



Introdução ao framework Vue.js

Desenvolvimento de Sistemas Web (DSWI6)

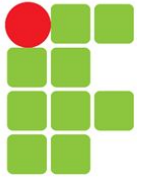
Prof. Luiz Gustavo Diniz de Oliveira Vêras

E-mail: gustavo_veras@ifsp.edu.br



Sumário

- Introdução ao Vue.js
- Arquitetura MVVM
- Componentes reutilizáveis
- Templates
- Diretivas
- Tutoriais úteis



Introdução ao Vue.js

O Vue foi criado pelo desenvolvedor independente chinês Evan You, após trabalhar para o Google usando o AngularJS em vários projetos.

Seu objetivo era criar um framework realmente muito leve e apenas com as partes do Angular que Evan gostava.

O primeiro commit do código do Vue data de 2013, e seu lançamento oficial foi em 2014.

É um dos frameworks de desenvolvimento de interface web mais populares junto com o React e o ASP.NET.

[https://insights.stackoverflow.com/survey/2020#technology-most-loved-d-dreaded-and-wanted-web-frameworks-loved2](https://insights.stackoverflow.com/survey/2020#technology-most-loved-dreaded-and-wanted-web-frameworks-loved2)

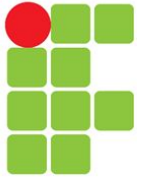


Introdução ao Vue.js



The Progressive
JavaScript Framework

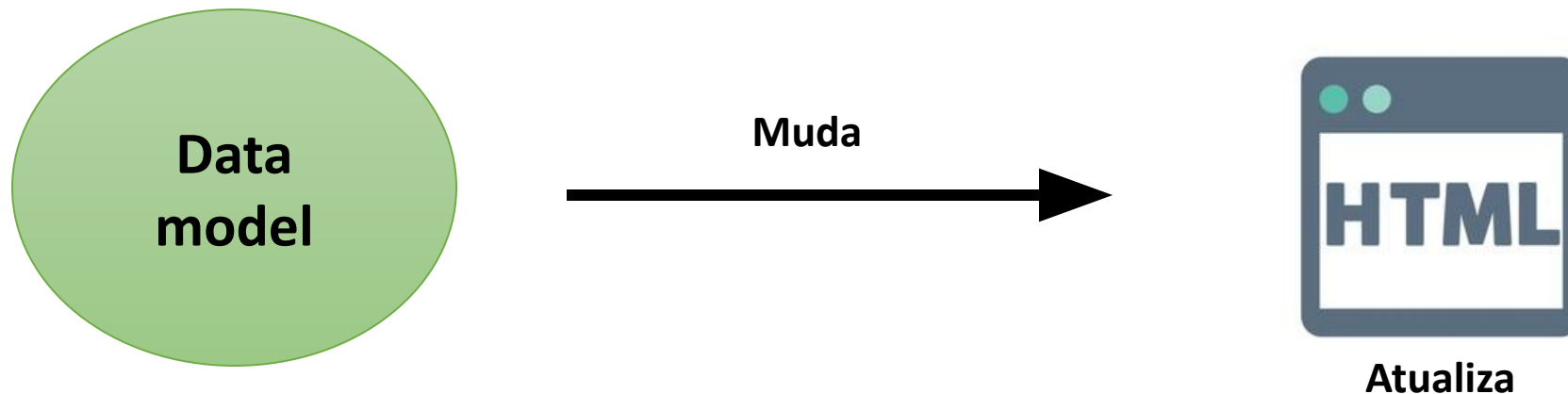
Vue (pronuncia-se /vju:/, como view, em inglês) é um framework **progressivo** para a **construção de interfaces de usuário**. Ao contrário de outros frameworks monolíticos, Vue foi projetado desde sua concepção para ser adotável incrementalmente.

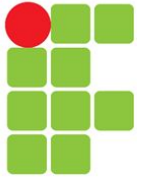


Introdução ao Vue.js

É basicamente um modelo de template vinculado a um modelo de dados.

Se o modelo for atualizado, o HTML do navegador será atualizado, ou seja, ele reage às atualizações do modelo.



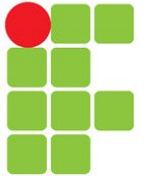


Introdução ao Vue.js

A palavra “**progressivo**” significa que a estrutura é implementada como uma marcação adicional para HTML, mas que bibliotecas oficiais podem ser adicionadas (ou não) para aumentar suas funcionalidades.

A biblioteca principal é focada exclusivamente na camada visual (view layer), sendo fácil adotar e integrar com outras bibliotecas ou projetos existentes. Com essas outras bibliotecas de apoio é possível:

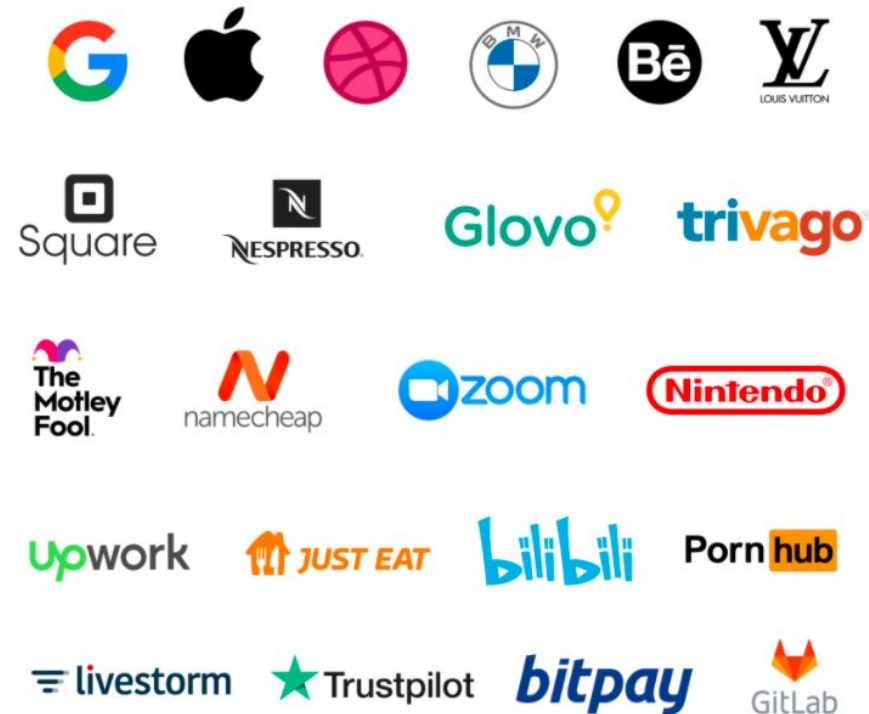
- Controlar rotas de páginas (com o Vue-Router)
- Trabalhar com gerenciamento de estados (com o Vuex)
- Renderizar HTML no servidor (com o Vue Server Rendered)
- Criar Single-Pages Applications – SPA (Misturando as bibliotecas acima)
- Muito mais...



Introdução ao Vue.js

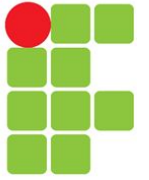
Quem usa Vue.js

<https://stackshare.io/vue-js>



E no Brasil?

- <https://empresas-usando-vuejs.netlify.app/>

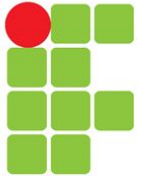


Utilizando via CDN

Para utilizar o Vue na sua página, insira a tag script a seguir no seu HTML

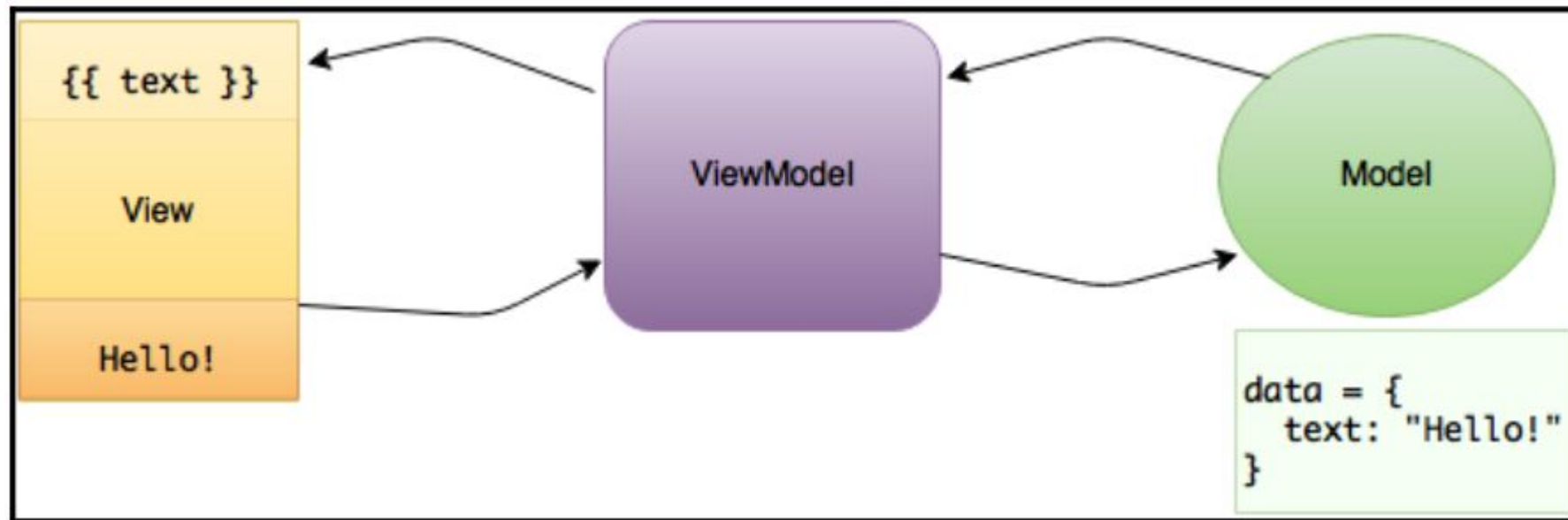
```
<script src="https://unpkg.com/vue@3"></script>
```

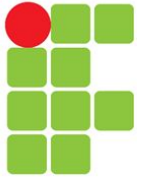
Obs: A versão 3 é a que estaremos utilizando.



Arquitetura MVVM

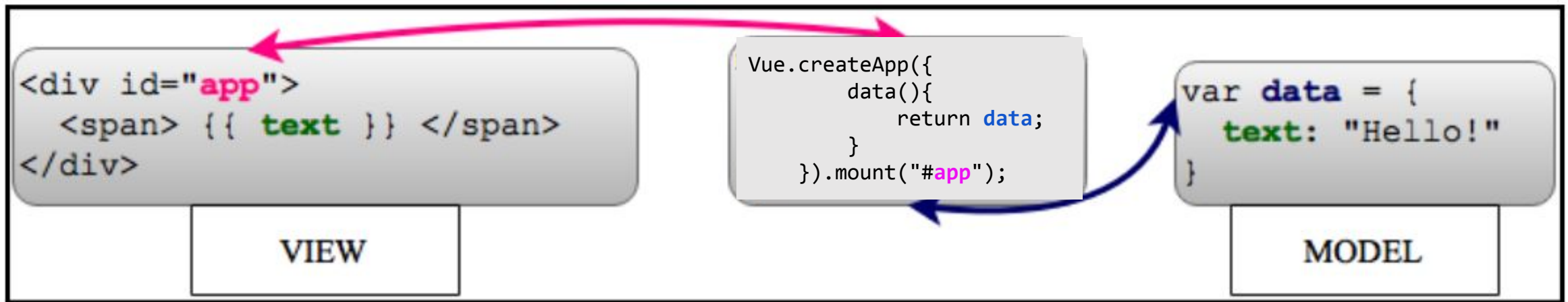
MVVM – Model-View-View-Model

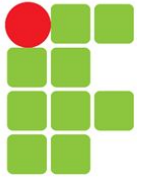




Arquitetura MVVM

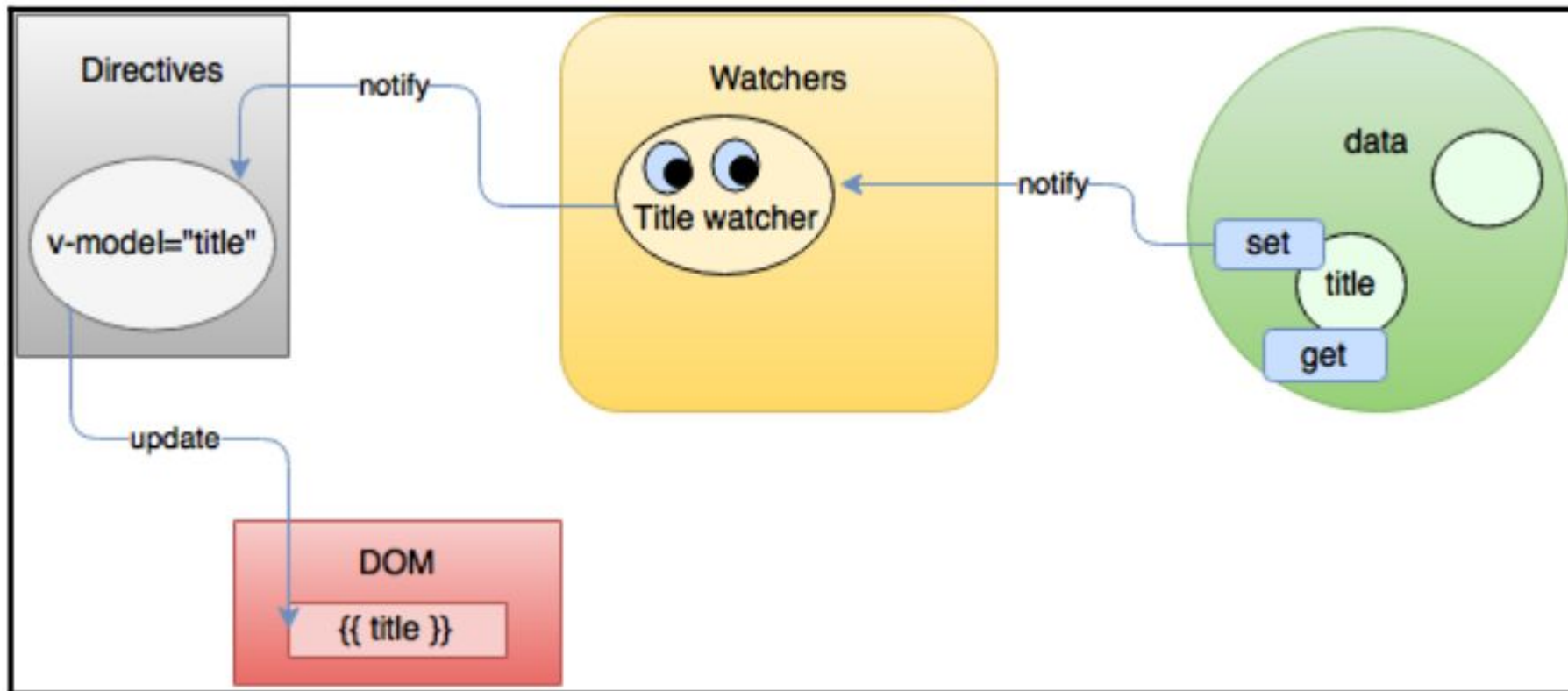
MVVM – Model-View-View-Model

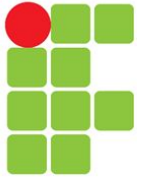




Arquitetura MVVM

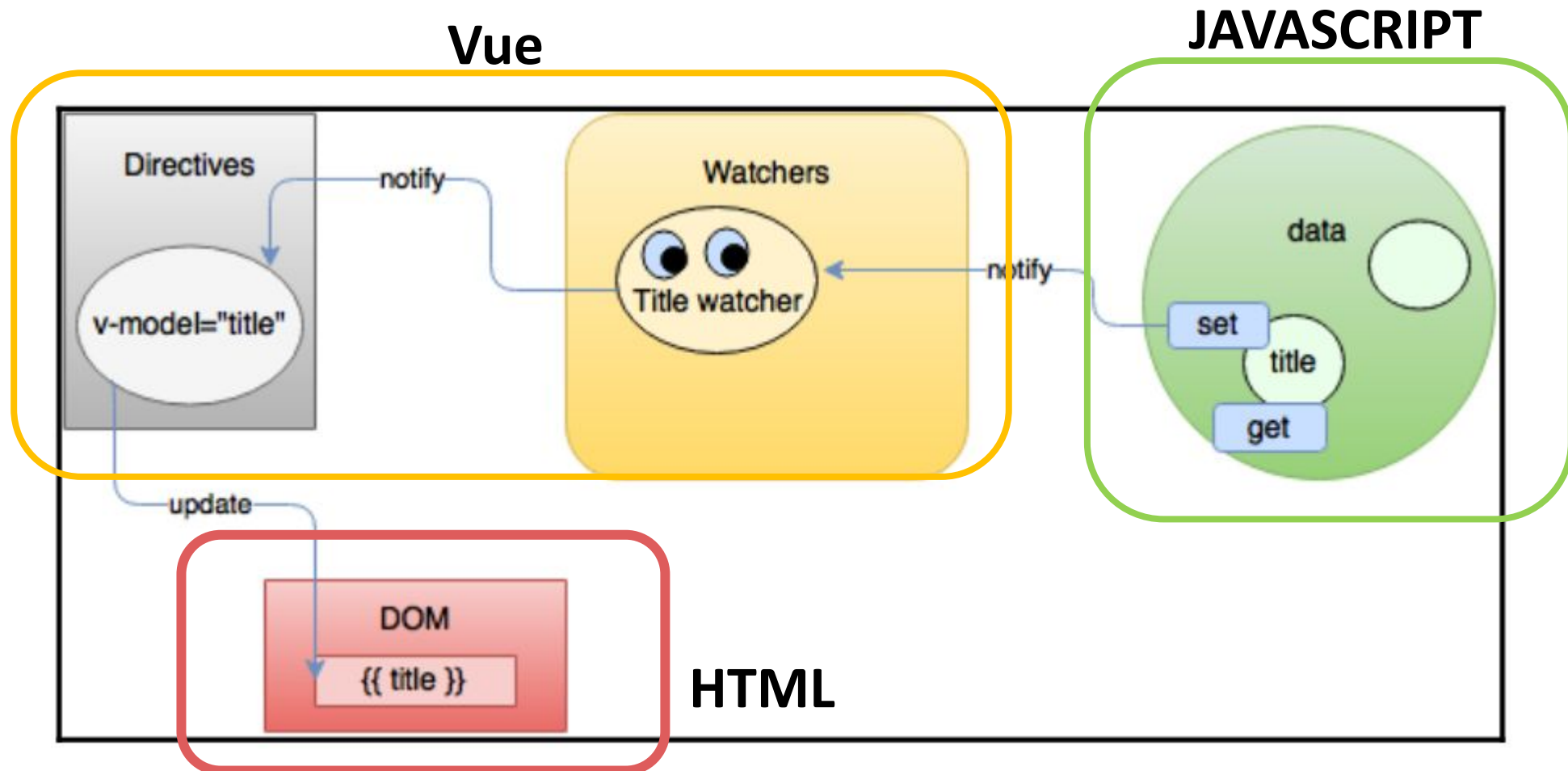
Como o Vue torna a sua aplicação reativa?





Arquitetura MVVM

Como o Vue torna a sua aplicação reativa?

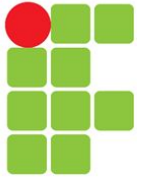




Componentes reutilizáveis

- O framework Vue possui uma forte estrutura para a criação de componentes.
- Componentes são “tijolos” pré formatados que podem ser utilizados para construir sua aplicação, bastando alimentados com os dados.





Componentes reutilizáveis

Sintaxe básica

Passos para criar um componente

Criando e registrando o componente

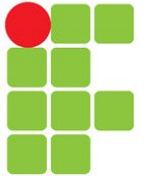
```
const app = Vue.createApp({});
app.component(
  // O nome do componente registrado
  'MeuComponente',
  // A implementação
  {
    data(){
      return {attr1: 'Filme'};
    },
    template: `<div>{{attr1}}</div>`
  }
);
app.mount("#app");
```

Utilizando o componente (no HTML)

```
<nome-component></nome-component>
```

Estrutura:

- **data** – função que retorna o modelo de dados relativo ao componente;
- **template** – código HTML que estrutura o componente.



Componentes reutilizáveis

Exemplos

Definindo o model do componente

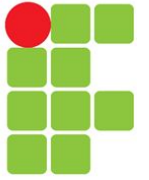
```
// Model do Componente
let modelComponente = {
  titulo: 'Filme',
  descricao: 'Descricao do Filme'
}
```

Utilizando o componente

```
<div id="app">
  <meu-componente></meu-componente>
  <meu-componente></meu-componente>
  <meu-componente></meu-componente>
  <meu-componente></meu-componente>
</div>
```

Criando e Registrando o componente

```
app.component(
  // O nome registrado
  'MeuComponente',
  // A implementação
  {
    data(){
      return modelComponente;
    },
    template: `<ul>
      <li>{{titulo}}</li>
      <li>{{descricao}}</li>
    </ul>`
  }
);
```



Componentes reutilizáveis

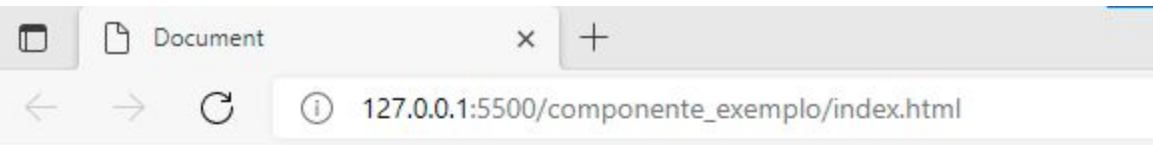
Exemplos

Definindo o model do componente

```
// Model do Componente
let modelComponente = {
  titulo: 'Filme',
  descricao: 'Descrição do Filme'
}
```

Utilizando o componente

```
<div id="app">
  <meu-componente></meu-componente>
  <meu-componente></meu-componente>
  <meu-componente></meu-componente>
  <meu-componente></meu-componente>
</div>
```



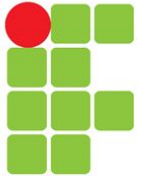
- Filme
- Descricao do Filme
- Filme
- Descricao do Filme
- Filme
- Descricao do Filme

ente

ente;

```
ulo}}</li>
ricao}}</li>
```

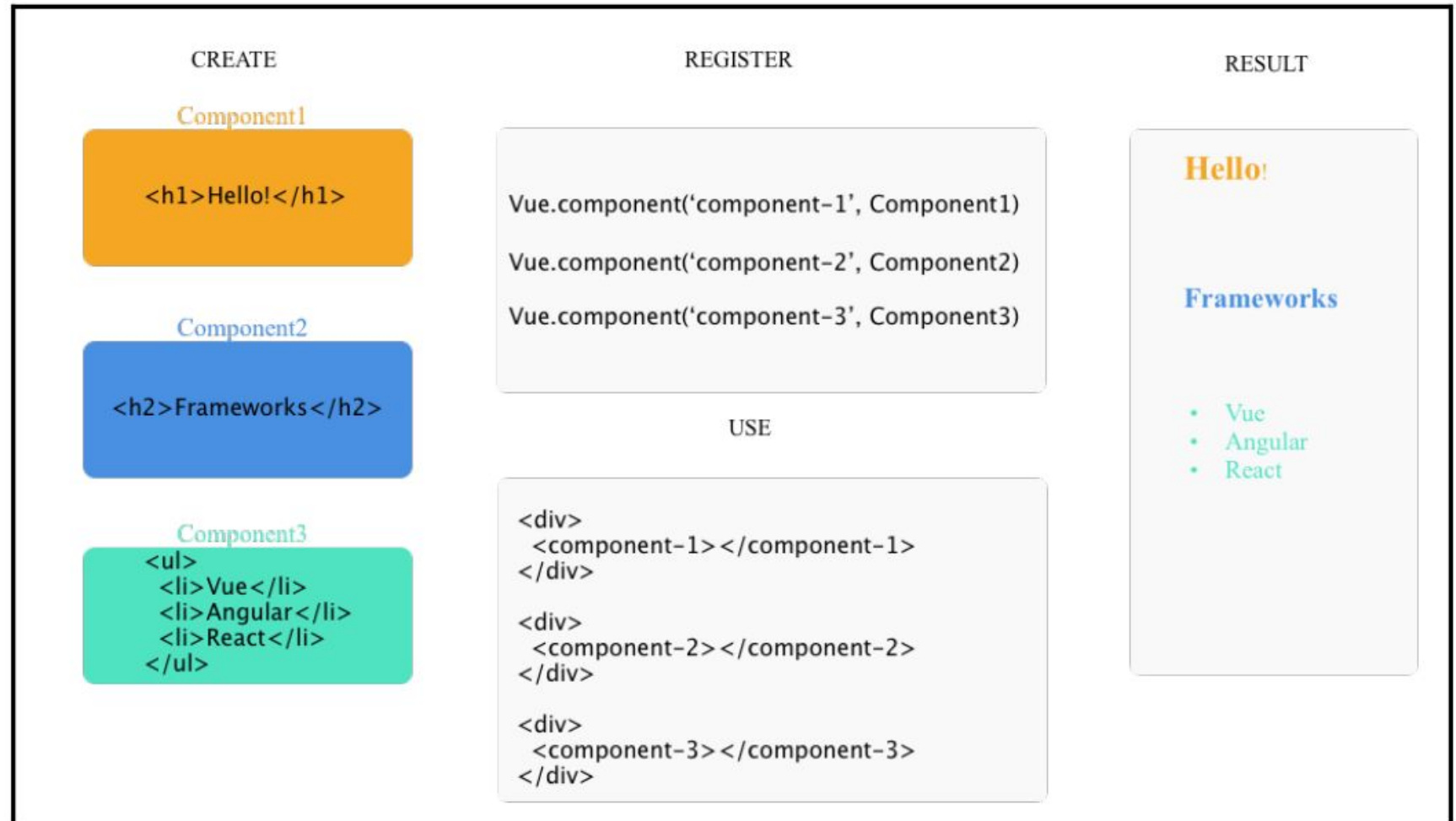
```
}
);
```

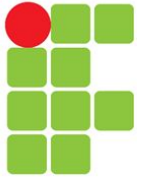



Componentes reutilizáveis

Exemplos

Obs: Exemplo está no Vue 2.



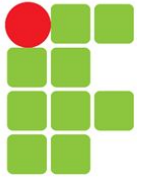


Reorganizando os componentes

- Imagine que precisássemos construir um componentes complexo, com muito código HTML.
- Não é uma boa ideia escrever código HTML como string, pois a sua manutenção se torna difícil e o código javascript bagunçado.

Como organizar componentes então?

- Utilizando a tag HTML **<template>**



Reorganizando os componentes

Exemplo:

Usando o template (no HTML)

```
<template id="hello">
  <h2>Hello!</h2>
</template>
```

Usando o componente (no HTML)

```
<hello-component></hello-component>
```

Criando e Registrando o componente

```
app.component(
  'HelloComponent',
  {
    template: "#hello"
  }
);
```

Note a relação entre a definição do **template** e a criação do componente via o id "hello". No campo **template** atente-se para a **hashtag (#)** à frente do id do atributo **template**.

Nome dado ao componente se torna uma **tag** no **HTML**.

Reorganizando os componentes

Importante!

Usando direto no browser, o nome do componente *HelloComponent* é convertido para a tag *hello-component*.

Exemplo:

Usando o template (no HTML)

```
<template id="hello">
  <h2>Hello!</h2>
</template>
```

Criando e Registrando o componente

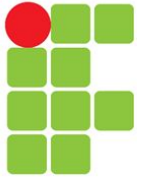
```
app.component(
  'HelloComponent',
  {
    template: "#hello"
  }
);
```

Usando o componente (no HTML)

```
<hello-component></hello-component>
```

Note a relação entre a definição do **template** e a criação do componente via o id "hello". No campo **template** atente-se para a **hashtag (#)** à frente do id do atributo **template**.

Nome dado ao componente se torna uma **tag** no **HTML**.



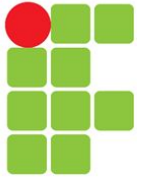
Reorganizando os componentes

Manipulando dados e propriedades dentro de um componente

- A sintaxe de criação de um componente é parecida à sintaxe da instância do Vue, mas os criamos com o método `app.component()`.
- Entretanto, para definir o elemento e o dado do componente, devemos definir funções ao invés de passarmos valores como diretamente valores aos campos.

Componente

```
data = {  
  attr1 = "",  
  attr2 = ""  
}
```



Reorganizando os componentes

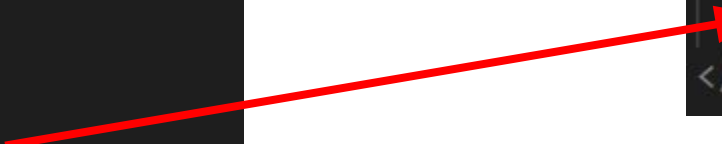
Manipulando dados e propriedades dentro de um componente

Você deve definir o modelo de dados do componente numa função

```
app.component(  
  'HelloComponent',  
  {  
    data(){  
      return {msg: "Hello"};  
    },  
    template: "#hello"  
  }  
);
```

Chamando o dado no template (no HTML)

```
<template id="hello">  
  <h2>{{msg}}</h2>  
</template>
```

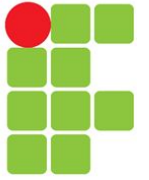




Reorganizando os componentes

Escopo dos componentes

- Todos os componentes têm seu próprio escopo, o qual é inacessível por outros componentes.
- Você pode ver o escopo dos **componentes** como **local** e o **escopo do aplicativo (instâncias vue)** como **global**. No entanto, o uso dos dados da instância vue por um componente não pode ser feito de forma direta.
- Você deve indicar explicitamente dentro de um componente quais dados do escopo global devem ser acessadas usando o campo **props** e vinculá-las a um componente usando a diretiva **v-bind**.



Reorganizando os componentes

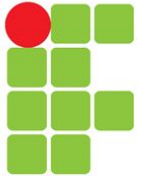
O data de um componente é diferente do data da instância do Vue (restrição de escopo)

app

```
data() {  
  return {  
    attr1 = "",  
    attr2 = ""  
  }  
}
```

Componente

```
data() {  
  return {  
    attr1 = "",  
    attr2 = ""  
  }  
}
```

Reorganizando os componentes

Escopo de dados

Criação e registro do componente

```
app.component(  
  'HelloComponent',  
  {  
    data(){  
      return {msg: "Hello"};  
    },  
    template: "#hello",  
    props: ['usuario']  
  }  
);
```

Template do componente

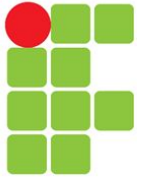
```
<template id="hello">  
  <div>  
    <h2>{{msg}} {{usuario}}</h2>  
  </div>  
</template>
```

No HTML (dentro da div do app)

```
<hello-componente v-bind:usuario = "meuNome"></hello-componente>
```

Criação do aplicativo

```
const app = Vue.createApp({  
  data(){  
    return {meuNome: "IFSP"}  
  }  
});
```



Reorganizando os componentes

Escopo de dados

O campo **props** define o que será mapeado para dentro do componente. No caso o atributo “usuário” será utilizado.

Criação e registro do componente

```
app.component(  
  'HelloComponent',  
  {  
    data(){  
      return {msg: "Hello"};  
    },  
    template: "#hello",  
    props: ['usuario']  
  }  
);
```

Template do componente

```
<template id="hello">  
  <div>  
    <h2>{{msg}} {{usuario}}</h2>  
  </div>  
</template>
```

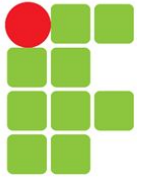
No HTML (dentro da div do app)

```
<hello-componente v-bind:usuario = "meuNome"></hello-componente>
```

Criação do aplicativo

```
const app = Vue.createApp({  
  data(){  
    return {meuNome: "IFSP"}  
  }  
});
```

Mapeia o campo “meuNome” do data para a propriedade “usuário” do componente.



Diretivas

Esses são atributos especiais de elementos HTML no Vue.js que permitem *data binding* (**vinculação de dados**) de diferentes maneiras. Essencialmente, uma diretiva é um *token* especial na marcação que instrui a biblioteca Vue a fazer algo com um elemento DOM.

Exemplos:

v-model

v-bind

v-if

v-on

v-for

entre outros

Para a descrição de uso das diretivas, ver:

<https://vuejsbr-docs-next.netlify.app/api/directives.html>



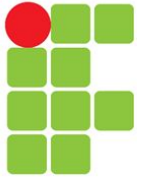
Diretivas

v-model (<https://vuejsbr-docs-next.netlify.app/guide/migration/v-model.html>)

Utilizado para realizar ligação bidirecional entre os dados e o **DOM (Document Object Model)**. A ligação bidirecional é um tipo de ligação em que não apenas as alterações de dados são propagadas para a camada DOM, mas também as alterações que ocorrem nos dados vinculados no DOM são propagadas para os dados.

Só pode ser utilizado nos seguintes elementos HTML:

- `<input>`
- `<select>`
- `<textarea>`



Diretivas

v-model

No Javascript

```
let modelComponente = {  
  title: 'Filme',  
  description: 'Descricao do Filme'  
}  
  
const app = Vue.createApp({  
  data(){  
    return modelComponente;  
  }  
});
```

No HTML

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

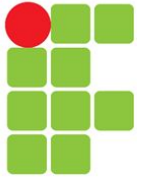
Na Interface

Document x +

127.0.0.1:5500/diretivas/index.hti

Matrix Reloaded

- Matrix Reloaded
- Descricao do Filme



Diretivas

v-model

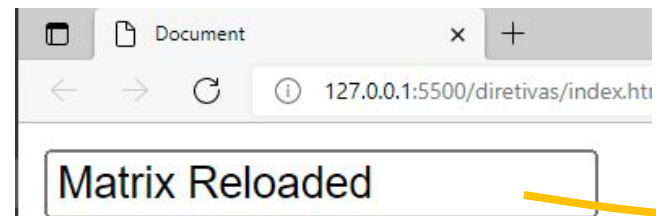
No Javascript

```
let modelComponente = {  
  title: 'Filme',  
  description: 'Descricao do Filme'  
}  
  
const app = Vue.createApp({  
  data(){  
    return modelComponente;  
  }  
});
```

No HTML

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

Na Interface



Alterando o INPUT também altera o model no Vue.

- Matrix Reloaded
- Descricao do Filme



Diretivas

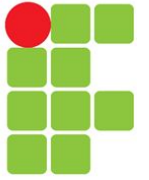
v-bind

(<https://vuejsbr-docs-next.netlify.app/guide/migration/v-bind.html>)

A diretiva v-bind permite vincular um atributo de uma tag ou uma propriedade de componente Vue para uma expressão ou campo de modelo de dados.

Para aplicá-lo ao atributo específico, usamos um delimitador de dois pontos:

- `v-bind:nomeatributodatag`



Diretivas

v-bind

Mapeia atribui o conteúdo de um atributo do model do Vue para um atributo do elemento HTML.

No Javascript

```
const app = Vue.createApp({
  data(){
    return {
      imggato = "https://cdn2.thecatapi.com/images/zkq4FPHEc.jpg"
    };
  }
});
```

No HTML

```

```

Na Interface





Diretivas

v-on

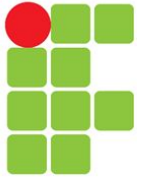
Você pode usar a diretiva v-on para escutar eventos do DOM e rodar algum JavaScript (função ou código incorporado) quando tal evento for disparado.

Exemplos:

```
<button v-on:click="counter += 1">Adiciona 1</button>
```

```
<button v-on:click="greet">Cumprimentar</button>
```

```
<!-- só chama `vm.submit()` quando o `key` é `Enter` -->  
<input v-on:keyup.enter="submit">
```



Diretivas

v-on

No Javascript

```
const app = Vue.createApp({
  data(){
    return modelComponente;
  },
  methods:{
    digaOi(){
      console.log("oi");
    }
  }
});
```

No HTML

```
<button v-on:click = "digaOi">Diga Oi</button>
```

ou

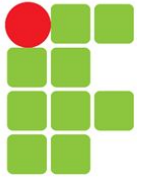
```
<button @click = "digaOi">Diga Oi</button>
```

Na Interface

Diga Oi

Click





Diretivas

v-on

No Javascript

```
const app = Vue.createApp({
  data(){
    return modelComponente;
  },
  methods:{
    digaOi(){
      console.log("oi");
    }
  }
});
```

O Vue define algumas abreviações para algumas das diretivas. Por exemplo, v-on pode ser indicado com @.

No HTML

```
<button v-on:click = "digaOi">Diga Oi</button>
```

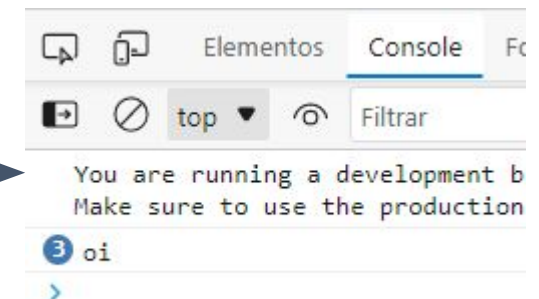
ou

```
<button @click = "digaOi">Diga Oi</button>
```

Na Interface

Diga Oi

Click





Diretivas

v-if

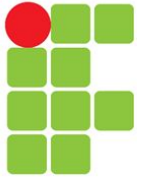
A diretiva v-if permite renderizar condicionalmente todo o elemento, o que também pode ser um elemento componente dependendo de alguma condição. A condição pode ser qualquer expressão e Ele também pode usar as propriedades dos dados.

Exemplos:

```
<div v-if="1 < 5">hello</div>
```

```
<div v-if="Math.random() * 10 < 6">Olá!</div>
```

```
<div v-if="new Date().getHours() >= 16">Tomar Suco!</div>
```



Diretivas

v-if

No Javascript

```
const app = Vue.createApp({  
  data(){  
    return {  
      tanahora: true  
    };  
  }  
});
```

No HTML

```
<h1 v-if="tanahora">Almoçar</h1>  
<h1 v-else>Descansar</h1>
```

Na Interface

Se tanahora = true

Almoçar

Se tanahora = false

Descansar



Diretivas

v-else-if

v-else

Assim como em condicionais tradicionais, você pode aninhar as diretivas v-if

```
<button v-if="isLoggedIn"> Logout </button>  
<label v-else-if="isLoginDisabled"> Register disabled </label>  
<button v-else> Log In </button>
```



Diretivas

v-for

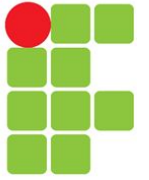
Para cada item em um array, essa diretiva renderizará um componente e vinculará a propriedade desse componente para o valor do item.

Associando a um campo do modelo de dados

```
<div v-for:item in items>  
  {{item}}  
</div>
```

Com componentes

```
<component v-for:item in items v-bind:param = "item.attr">  
</component>
```



Diretivas

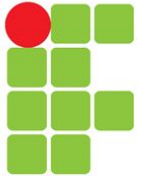
v-for

No Javascript

```
const app = Vue.createApp({
  data(){
    return {
      items: [
        {msg: "Mensagem 1"},
        {msg: "Mensagem 2"},
        {msg: "Mensagem 3"}
      ]
    };
  }
});
```

No HTML

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

Diretivas

v-for

No Javascript

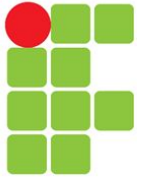
```
const app = Vue.createApp({
  data(){
    return {
      items: [
        {msg: "Mensagem 1"},
        {msg: "Mensagem 2"},
        {msg: "Mensagem 3"}
      ]
    };
  }
});
```

No HTML

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

Na Interface

- Mensagem 1
- Mensagem 2
- Mensagem 3



Diretivas

v-for

No Javascript

```
const app = Vue.createApp({  
  data(){  
    return {  
      items: [  
        {msg: "Mensagem 1"},  
        {msg: "Mensagem 2"},  
        {msg: "Mensagem 3"}  
      ]  
    };  
  }  
});
```

No HTML

```
<hello-componente v-for="item in items" v-bind:usuario = "item.msg">  
</hello-componente>
```

Template do componente

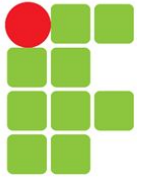
```
<template id="hello">  
  <h2>{{usuario}}</h2>  
</template>
```

Na Interface

Mensagem 1

Mensagem 2

Mensagem 3



Exemplo

Lista de Filmes

Meus Filmes

127.0.0.1:5500/index.html

Minha lista de Filmes

Novo título

Nova descrição

Adicionar

Lista de **Amanda Marinho (Mandinha)**

☐

Bastardos Inglórios
Filme sobre uma realidade alternativa da Segunda Guerra Mundial!



Tutoriais úteis

VUE

- <https://vuejsbr-docs-next.netlify.app/guide/introduction.html>
- <https://vuejs.org/guide/introduction.html> (Original em inglês)
- <https://vueschool.io/courses>