# VUE

# Variáveis

# Reativas

Bernardo Meyer

VUE – Conceito de variável reativa

O framework VUE implementa o conceito de **Variável Reativa**. O que é uma variável reativa ?

Durante a execução de um programa, no interior da memória do dispositivo eletrônico programável, uma variável de programa só tem seu valor modificado se estiver à esquerda de uma atribuição. Na primeira fase da Tecnologia de Informação, existia um programa em execução e uma console de inspeção das variáveis deste programa. Os resultados também eram apresentados na console.

Na segunda fase de programação, separou-se o programa do Banco de dados.

Depois, na terceira fase da Tecnologia de Informação, existia uma interface permanente de apresentação – não mais uma console de múltipla atribuição – da informação, da forma mais dinâmica possível. Para alterar o valor das variáveis, era preciso que um evento ocorresse. Mas, mesmo assim, o valor exibido na interface poderia estar desatualizado em relação ao valor da memória, pois era preciso um esforço de programação para sincronizar o calculado com o exibido.

E agora, na quarta fase da Tecnologia da Informação, frameworks que operam na interface foram desenvolvidas para sincronizar o calculado com o exibido. Tal é o caso do VUE, entre outros, como ***Angular, React, Kendo e jQuery***. Poderíamos deixar o ***jQuery*** em uma posição intermediária entre a terceira e quarta fase, pois ele não implementa uma interface reativa tão eficiente quanto os demais.

Em outras palavras, na interface reativa, se um **INPUT** altera uma variável a ele acoplada, a alteração deste **INPUT** se reflete no programa a ele associado, e vice-versa, caso tenha sido especificado. Da mesma forma, se uma **tag HTML** apresenta um conteúdo dependente do valor desta variável, este conteúdo também será modificado.

Vamos analisar a implementação deste modelo durante o debug de um exemplo bem simples:

```
<body>
  <div id="app">{{ message }}</div>
  <script type="text/javascript">
    new Vue({
      el: "#app",
      data: { message: "Hello World!" }
    });
  </script>
</body>
```

VUE – Conceito de variável reativa

# Inicialização

Na instanciação do modelo **Vue**, que consta da **DIV#app** (DIV cujo Id é “app”), entre outras coisas que acontecem, é coletada a variável Vue “message”. Isto é feito pela função initData, segundo o seguinte percurso:

**Instanciação**: new Vue
Chama this._init(options)
options contém a seção data, com os valores iniciais das variáveis
Chama Vue.prototype._init
Função definida dentro de initMixin(Vue)
Chama resolveConstructorOptions (Options)
Chama mergeOptions (une ***instanceData*** com ***defaultData***):
Chama initProxy(vm /* instancia de Vue */)
Coloca Proxy(vm, handlers) em vm._renderProxy

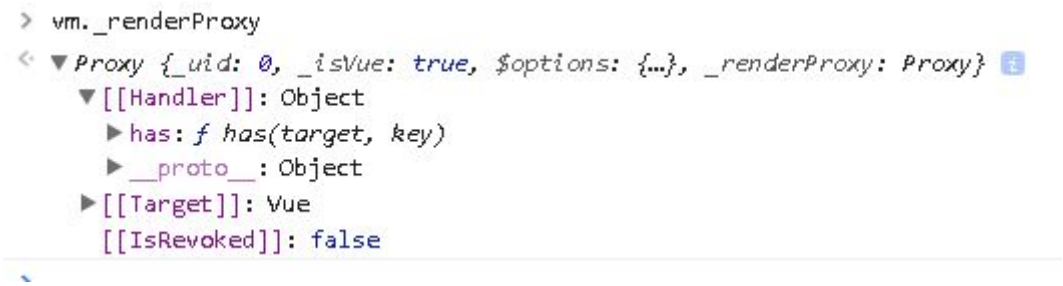



*Figura 1: Objeto Proxy básico preparado para receber variáveis.*

Chama initLifeCycle(vm)
Chama initEvents(vm)
Chama initRender(vm)
Chama callHook(vm, 'beforeCreate')
Chama initInjections(vm)

VUE – Conceito de variável reativa

Chama initState(vm)

Chama initData(vm)

Chama getData(data, vm)

Chama mergedInstanceDataFn

Retorna instanceData com defaultData (que redunda em message: “Hello World”

Chama proxy(target, sourceKey, key)

O proxy define uma estrutura básica de objeto para as variáveis:

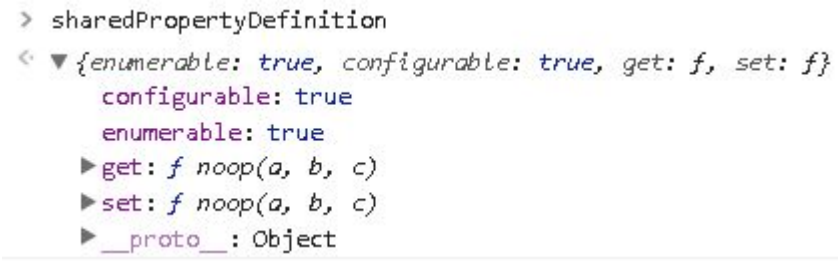



*Figura 2: Protótipo de variável Proxy*

Ao final da função, o protótipo fica da seguinte forma:

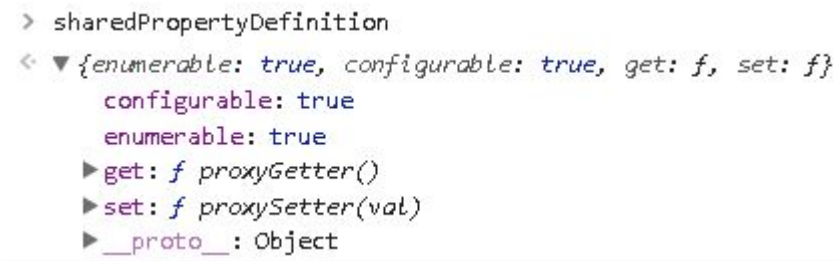
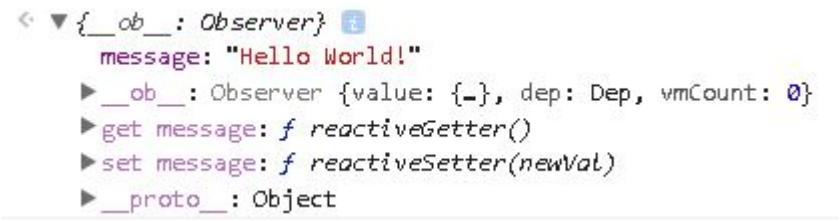



*Figura 3: A função noop do protótipo é substituída por proxyGetter e proxySetter, respectivamente em get e set.*

Chama observe(data, true), onde data contém ‘**message: “Hello World”**’

A função produz o objeto Observer:




A diferença entre o objeto **Proxy** e o **Observer** é a de que os métodos **get** e **set** invocam funções mais elaboradas (***reactiveGetter*** e ***reactiveSetter***). A forma final do **observer** é:

VUE – Conceito de variável reativa

# Uso da variável

Repetindo: a seção data de options, definida na instanciação do objeto Vue, contém os valores INICIAIS das variáveis. No corpo do HTML esta variável poderá ou não ser utilizada. E para isto, será necessária a compilação do contexto onde ela será usada (DIV#app).

Mostraremos o processo em dois momentos:

- Quando o contexto é compilado;
- Quando o contexto é montado e renderizado.

# Compilação do trecho texto

Para não complicar, mostraremos o momento do parse sobre o trecho "{{ message }}".

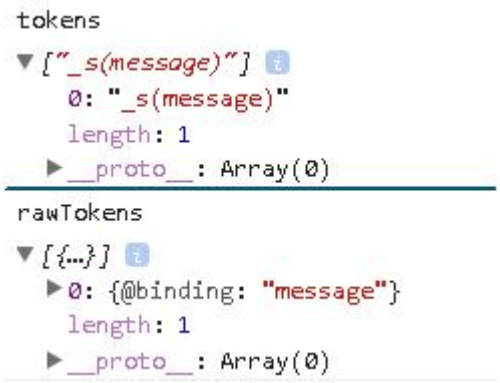
A compilação é disparada pelo processo Vue.$mount, até chegar em **parseHTML**.

"*{{ message }}*" é colocada na variável **text**. É chamado o método **options.chars**(***text***). Que é método de **parseHTML**, o qual chamamos. Este método (options.chars) chama **parseText**.

É feita uma operação Regexp com tagRE como expressão regular sobre text, até que se esgotem as variáveis Vue, resultando em:


*Figura 4: Exemplo com uma só variável para tornar a explicação compreensível.*

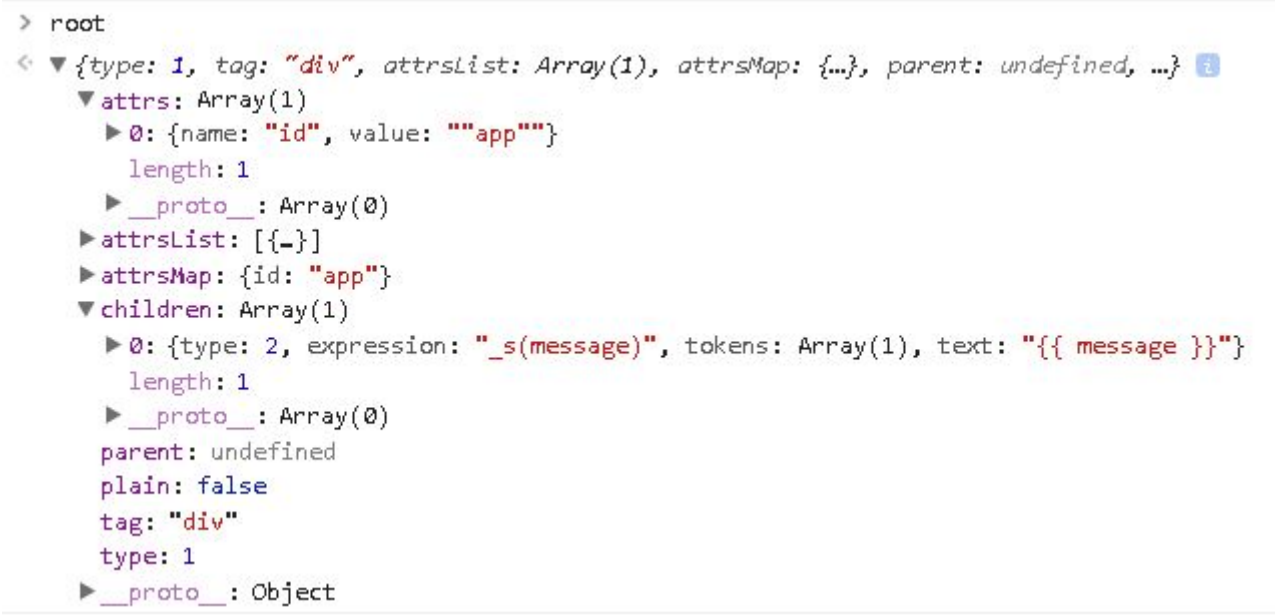
Ao final da análise deste trecho, duas estruturas recebem valores: tokens e rawTokens. Ambas são arrays:




Em **tokens** vemos a chamada de função a ser usada na renderização.

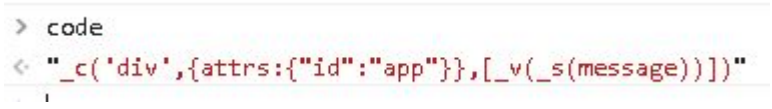
VUE – Conceito de variável reativa

O produto da compilação é a seguinte estrutura:




Está registrado que a tag de contexto é **DIV** (tag: “div”), cujo **id** é “app” e que os nós filhos são **children**, com a expressão “***message***”.

Com este código, o Vue chama a função ***generate(ast, options)***, onde ast é a estrutura mostrada anteriormente (**root**). Esta acaba chamando a função genElement para a porção “div”, determinante do contexto. O resultado é:



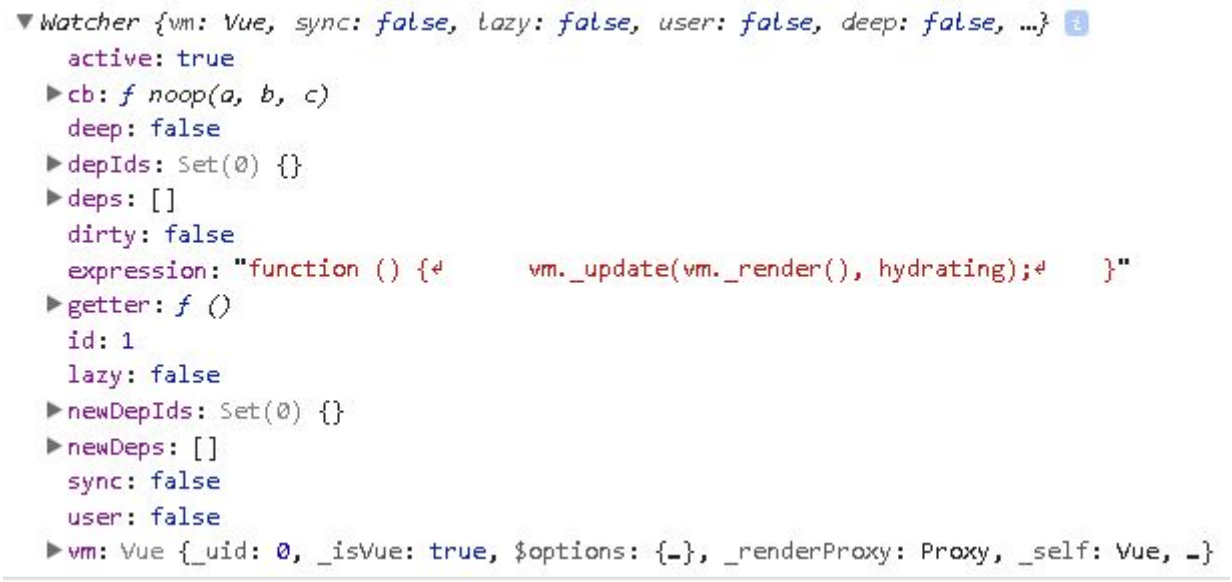

Este é o código em que foi transformado o contexto HTML onde o **Vue** vai agir. Mas a função **generate** ainda acrescenta mais um trecho, antes do retorno:

“with(this){return _c('div',{attrs:{"id":"app"}},[_v(_s(message))])}"

VUE – Conceito de variável reativa

A função compile, após o retorno daquelas que chamou, devolve uma estrutura AST, como segue:

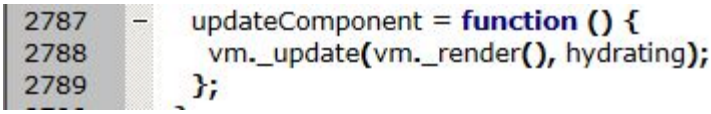

Quem faz a atualização dos valores são os Watchers, atualizados pela função mountComponent:



VUE – Conceito de variável reativa

A função getter está definida, nesta estrutura, como:

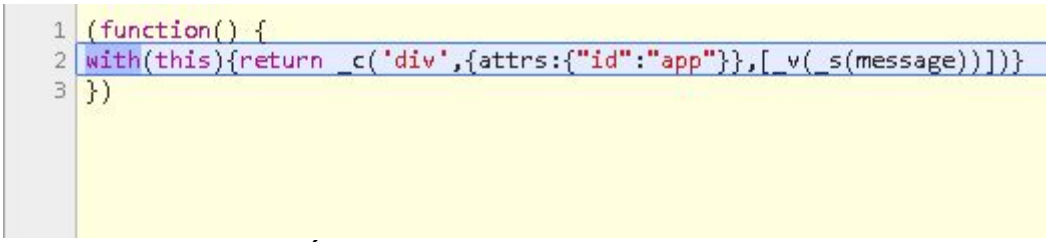

onde se define o local como a linha 2787 da biblioteca Vue utilizada:



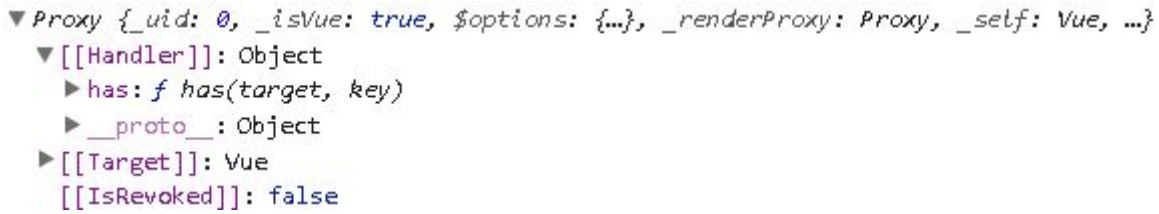

O processamento continua em **Vue.prototype._render**. O elemento chave da renderização é a chamada:

vnode = render.call(vm._renderProxy, vm.$createElement);

Quando esta função é chamada, vm._renderProxy aponta para:




E quem é o this neste contexto ? É o objeto Proxy:



VUE – Conceito de variável reativa

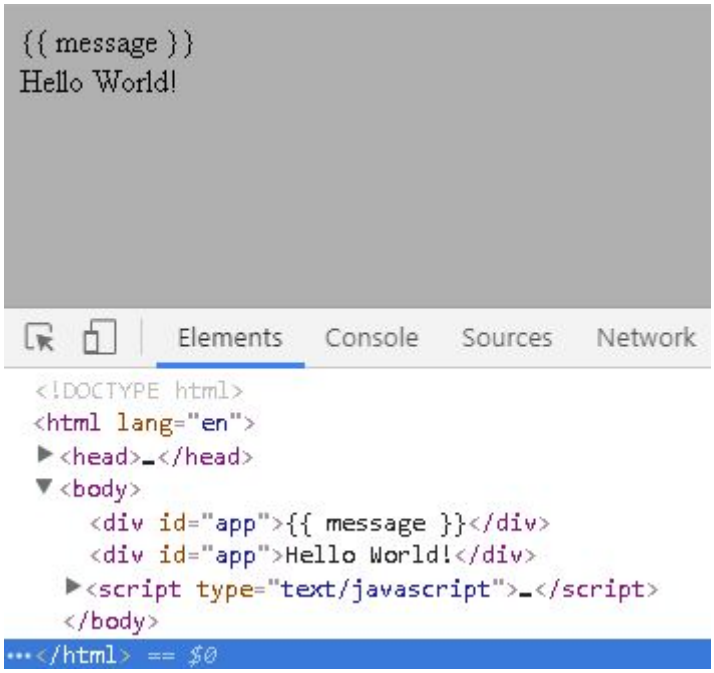
onde Target é:

VUE – Conceito de variável reativa

Na renderização são chamadas as funções:

_s( invoca a função toString;
_v( invoca a função createTextVNode;
_c( invoca a função createElement;

O efeito desta função de renderização é gerar um nó a mais no document de nossa página:




Este processo é efetuado pela função **patch**, chamada por **Vue._update**, chamada por **updateComponent**, chamada pelo **get** de **Watcher**, chamada por **mountComponent**, que é chamada por **Vue.$mount**. O primeiro nó DIV contém a template como digitada pelo programador. O segundo nó DIV possui o produto da renderização. Posteriormente, ele é removido, fornecendo o resultado desejado, que é a frase "***Hello World***" em nosso navegador.

Chegamos ao resultado que desejávamos.