A Instância Vue

Criando a Instância Vue

Toda aplicação Vue é iniciada com a criação de uma nova instância Vue com a função Vue:

```
var vm = new Vue({
  // opções
})
```

Embora não seja estritamente associado com o padrão MVVM, o design do Vue foi parcialmente inspirado por ele. Como convenção, muitas vezes usamos a variável vm (abreviação de ViewModel) para se referir à instância Vue.
Quando você cria uma instância Vue, é necessário passar um objeto de opções. A maior parte deste guia descreve como você pode utilizar estas opções para criar os comportamentos desejados. Para referência, você também pode navegar pela lista completa de opções na documentação da API.
Uma aplicação Vue consiste em uma instância Vue raiz criada com new Vue, opcionalmente organizada em uma árvore de componentes reutilizáveis aninhados. Por exemplo, um aplicativo de tarefas a realizar (do tipo todo list) poderia ter uma árvore de componentes como esta:

```
Instância Raiz
└── TodoList
    ├── TodoItem
    |   ├── TodoButtonDelete
    |   └── TodoButtonEdit
    └── TodoListFooter
        ├── TodosButtonClear
        └── TodoListStatistics
```

Falaremos sobre o sistema de componentes em detalhes futuramente. Por enquanto, saiba apenas que todos os componentes Vue também são instâncias, e aceitam o mesmo esquema de opções (exceto por algumas poucas opções específicas da raiz).
Dados e Métodos

Quando uma instância Vue é criada, ela adiciona todas as propriedades encontradas no objeto data ao sistema de reatividade do Vue. Quando os valores de qualquer destas propriedades muda, a camada visual “reage”, atualizando-se para condizer com os novos valores.

```
// Nosso objeto de dados
var data = { a: 1 }

// O objeto é adicionado à instância Vue
var vm = new Vue({
  data: data
})

// É uma referência ao mesmo objeto!
vm.a === data.a // => true

// Atribuir à propriedade na instância
// também afeta o dado original
vm.a = 2
data.a // => 2

// ... e vice-versa
data.a = 3
vm.a // => 3
```

Quando este dado for modificado, a camada visual irá “re-renderizar”. Deve-se observar que propriedades em data são reativas somente se já existem desde quando a instância foi criada. Significa que se você adicionar uma nova propriedade, como:
vm.b = 'hi'
Então as mudanças em b não irão disparar qualquer atualização na interface. Se você sabe que precisará de uma propriedade no futuro, mas ela inicia vazia ou não existente, precisará especificar algum valor inicial. Por exemplo:

```
data: {
  newTodoText: '',
  visitCount: 0,
  hideCompletedTodos: false,
  todos: [],
  error: null
}
```

A única exceção é ao usar Object.freeze(), que previne propriedades existentes de serem modificadas. Também significa que o sistema de reatividade não pode rastrear mudanças.

```
var obj = {
  foo: 'bar'
}

Object.freeze(obj)

new Vue({
  el: '#app',
  data: obj
})
<div id="app">
  <p>{{ foo }}</p>
  <!-- não irá atualizar mais o `foo`! -->
  <button v-on:click="foo = 'baz'">Change it</button>
</div>
```

Em adição às propriedades de dados, instâncias Vue expõem uma quantidade relevante de propriedades e métodos. Estes são diferenciados pelo prefixo $ para não confundi-los com propriedades definidas pelo usuário. Por exemplo:

```
var data = { a: 1 }
var vm = new Vue({
  el: '#exemplo',
  data: data
})

vm.$data === data // => true
vm.$el === document.getElementById('exemplo') // => true

// $watch é um método da instância
vm.$watch('a', function (newValue, oldValue) {
  // Esta função será executada quando `vm.a` mudar
})
```

No futuro, você pode consultar a documentação da API para a lista completa de propriedades e métodos da instância.
Ciclo de Vida da Instância

Assista a uma lição em vídeo gratuita na Vue School
Cada instância Vue passa por uma série de etapas em sua inicialização - por exemplo, é necessário configurar a observação de dados, compilar o template, montar a instância no DOM, atualizar o DOM quando os dados forem alterados. Ao longo do caminho, ocorrerá a invocação de alguns gatilhos de ciclo de vida, oferecendo a oportunidade de executar lógicas personalizadas em etapas específicas.
Por exemplo, o gatilho created pode ser utilizado para executar código logo após a instância ser criada:

```
new Vue({
```

```
  data: {
    a: 1
  },
  created: function () {
    // `this` aponta para a instância
    console.log('a é: ' + this.a)
  }
})
// => "a é: 1"
```

Existem outros gatilhos (em inglês, hooks) chamados em diferentes etapas do ciclo de vida da instância, como mounted, updated e destroyed. Qualquer gatilho de ciclo de vida é executado com seu contexto this apontando para a instância Vue que o invocou.

Não utilize arrow functions em propriedades de opções ou callback, como em created: () => console.log(this.a) ou vm.$watch('a', newValue => this.myMethod()). Como as arrow functions não tem um this,this será tratado como qualquer outra variável e lexicamente pesquisada através de escopos parentais até ser encontrada, frequentemente resultando em erros como Uncaught TypeError: Cannot read property of undefined ou Uncaught TypeError: this.myMethod is not a function.