

# Vue



Aluno: Gabriel Godoy da Silva

## Vue: Histórico

- **Vue** foi criado por Evan You, após trabalhar na Google com o **Angular** em diversos projetos.
  - ◆ Segundo o próprio, sobre a criação do **Vue**: “Eu pensei, e se eu pudesse extrair a parte que eu realmente gosto do **Angular** e construir algo realmente leve?” (tradução minha).
- **Vue** teve seu primeiro commit no código-fonte em **junho de 2013** e teve seu primeiro anúncio público em **fevereiro de 2014**.

## Vue: Nomenclatura

- O nome “Vue” é uma forma de “**erro ortográfico intencional**”, fenômeno comum na língua inglesa. Tem o mesmo significado de “**View**”, ou **vista**.
- ◆ Um exemplo clássico desse fenômeno é a palavra “**ok**”, que vem de “**oll korrekt**”, mesmo significado de “**all correct**”, ou “**tudo certo**”.
  - ◆ Outros casos em nomes de produtos e marcas seriam “**Tumblr**” (“**tumblelog**”, forma de blog mais curta) e “**Google**” (“**Googol**” =  $10^{100}$ ).

# Vue: Nomenclatura de versões

→ Versões em **Vue** tem o costume de serem nomeados com títulos de anime/manga.

◆ São alguns exemplos de versões:

- **0.11 - Cowboy Bebop** - 07/11/2014
- **0.12 - Dragon Ball** - 12/06/2015
- **1.0 - Evangelion** - 27/10/2015
- **2.3 - JoJo's Bizarre Adventure** - 27/04/2017
- **2.7 - Naruto** - 01/06/2022
- **3.3 - Rurouni Kenshin** - 11/05/2023
- **3.5 (atual) - Tengen Toppa Gurren Lagann** - 01/09/2024

# Vue: Um framework progressivo

- Vue se propõe a ser uma alternativa flexível, que se adapta ao seu projeto; proposta diferente de outros frameworks, como Angular.
- Essa característica torna Vue uma ferramenta interessante para:
  - ◆ Iniciantes, que podem depender mais de certas adaptações em JS vanilla (JS puro).
  - ◆ Projetos que exijam maior integração com outros frameworks.
  - ◆ Projetos incrementais, de um modo geral.

# Reatividade em Vue

- A forma mais recomendada de declarar um estado reativo é usar a função “**ref()**”.
  - Atribui à variável um campo “**.valor**”, que é reativo.
  - Pode receber **qualquer tipo primitivo** como parâmetro.
  - Permite reatividade profunda, mas somente de objetos primitivos.
  - Usa “getters” e “setters” para interceptar mudanças de estado no JS.
- Alterações em estados reativos atualizam o DOM automaticamente.

# Reatividade em Vue

- Outra forma de declarar estados reativos é usando a API “**reactive()**”
  - **Torna a própria variável reativa**, sem necessidade do campo “.valor”.
  - **Permite reatividade profunda, somente com tipos não primitivos.**
  - Cria um “**proxy**” para realizar a reatividade (somente o proxy sendo reativo).
  - Tem mais limitações que “**ref()**”, como:
    - Só funciona com objetos, não primitivos.
    - Não permite reposição de objetos (sem perder conectividade reativa).
    - Não é amigável à desestruturação.

# Componentes **Vue**: Definição

- Componentes em **Vue** funcionam como “**blocos de montar poderosos**”, com possibilidade de armazenamento de estado e dinamismo com reações.
  - ◆ Cada componente serve uma função e é parte da estrutura de modularização do framework.
  - ◆ Ao mesmo tempo, eventos podem alterar o estado de componentes já existentes.
- Componentes podem ser declarados em um arquivo exclusivo (Single-File Component, ou SFC) ou diretamente no arquivo **JS**.



# Componentes em arquivo único (SFC): Definição

- Apesar de não ser um conceito único ao **Vue**, é creditado por popularizar o conceito no contexto de frameworks web.
- Forma recomendada de usar **Vue** em diversos cenários, por exemplo:
  - ◆ Aplicações de página única (SPA).
  - ◆ Geração estática de sites (SSG).
- Agrupa conteúdo (pseudo-**HTML**), formatação (**CSS**) e lógica (**JS**) do site em um arquivo único (**\*.vue**).

## Componentes em arquivo único (SFC): Vantagens

- Escrita de componentes modularizados, utilizando sintaxe **HTML**, **CSS** e **JS**.
- Pré-compilação de templates sem custo de compilação em tempo de execução.
- **CSS** com escopo vinculado a componente.
- Mais otimizações em tempo de compilação, analisando template e script em conjunto.

# Componentes em arquivo único (SFC): Exemplo (site)

```
<script setup>

import { ref } from 'vue'

const count = ref(0)

</script>

<template>

  <button @click="count++">Count is: {{ count }}</button>

</template>

<style scoped>

button { font-weight: bold; }

</style>
```

# Componentes em arquivo único (SFC): Execução (site)

The screenshot shows a web browser interface for a Vue.js Single File Component (SFC). The top navigation bar includes links for Docs, API, Playground, Ecosystem, About, Sponsor, and Experts. The main content area is divided into three sections:

- JS:** Contains the Vue.js application code:

```
import { createApp, ref } from 'vue'  
  
createApp({  
  setup() {  
    return {  
      count: ref(0)  
    }  
  }  
}).mount('#app')
```
- template:** Contains the HTML template for the component:

```
<div id="app">  
  <button @click="count++">  
    Count is: {{ count }}  
  </button>  
</div>
```
- Result:** Displays the rendered component, showing a button with the text "Count is: 39".

On the right side, there is a panel with two tabs: "Elementos" (Elements) and "Filtros" (Filters). The "Elementos" tab is active, showing a tree view of the component's DOM structure. The "Filtros" tab is also visible, showing a list of filters. The bottom of the page features a footer with the text "Estilos" and "Calculado".

## Vue CDN: Definição

- **Content Delivery Network.**
- Basta adicionar um `<script>` no arquivo html para carregar o [Vue](#).
- Na prática, um desenvolvimento web tradicional simplificado, por não haver otimização ou pré-processamento padrão.
- Fácil uso e pequeno overhead, muito bom para a utilização em sistemas legado.

## Vue CLI: Definição

- **Command Line Interface.**
- Ambiente próprio do **Vue** para programar utilizando linhas de comando.
- Permite ajustes ao código de forma mais rápida, otimizados e pré-processados.

# Sintaxe declarativa

- **Vue** usa sintaxe declarativa (“o que” > “como”), gerando uma interface mais amigável ao programador.
- ◆ Lidar com a natureza das operações é algo mais simples do que pensar em cada detalhe de como realizá-la.
- ◆ Alguns exemplos:
  - **v-if**: Operação de condicional
  - **v-else**: Operação de condicional
  - **v-for**: Operação de loop (iteração sobre objeto)