# Vue.js - Reatividade

ref() • computed() • watch() • watchEffect()

Professor: F.Laurindo Costa jr

# Sumário

Visão geral dos quatro conceitos reativos principais do Vue.js

## 1. ref() — Estado Reativo

Criação de objetos reativos, detecção automática de mudanças e atualização da interface

## 2. computed() — Valores Derivados

Valores calculados com cache automático, atualizados apenas quando suas dependências mudam

## 3. watch() — Reagir a Mudanças Específicas

Observação de valores reativos específicos e execução de callbacks em resposta a mudanças

## 4. watchEffect() — Dependências Automáticas

Detecção automática de dependências e reexecução de função quando qualquer uma mudar

# </> ref() - Estado Reativo

## O que é ref()?

**ref()** cria um objeto reativo que guarda um valor interno acessível via **.value**.
Sempre que **.value** muda, o Vue atualiza a interface automaticamente.

## Exemplo Prático

```
// consultas.store.ts
const consultas = ref<Consulta[]>([])
// Guarda a lista de consultas. Quando a store faz:
consultas.value.push(...) a lista atualiza na UI
```

## Pontos Importantes

- Necessário acessar o valor interno via **.value**
- Detecção automática de mudanças para atualizar a UI
- Pode ser usado para variáveis, listas e objetos

# </> ref() - Estado Reativo

## Curiosidade:

**ref<Consulta[ ]>([ ])**

**ref<TIPO>(VALOR_INICIAL)**

**Então:**

- `Consulta[]` → tipo: uma lista (array) de objetos Consulta
- `[]` → valor inicial: uma lista vazia
- `ref<T>()` → diz ao Vue e ao TypeScript qual será o tipo armazenado

**Por que isso é útil?**

Porque sem `<Consulta[]>`, o TypeScript iria tratar o array como: **any[ ]**

E o editor **não saberia** que cada item tem:

- pacienteId
- medicoId
- data
- horario
- motivo
- status

  etc.

Com `<Consulta[]>`, você ganha:

- autocomplete
- validação em tempo real
- erros quando tenta acessar campo inexistente
- mais segurança no código

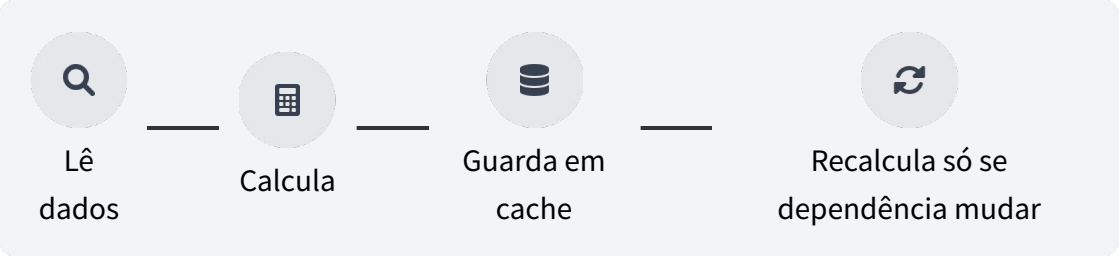
**“`ref<Consulta[]>([])` = “uma caixa reativa que guarda uma lista de Consultas”..”**

# computed() - Valores Derivados

## Definição

Cria valores calculados, que só mudam quando suas dependências mudam. É cacheado automaticamente pelo Vue.

## Fluxo Mental




## Exemplos

```
// Exemplo básico: duplica o valor de count
const doubleCount = computed(() => {
  return count.value * 2;
});
```

O valor é recalculado automaticamente quando **count.value** muda

```
// Exemplo do projeto: filtra consultas concluídas
const consultasConcluidas = computed(() => {
  return consultas.value.filter(c => c.status ==="CONCLUIDA");
});
```

O resultado é automaticamente atualizado quando **consultas.value** muda

# watch() - Reagir a Mudanças Específicas

## Definição

watch() observa valores reativos específicos e executa um callback quando eles mudam.

## Casos de Uso

- Validação de formulário
- Buscar dados ao mudar um select
- Preencher formulário ao entrar no modo de edição
- Sincronizar estados externos
- Salvar rascunhos

## Fluxo mental

1. O aluno muda algo na tela
2. A variável reativa muda
3. O Vue detecta a mudança
4. O Vue chama o callback

O callback é **a resposta à mudança**.

## Exemplo Prático

```
// Detectar quando uma consulta entra em modo de edição
const consultaStore = useConsultaStore()
const { editingConsulta } = storeToRefs(consultaStore)

watch(editingConsulta, (novaConsulta) => {
    if (novaConsulta) {
        form.pacienteId = novaConsulta.pacienteId
        form.medicoId = novaConsulta.medicoId
        form.data = novaConsulta.data
        form.horario = novaConsulta.horario
        form.motivo = novaConsulta.motivo
    }
})
```

### Explicação:

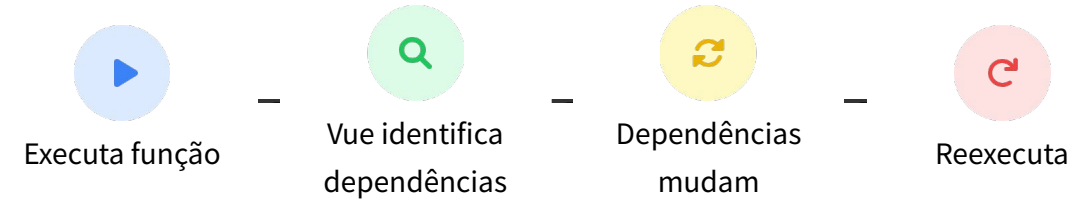
- Quando **editingConsulta** muda no store...
- ...o **watch** detecta a mudança...
- ...e preenche o formulário com os dados da consulta.

# watchEffect() - Dependências Automáticas

## Definição

Executa uma função imediatamente e a reexecuta toda vez que qualquer dependência reativa usada dentro dela mudar. Não precisa declarar o que observar.

## Fluxo Mental




### Vantagem

O Vue descobre sozinho quais dependências estão sendo usadas dentro da função, sem precisar declará-las explicitamente como no watch().

## Exemplo Prático

```
watchEffect(() => {
  console.log('Consulta em edição:', editingConsulta.value)
})
```

### O que acontece:

- Função é executada imediatamente
- O Vue observa que

# Resumo Comparativo

Guia prático para escolher o conceito reativo certo

| | Conceito | Serve para | Quando usar |
|---|---|---|---|
|  | **ref()** | Estado reativo | Variáveis, listas, objetos |
|  | **computed()** | Valores derivados + cacheados | Filtrar, totalizar, formatações |
|  | **watch()** | Reagir a mudanças específicas | Preencher formulário, validar, buscar dados |
|  | **watchEffect()** | Executar lógica sempre que dependência mudar (auto) | Debug, sincronização simples |