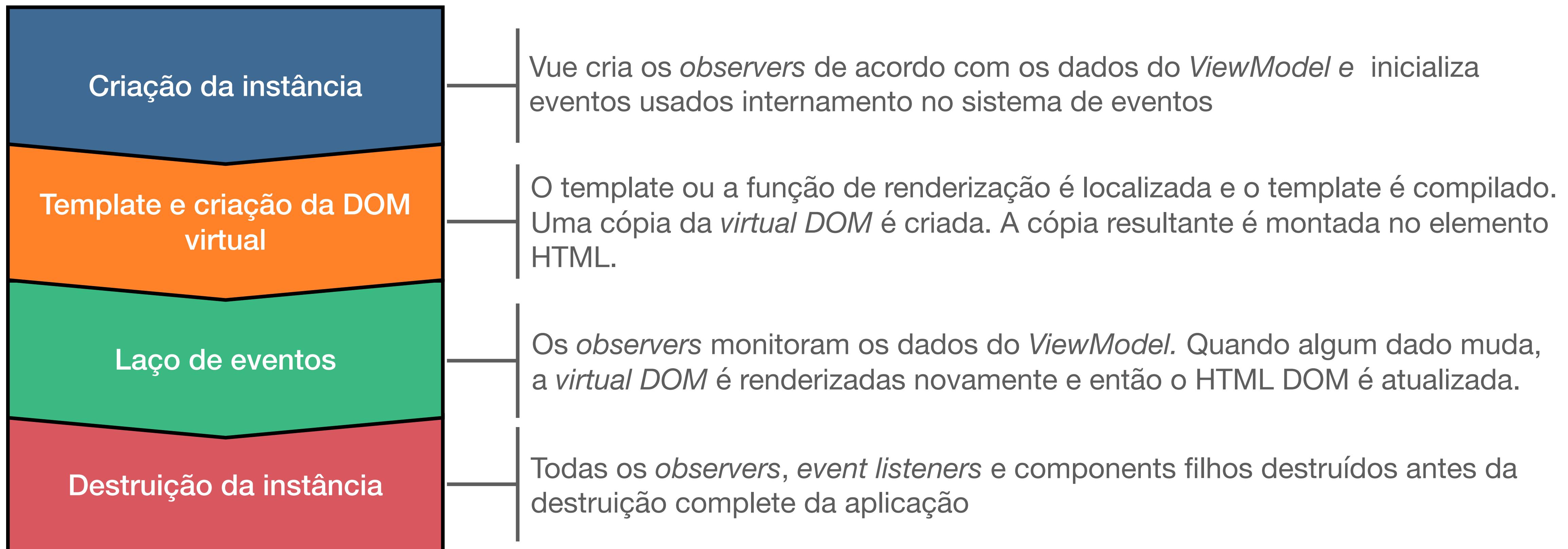
# Principais aspectos de uma aplicação em VueJs

## Componentes



# Principais aspectos de uma aplicação em VueJs

## Ciclo de Vida



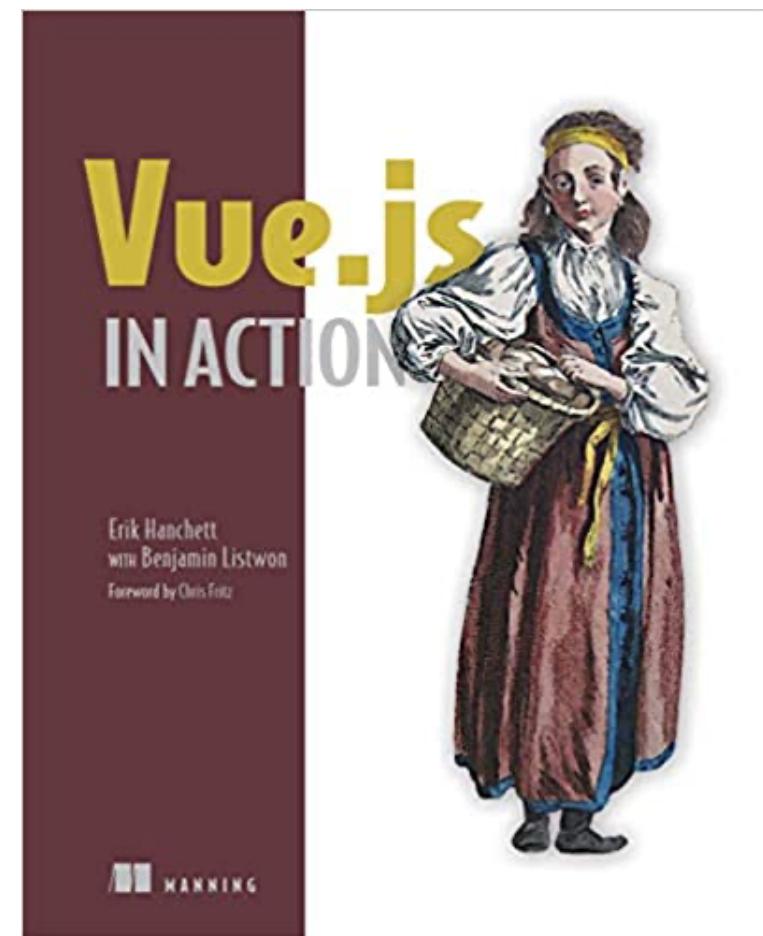
# Principais aspectos de uma aplicação em VueJs

## Ciclo de Vida

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

# 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