

Mais propriedades vue e comunicação entre componentes irmãos



DEVinHouse

Parcerias para desenvolver a sua carreira

SENAI

<LAB365>

AGENDA

- O que são watchers
- O que são propriedades computadas
- Ciclo de vida de um componente Vue
- Comunicação entre componentes irmãos através de barramento
- Prática

Watch

A propriedade watch, como diz o nome fica “assistindo/monitorando” o estado de uma variável e executa uma ação ao detectar essa mudança.

Então como monitorar uma variável utilizando a propriedade watch?

É necessário declarar a propriedade watch dentro do escopo do script do componente.

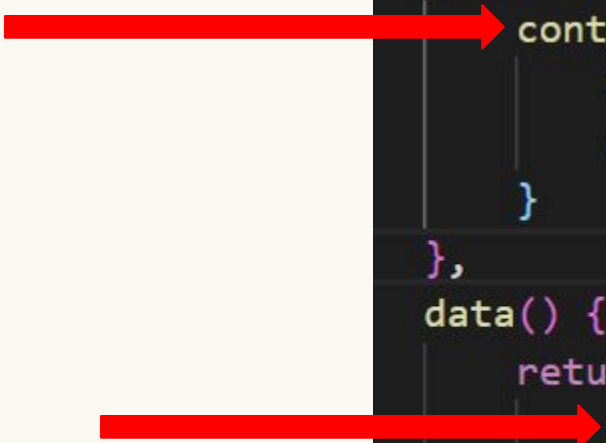


```
<script>
  export default {
    watch: {

    }
  }
</script>
```

watch

Dentro do escopo da propriedade watch, deve ser instanciado um método com o mesmo nome da variável que se deseja monitorar.

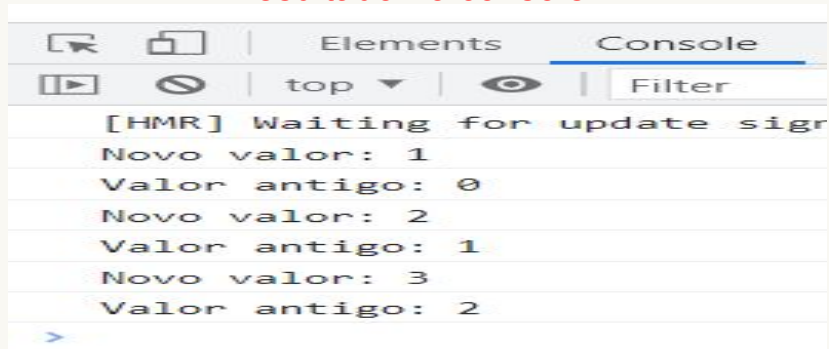


```
watch:{
  contagem(newValue, oldValue) {
    console.log(newValue)
    console.log(oldValue)
  }
},
data() {
  return {
    contagem: 0
  }
}
```

Watch

```
watch:{
  contagem(newValue, oldValue) {
    console.log('Novo valor: ' + newValue)
    console.log('Valor antigo: ' + oldValue)
  }
},
```

Resultado no console



Repare que o método instanciado recebe como parâmetro dois argumentos.

Vamos detalhar esses parâmetros:

- O primeiro parâmetro sempre receberá o estado atual da variável que está sendo monitorada.
- O segundo parâmetro sempre receberá o estado anterior da variável monitorada. Sendo este segundo parâmetro, não obrigatório para o funcionamento do método.

Computadas




Uma propriedade computada serve para processar uma determinada informação e ao término desse processamento retorna uma resposta para que a chama.

Para utilizar do recurso das propriedades computadas deve ser declarada propriedade computed dentro do escopo do script.

```
<script>
export default {
  computed: {
    verificarMaiorIdade() {
      return this.idade >= 18 ? 'É maior de idade': 'É menos de idade';
    }
  }
}
</script>
```

```
<script>
export default {
  computed: {
    verificarMaiorIdade() {
      return this.idade >= 18 ? 'É maior de idade': 'É menos de idade';
    }
  }
}
</script>
```



Atente-se que, o método da propriedade computada **é obrigado a possuir um retorno**, afinal como está sendo feito o processamento de uma informação é esperado uma resposta.

Computadas

```
<template>
  <div class="computadas">
    {{ idade >= 18 ? 'É maior de idade': 'É menor de idade' }}
  </div>
</template>

<script>
export default {
  data() {
    return {
      idade: 18
    }
  }
}
</script>
```

No trecho de código ao lado estamos utilizando uma lógica dentro de uma interpolação para verificar se uma pessoa é maior ou menor de idade .

Para um código pequeno como este de exemplo, digamos que isso até seja aceitável. Mas imagine um componente com uma complexidade maior, cheio de trechos como esse. Além de tornar nosso código menos legível, não é uma boa prática implementar lógicas dessa forma. Então pensando nisso, o Vue nos fornece as propriedades computadas.

Computadas

```
<template>
  <div class="computadas">
    {{ verificarMaiorIdade }}
  </div>
</template>

<script>
export default {
  computed: {
    verificarMaiorIdade() {
      return this.idade >= 18 ? 'É maior de idade': 'É menos de idade';
    }
  },
  data() {
    return {
      idade: 18
    }
  }
}
</script>
```

Repare que a lógica que estava sendo feita na interpolação não está mais sendo feita no escopo do HTML, apenas está sendo interpolado o retorno da propriedade computada.

E que agora o método da computada faz o processamento da lógica e retorna o valor que será exibido na interpolação.

Diferença entre watch e computada

Watch:

- Monitora a mudança de estado de uma variável
- Deve possuir o mesmo nome da variável que ele está monitorando
- Não possui retorno (return)

Computada:

- Faz o processamento lógico de uma variável
- Não deve possuir o mesmo nome de uma variável declarada no atributo data
- É obrigado a possuir retorno.

4. Fase adulta:

Maturidade sexual alcançada, já pode se reproduzir.

3. Crisálida ou pupa:

Dura vários dias.

1. Ovo:

Dura alguns dias ou semanas.

2. Larva:

Possui de 4 a 7 fases.

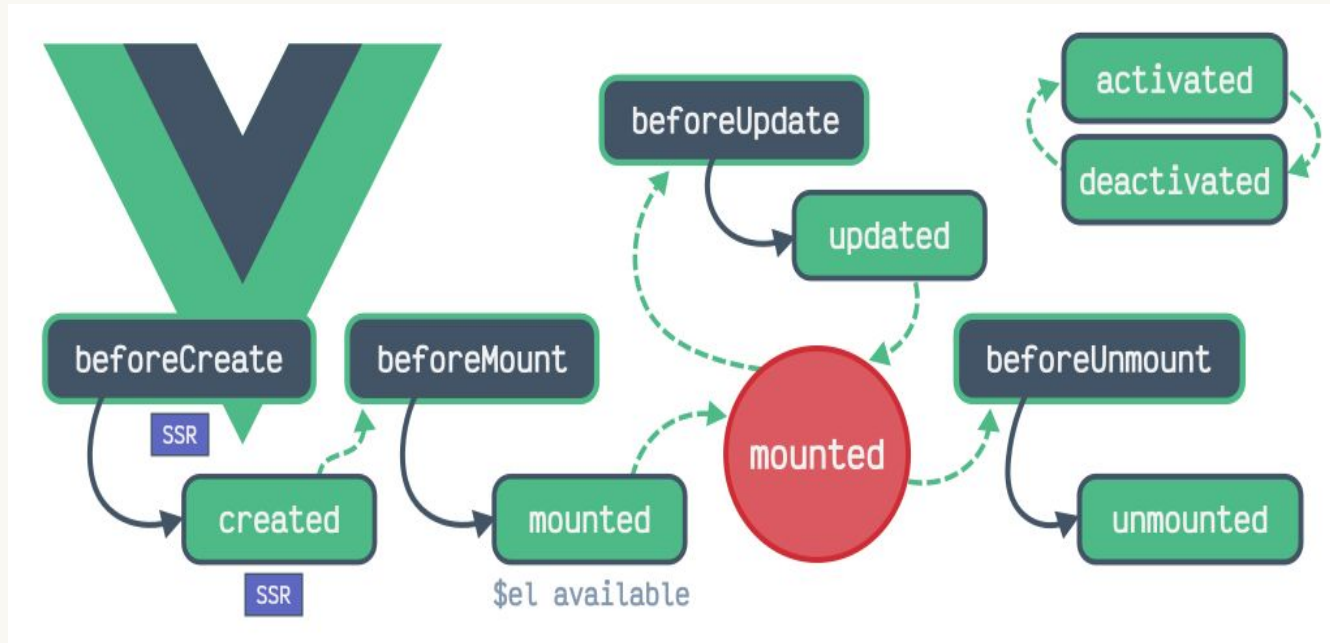


Ciclo de vida

Assim como animais e até o próprio ser humano possui um ciclo de vida, os componentes Vue também possuem.

São eles:

- beforeCreate
- created
- beforeMount
- mounted
- beforeUpdate
- updated
- beforeUnmount
- unmounted



Ciclo de vida - beforeCreate

O beforeCreate acontece apenas uma vez nos momentos iniciais da instânciação do componente. Nesse momento podem ser carregados alguns dados que possam vir a ser necessários para a montagem do componente.

Ciclo de vida - created

Nessa parte do ciclo de criação o componente já possui todos os dados necessários para a sua montagem e renderização. Essa parte do ciclo também só acontece uma vez.

Ciclo de vida - beforeMount

Esse ciclo de vida também acontece apenas uma vez e é o momento que antecede as atribuições dos valores que são feitas às variáveis definidas do componente.

Ciclo de vida - mounted

Aqui o componente já está todo montado e pronto para ser renderizado na tela.

Ciclo de vida - beforeUpdate

Sabe quando utilizamos o v-model para atualizar o valor de uma variável? Pronto, o beforeUpdate acontece antes de renderizar o novo valor dessa variável na tela.

Esse ciclo acontece várias vezes ao longo da vida útil do componente.

Ciclo de vida - updated

Esse ciclo acontece após a renderização das atualizações feitas no componente.

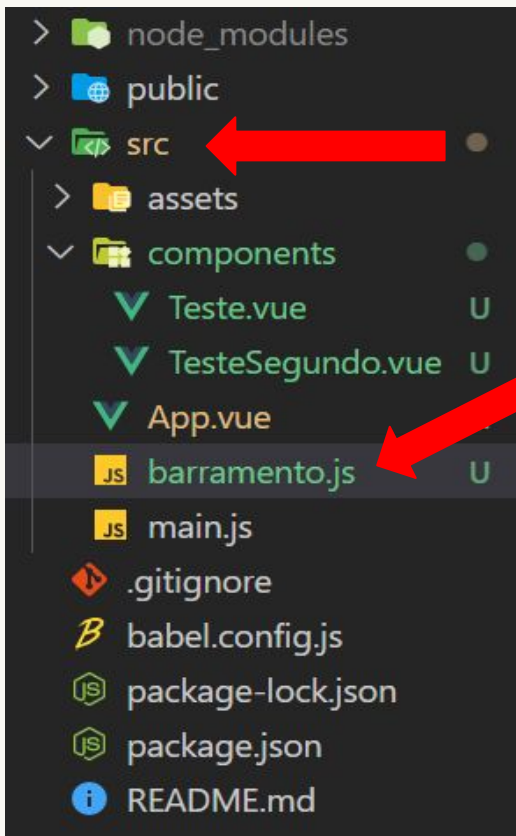
Ciclo de vida - beforeUnmount

Essa parte do ciclo já é o início do fim da vida do componente. Essa parte do ciclo pode ser utilizada para finalizar algum fluxo que por ventura ainda possa está sendo executado como por exemplo o uso de algum outro componente, uma comunicação com a camada de service e etc.

Ciclo de vida - unmounted

Aqui é onde o componente dá seu último suspiro. todas as instâncias necessárias para sua renderização são finalizadas e se porventura alguma comunicação não tiver sido finalizada no ciclo anterior, é finalizada aqui.

Comunicação entre componentes irmãos



Para realizar a comunicação entre componentes irmãos precisamos de uma instância a parte do vue para funcionar como um repositório de informações.

Para criar essa nova instância precisamos:

1. Criar um novo arquivo javaScript na pasta scr
2. Importar o Vue
3. Exportar a nova instância do Vue

```
src > JS barramento.js > [⌨] default
1   import Vue from 'vue'
2   export default new Vue()
```

Comunicação entre componentes irmãos

```
<script>
import barramento from '../barramento'
export default {
  data() {
    return {
      idade: null
    }
  },
  methods: {
    enviar() {
      barramento.$emit('teste', this.idade)
    }
  }
}
```



Então usaremos praticamente a mesma lógica utilizada na emissão de eventos para propagar o valor que queremos. Só precisaremos fazer duas coisas diferentes:

1. Precisamos importar o arquivo barramento.js
2. E na emissão do evento não utilizaremos o this.\$emit() mas sim o barramento.\$emit()

Comunicação entre componentes irmãos

```
<script>
import barramento from '../barramento'

export default {
  data() {
    return {
      idade: null
    }
  },
  computed: {
    verificarMaiorIdade() {
      return this.idade >= 18 ? 'É maior de idade': 'É menor de idade';
    }
  },
  mounted() {
    barramento.$on('teste', idade => {
      this.idade = idade
    })
  }
}
</script>
```



Para ouvir o evento emitido pelo irmão precisamos fazer o seguinte:

1. Precisamos importar o arquivo barramento.js
2. E para ouvir o evento emitido pelo irmão utilizamos, dentro de um ciclo de vida, a instrução `barramento.$on(parametro1, parametro2)`

Repare que o método `$on` possui dois argumentos. O primeiro deve ser preenchido com o nome do evento que se deseja ouvir e o segundo com uma arrow function para executar alguma ação.

Dúvidas?





DEVinHouse

Parcerias para desenvolver a sua carreira

OBRIGADO!



<LAB365>