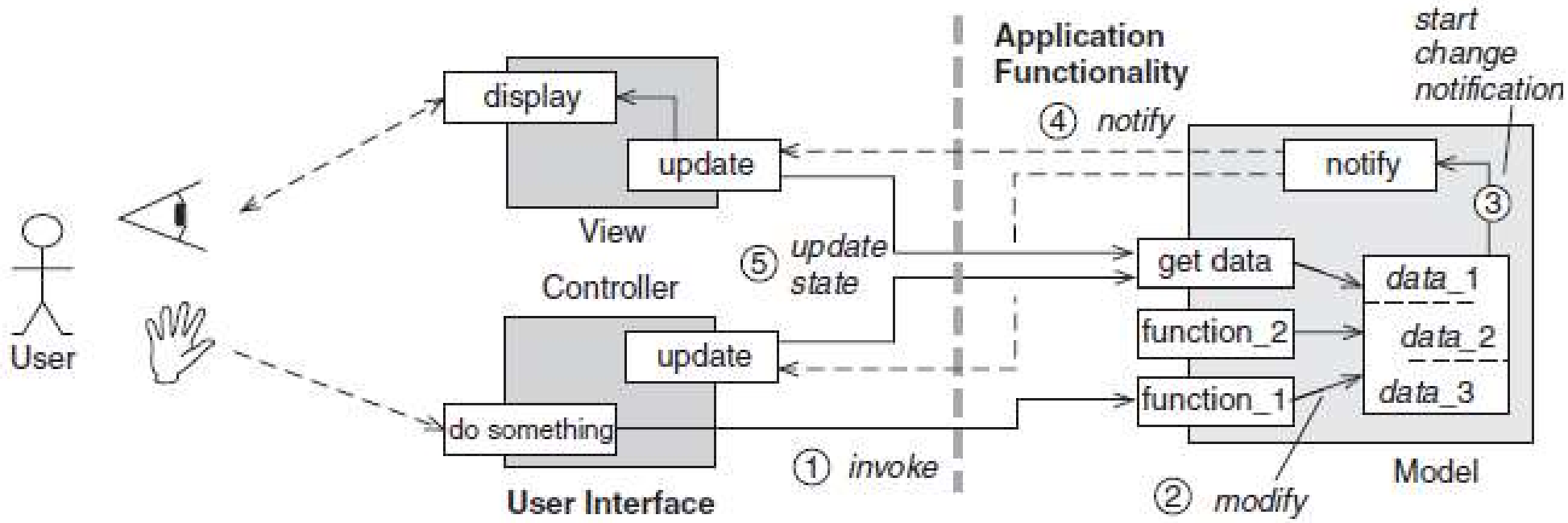
# Vue.js

- Framework para a criação de interface de usuário do lado cliente
- Baseado em JavaScript
- Baseado no conceito componentes
  - Detecta mudanças de estados (eventos) e realiza updates do documento (utilizando o conceito de DOM)
  - Componentes podem conter código HTML, CSS ou JavaScript
  - Um "objeto" vue é associado ao HTML e quando o objeto muda, o código HTML acompanha
  - Utiliza {{ }} como um *placeholder* para dados
- Compacto
- Alguns usuários
  - Huawei, Alibaba.com, Apple, Adobe, Google, Microsoft, Netflix, BMW
- Maiores informações
  - https://v3.vuejs.org/guide/introduction.html

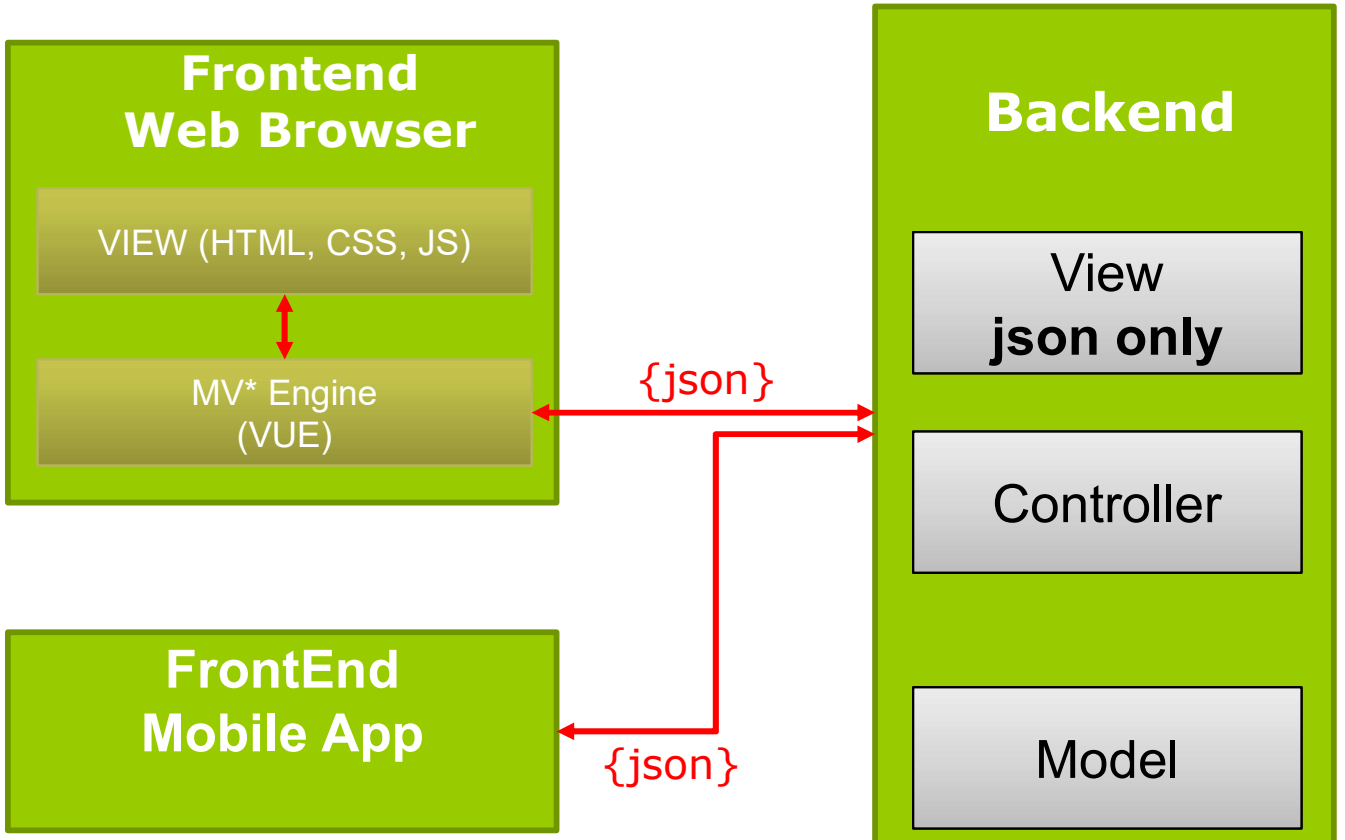
# Model-View-Controller

- Uso
  - Em situações onde a interface do usuário de uma aplicação pode mudar de forma mais frequente que o seu domínio



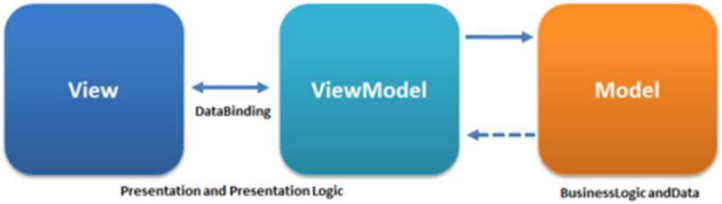
# Model-View-*
# Multiplas Plataformas no Frontend

- Uma estratégia é utilizar no Frontend web alguma engine MV* para preparar o conteúdo para ser exibido no browser e realizar ligação entre o backend e o que é visto no Frontend pelo usuário
- Neste caso uma representação neutra e livre de marcações HTML/CSS, o JSON, é devolvida e processada corretamente por cada plataforma de Frontend (mobile ou web)



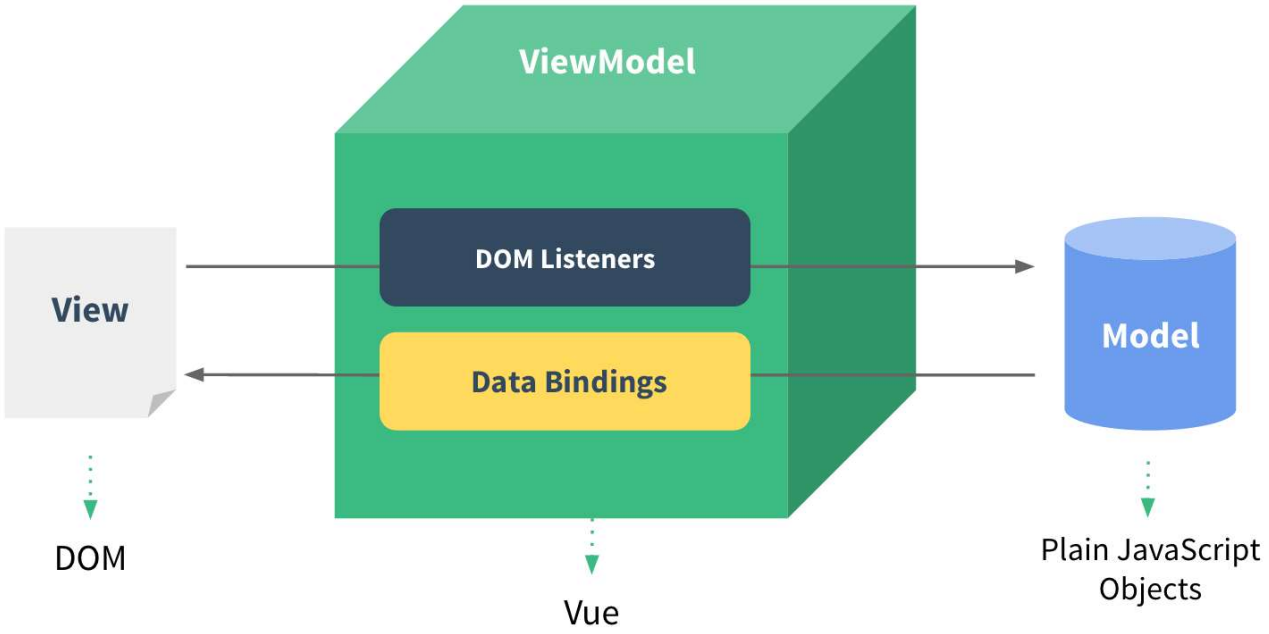
# Model-View-ViewModel (MVVM)

- O VUE trabalha com um conceito do Model-View-Controller (MVC) aplicado somente à interface gráfica.
- Este padrão baseado no MVC é chamado Model-View-ViewModel (MVVM)
  - Oferece uma ligação de mão dupla entre o view e o ViewModel
  - O ViewModel Utiliza o padrão **observer** para propagar mudanças
  - Model
    - Contém os dados e a lógica relacionada. Os dados são transferidos entre
  - View
    - São componente da UI como HTML, CSS, elementos de frameworks como JQuery ou BootStrap
    - É responsável por mostrar os dados recebidos do ViewModel
  - ViewModel
    - É responsável por apresentar as funções e métodos que suportam o estado da View. Opera sobre o Model e ativa eventos da View.
  - Mais informações: https://www.guru99.com/mvc-vs-mvvm.html



# VUE Model

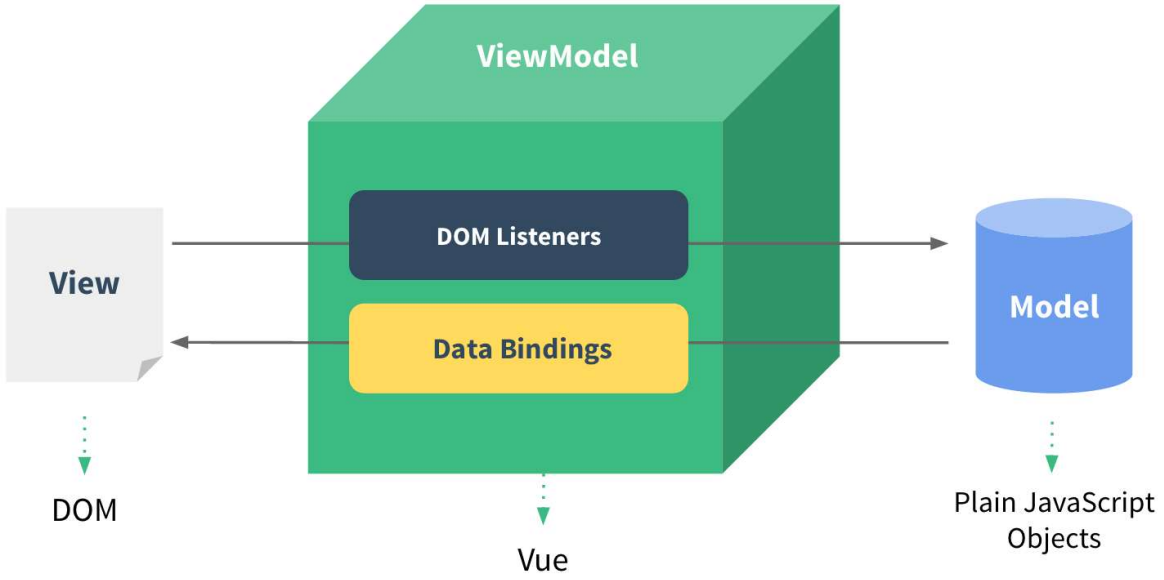
- O VUE trabalha com o Model-View-Controller (MVC) aplicado somente à interface gráfica.
- VUE é um sistema reativo que mantém o Model e View em sincronismo
- Através de uma sintaxe especial no em “templates” HTML é possível ligar o Document Object Model (DOM) com os dados.
- Modificações nos dados são transferidas para a view



# VUE Model

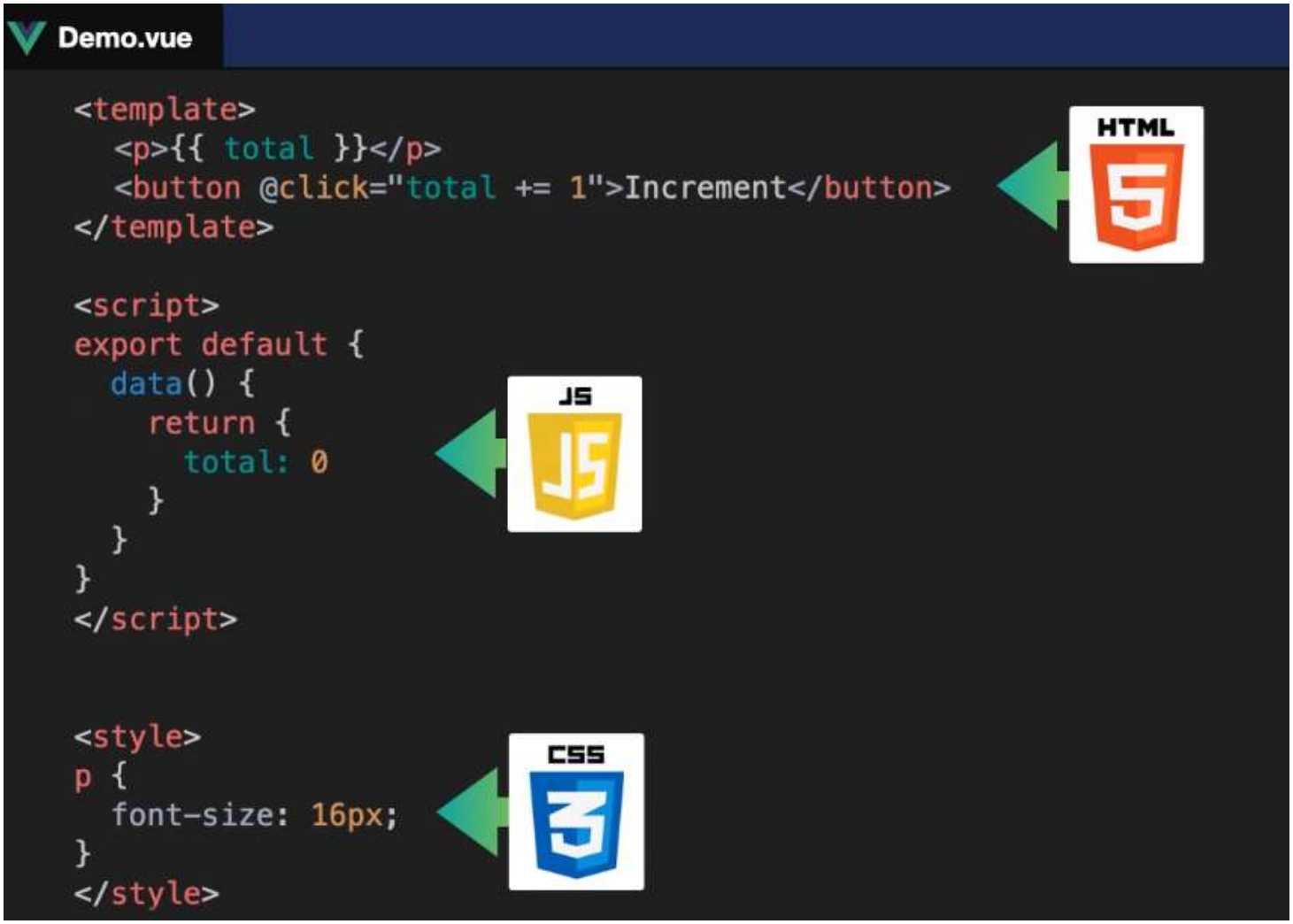
- Ambiente online para test do vue:
  - https://codepen.io/pen/
  - https://www.w3schools.com/html/tryit.asp?filename=tryhtml_default

- Exemplo de binding (ligação)
  - https://codepen.io/team/Vue/pen/dyoeGjW



# Exemplo de Código

- Visão Geral: Template + Vue JS + CSS



# Vue Hello World!

```
<div id="app">
<h1>{{ message }}</h1>
</div>

<script>
const App = {
  data() {
    return {
      message: "hello world."
    };
  }
};
Vue.createApp(App).mount("#app");
</script>
```

# Vue Embutido no HTML

- A tag <script> pode ser utilizada para embutir o VUE em uma página HTML
- A última versão do VUE disponível

```
<script src="https://unpkg.com/vue@next"></script>
```

- VUE 3.2.26

```
<script src="https://unpkg.com/vue@3.2.26"></script>
```

- De forma genérica

```
unpkg.com/:package@:version/:file
<script src="https://unpkg.com/vue@3.2.26/dist/vue.global.js"></script>
```

- Versão otimizada para produção
  - https://unpkg.com/browse/vue@3.2.26/dist/vue.runtime.global.prod.js

```
<script
src=" https://unpkg.com/browse/vue@3.2.26/dist/vue.runtime.global.prod.js ">
</script>
```

# Vue Embutido no HTML

https://codepen.io/flavio-ufu/pen/BawVZbO

```
<html>
    <head>
        <link rel="stylesheet" href="index.css">
        <title>Vue App</title>
        <script src="https://unpkg.com/vue@next">
    </head>
    <body>
        <div id="app">
          {{ message }}
        </div>
        <script>
            const App = {
              data() {
                return {
                  message: "hello world."
                };
              }
            };
            Vue.createApp(App).mount("#app");
        </script>
    </body>
</html>
```

# Hello VUE

- HTML

```
<html lang="en">
<head>
  <title>Vue App</title>
  <script src="https://unpkg.com/vue@next"></script>
</head>
<body>
  <div id="app">
    {{ message }}
  </div>
</body>
</html>
```

- VUE

# Declarações Reativas

- HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Vue App</title>
  <script src="https://unpkg.com/vue@next"></script>
</head>
<body>
  <div id="counter">
    Counter: {{ counter }}
  </div>
</body>
</html>
```

# Declarações Reativas Exemplo #1

- VUE.js

```
const Counter = {
  data() {
    return {
      counter: 0
    }
  }
}
Vue.createApp(Counter).mount('#counter')
```

# Declarações Reativas Exemplo #2

- VUE.js

https://codepen.io/flavio-ufu/pen/BawVZbO

```
const Counter = {
  data() {
    return {
      counter: 0
    }
  },
  mounted() {
    setInterval(() => {
      this.counter++
    }, 1000)
  }
}
Vue.createApp(Counter).mount('#counter')
```

# Uma lista de Produtos

```
<div id="app" class="container">
   <h2>{{titulo}}</h2>
   <ul>
      <li>{{livros[0]}}</li>
      <li>{{livros[1]}}</li>
   </ul>
</div>
```

# VUE Model

```
<!-- this is our View – HTML -->
<div id="example-1">
Hello {{ name }}!
</div>
// this is our Model - JS
var exampleData = {
name: 'Vue.js’
}

// This is the ViewModel
// create a Vue instance, or, a "ViewModel"
// which links the View and the Model
var exampleVM = new Vue({
el: '#example-1',
data: exampleData
})
```

# Vue Embutido no HTML

```
<html>
    <head>
        <link rel="stylesheet" href="index.css">
        <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
    </head>
    <body>
        <div id="app">
          {{ message }}
        </div>

        <script>
          var app = new Vue({
           el: '#app',
           data: {
            message: 'Hello Vue!'
           }
          });
        </script>
    </body>
</html>
```

# VUE embutido no HTML

```
<html lang="en">
  <head>
    <title>App</title>
    <script src="https://unpkg.com/vue@next"></script>
  </head>
  <body>
    <div id="app">
      {{ message }}
    </div>
    <script>
      const app = {
        data() {
          return {
            message: "hello world"
          };
        }
      };

      Vue.createApp(app).mount("#app");
    </script>
  </body>
</html>
```

# VUE Diretivas

- O VUE possui um conjunto de diretivas que permitem manipular o DOM da página
- As diretivas utilizam o prefixo v- e adicionam um comportamento reativo aos elementos da página associados com uma instância de uma aplicação VUE
- Para maiores informações:
  - https://v3.vuejs.org/api/directives.html
- Algumas diretivas
  - v-for
    - Utilizado para exibir uma lista de itens utilizando os dados de um vetor
  - v-bind
    - Permite ligar um atributo a um valor de uma instância da aplicação
  - v-on
    - Permite anexar listeners de eventos que invocam métodos em elementos de uma instância da aplicação

# v-on Exemplo

- VUE

```
const EventHandling = {
  data() {
    return {
      message: 'Hello Vue.js!'
    }
  },
  methods: {
    reverseMessage() {
      this.message = this.message
        .split('')
        .reverse()
        .join('')
    }
  }
}
Vue.createApp(EventHandling).mount('#event-handling')
```

- HTML

```
<div id="event-handling" class="demo">
  <p>{{ message }}</p>
  <button v-on:click="reverseMessage">Reverse
Message</button>
</div>
```

# v-bind Exemplo

- JS

```
const AttributeBindingApp = {
  data() {
    return {
      message: 'You loaded this page on ' + new Date().toLocaleString()
    }
  }
}
Vue.createApp(AttributeBindingApp).mount('#bind-attribute’)
```

- HTML

```
<div id="bind-attribute" class="demo">
  <span v-bind:title="message">
    Hover your mouse over me for a few seconds to see my dynamically-bound
    title!
  </span>
</div>
```

# v-for Exemplo

- VUE

```
const ListRendering = {
  data() {
    return {
      todos: [
        { text: 'Learn JavaScript' },
        { text: 'Learn Vue' },
        { text: 'Build something awesome' }
      ]
    }
  }
}

Vue.createApp(ListRendering).mount('#list-rendering')
```

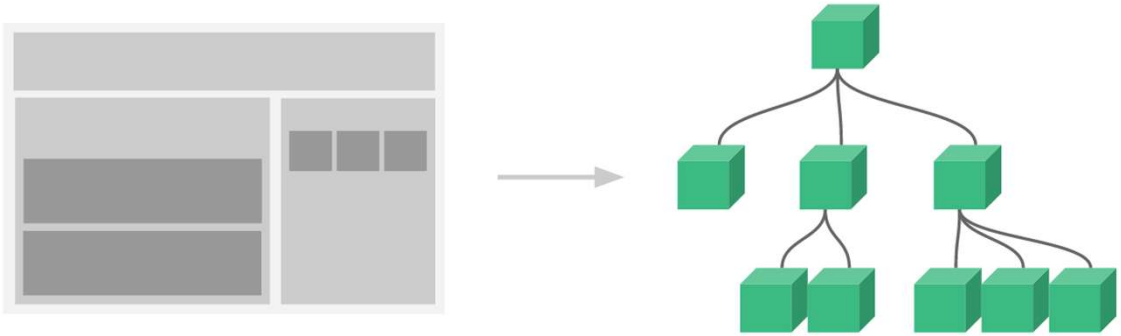
- HTML

```
<div id="list-rendering">
  <ol>
    <li v-for="todo in todos">
      {{ todo.text }}
    </li>
  </ol>
</div>
```

# VUE Component

- O VUE trabalha com o conceito de componentes associados à uma página web
- Um página web pode ser vista como um conjunto de componentes
- Um componente é uma instancia de um elemento com valores pré-definidos
- Exemplo de uso

```
<div id="app">
  <app-nav></app-nav>
  <app-view>
    <app-sidebar></app-sidebar>
    <app-content></app-content>
  </app-view>
</div>
```



# VUE Component
# Visão Geral

```
// Create Vue application
const app = Vue.createApp(...)

// Define a new component called todo-item
app.component('todo-item', {  template: `<li>This is a todo</li>`
})

// Mount Vue application
app.mount(...)

<ol>
  <!-- Create an instance of the todo-item component -->
  <todo-item></todo-item>
</ol>

app.component('todo-item', {
  props: ['todo'],
  template: `<li>{{ todo.text }}</li>`
})
```

# Exemplo de Componente

- VUE

https://codepen.io/team/Vue/pen/VwLxeEz

```
const ComponentsApp = {
  data() {
    return {
      groceryList: [
        { id: 0, text: 'Vegetables' },
        { id: 1, text: 'Cheese' },
        { id: 2, text: 'Whatever else humans are supposed to eat' }
      ]
    }
  }
}
const app = Vue.createApp(ComponentsApp)

app.component('todo-item', {
  props: ['todo'],
  template: `<li>{{ todo.text }}</li>`
})
app.mount('#components-app')
```

# Exemplo de Componente

- HTML

https://codepen.io/team/Vue/pen/VwLxeEz

```
<div id="components-app" class="demo">
  <ol>
    <!--
      Now we provide each todo-item with the todo object it's representing, so that its content can be dynamic.
    -->
    <todo-item
      v-for="item in groceryList"
      v-bind:todo="item"
      v-bind:key="item.id"
    ></todo-item>
  </ol>
</div>
```

# VUE Sample
# Reverse Message

JS

```
const EventHandlingApp = {
  data() {
    return {
      message: 'Hello Vue.js!'
    }
  },
  methods: {
    reverseMessage() {
      this.message = this.message
        .split('')
        .reverse()
        .join('')
    }
  }
}

Vue.createApp(EventHandlingApp).mount('#event-handling')
```

# VUE Sample
# Reverse Message

- HTML

```
<div id="event-handling" class="demo">
  <p>{{ message }}</p>
  <button v-on:click="reverseMessage">Reverse Message</button>
</div>
```

- CSS

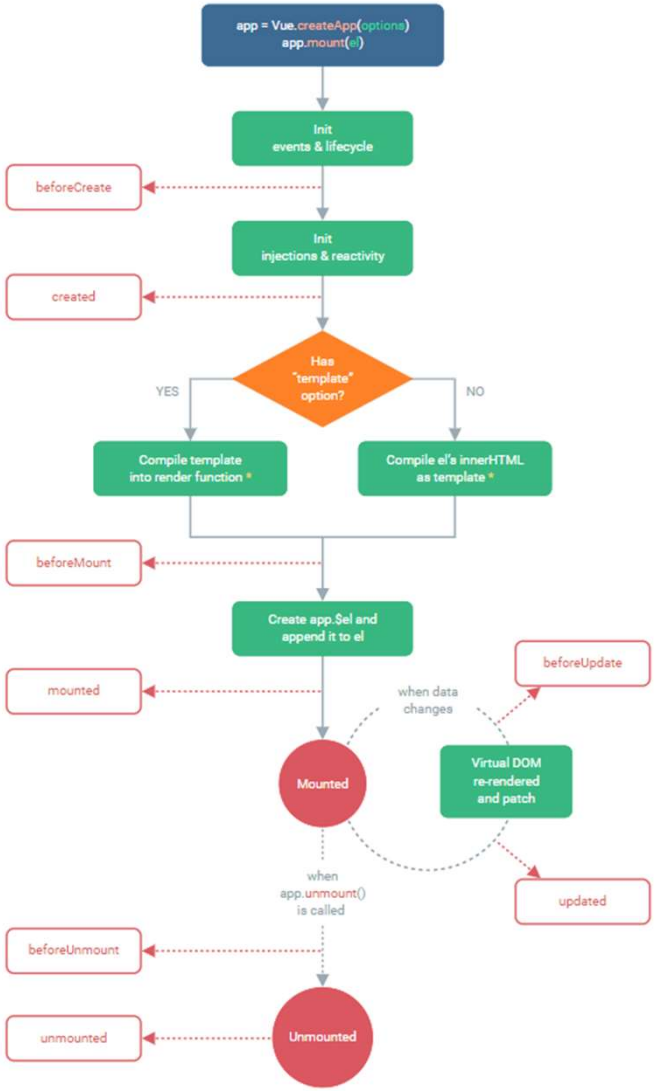
```
.demo {
  font-family: sans-serif;
  border: 1px solid #eee;
  border-radius: 2px;
  padding: 20px 30px;
  margin-top: 1em;
  margin-bottom: 40px;
  user-select: none;
  overflow-x: auto;
}
```

https://codepen.io/team/Vue/pen/dyoeGjW

# VUE App Lifecyle

- A figura ao lado mostra o ciclo de vida de uma aplicação baseada no VUE 3.

- Maiores informações:
  - https://v3.vuejs.org/guide/instance.html#lifecycle-hooks



# VUE Single-File Components

- O VUE utiliza um tipo de arquivo chamado *.vue que é um formato especial de arquivo que permite encapsular o template, a lógica e o estilo em um único componente
- No geral a ideia do SFC é criar componentes menores e reutilizáveis
- O arquivo possui a seguinte estrutura:

  **<template> //view**

  **<script> //logic**

  **<style> //view presentation style**

- A sintaxe deste arquivo é compatível com a linguagem HTML

- Maiores informações:
  - https://vuejs.org/guide/scaling-up/sfc.html#introduction

# VUE CLI

- O VUE CLI é um ambiente completo para desenvolvimento rápido de aplicações baseadas no VUE
- O VUE CLI possui três componentes
  - CLI – pacote que permite acessar o vue do terminal do SO
  - CLI SERVICE – serviço básico para a criação das aplicações
  - CLI PLUGINS – pacotes de suporte para aplicações baseadas no VUE

- Maiores informações:
  - https://cli.vuejs.org/