



# Vue.js 2.0

**Aplicações, diretivas e comunicação com APIs**

**Ely – [elydasilvamiranda@gmail.com](mailto:elydasilvamiranda@gmail.com)**

# Referências

- Esses slides usam como base as seguintes fontes:
  - <https://vuejs.org/v2/guide/index.html>
  - Capítulos de 1 a 13 do livro The Majesty of Vue.js 2;

# Objetivos

- Apresentar o Vue.js 2 ou simplesmente Vue como uma das alternativas para "cliente";
- Mostrar elementos básicos do framework como estrutura de uma vue app e diretivas;
- Realizar chamadas simples a uma Web API;

# Limitações

- Por ser um primeiro contato com Vue, essa aula não contempla:
  - Autenticação;
  - Criação de componentes;
  - Design;
  - Robusto tratamento de erros;
- Tais pontos serão contemplados posteriormente.

# Vue.js

- Framework progressivo e expansível de JS;
- Foco na camada de visão;
- Segundo os autores, o foco do framework é:
  - Reatividade;
  - Componentes;
  - Modularidade

# "Instalando"

- Download:
  - <http://vuejs.org/js/vue.js>
  - Versão minificada: <https://vuejs.org/js/vue.min.js>
- Basta incluí-lo na tag script:

```
<!DOCTYPE html>  
<html>  
<head>  
  <title>My first Vue app</title>  
  <script src="vue.min.js"></script>  
</head>  
<body>  
</body>  
</html>
```

# Criando uma app Vue

- Uma página que use o Vue deve ter uma variável JS do tipo Vue:

```
<script>  
  var app = new Vue({  
    el: '#app'  
  })  
</script>
```

- Essa variável possui um dicionário e uma das entradas é a variável el (elemento);
- Esse elemento deve referencia o id de uma tag HTML:

```
<div id="app"> </div>
```

- Tudo dentro dessa tag é gerenciado pelo Vue. <sup>7</sup>

# Dados e Binding


- Dentro da app Vue podemos declarar vários subdicionários;
- Um deles é um subdicionário com a chave *data*;
- Nessa entrada são declaradas as variáveis da app análogos a um JSON;
- Cada variável pode ser referenciada no HTML entre chaves `{{ nome_variável }}`;
- Podemos alterar uma variável vinculando (binding) uma tag através do v-model.



# Hello Vue

...

```
<body>
  <div id="app">
    <input v-model="message"/>
    {{ message }}
  </div>
  <script>
    var app = new Vue({
      el: '#app',
      data: {
        message: 'Hello Vue!'
      }
    })
  </script>
</body>
```



*O que for digitado na caixa de texto reflete na variável message*

...

# Diretivas

- No Vue, alguns "comandos" são feitos através de diretivas, como a v-model;
- Alguns exemplos:
  - v-show: usada para exibir ou não um elemento, através de uma condição;
  - v-if: pode ser usado ao invés do v-show;
  - v-else: exibe um elemento quando v-if retorna falso;
  - v-for: faz um loop através de uma coleção.

# v-if e v-else

...

```
<div id="app">
```

```
  <input v-model="message"/><br/>
```

```
  <span v-if="message">
```

Este texto aparece se você preencher algo

```
  </span>
```

```
  <span v-else>
```

Este texto aparece se a caixa estiver vazia

```
  </span>
```

```
</div>
```

```
<script>
```

```
  var app = new Vue({
```

```
    el: '#app',
```

```
    data: {
```

```
      message: ''
```

```
    }
```

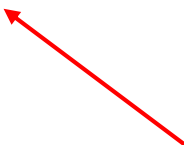
```
  })
```

```
</script>
```

...

# v-for

```
...  
<div id="app2">  
  <ol>  
    <li v-for="todo in todos">  
      {{ todo.text }}  
    </li>  
  </ol>  
</div>  
<script>  
var app2 = new Vue({  
  el: '#app2',  
  data: {  
    todos: [  
      { text: 'Learn JavaScript' },  
      { text: 'Learn Vue' },  
      { text: 'Build something awesome' }  
    ]  
  }  
})  
</script>  
...
```



Para cada um dos todos, exibe-os numa lista

# v-for

```
...
<div id="app2">
  <ol>
    <li v-for="todo in todos">
      Todo: {{ todo.text }} / Percent concluded: {{todo.percent}} %
    </li>
  </ol>
</div>
<script>
  var app2 = new Vue({
    el: '#app2',
    data: {
      todos: [
        {
          text: 'Learn JavaScript',
          percent: 100
        },
        {
          text: 'Learn Vue',
          percent: 20
        }
      ]
    }
  })
</script>
...
```

# Eventos

- Vue pode tratar de forma simples os eventos dos componentes HTML através da diretiva v-on;
- Exemplo:
  - v-on:click - captura o evento onclick;
  - @click – versão resumida do evento acima;
  - @click.prevent – captura o onclick e anula o evento principal (ex: submit);
- Devemos escrever uma função para capturar o evento em um dicionário de métodos:

```
var app3 = new Vue({  
  el : '#app3',  
  data: {  
  },  
  methods : {  
    funcao1 : function() { }  
  }  
})
```

# Eventos

```
...
<div id="app3">
  <p>Votes: {{votes}}</p>
  <button @click = "upvote">Vote</button>
</div>
<script>
  var app3 = new Vue({
    el : '#app3',
    data: {
      votes : 0
    },
    methods : {
      upvote : function() {
        this.votes++
      }
    }
  })
</script>
...
```

# Interagindo com uma API

- A equipe do Vue recomenda a biblioteca Axios para fazer chamadas a uma API:
  - Disponível em:
    - <https://unpkg.com/axios/dist/axios.min.js>
  - Documentação:
    - <https://github.com/mzabriskie/axios>
- Dessa forma, teremos duas tags scripts: uma com o Vue e outra com o Axios;
- Após declarar a tag script, basta acessar a variável com o nome axios.



# Axios

- Principais métodos:
  - `axios.request(config)`
  - `axios.get(url[, config])`
  - `axios.delete(url[, config])`
  - `axios.head(url[, config])`
  - `axios.post(url[, data[, config]])`
  - `axios.put(url[, data[, config]])`
  - `axios.patch(url[, data[, config]])`

# Axios

- Uma chamada a uma API via axios inclui basicamente:
  - O método a ser chamado;
  - O tratamento em caso de sucesso;
  - O tratamento em caso de falhas.
- Ex:

```
axios.metodo('URL')  
  .then(function (response) {  
    // caso de sucesso  
  })  
  .catch(e => {  
    // caso de falhas  
  })
```

# Importante

- Habilite o plugin Allow-Control-Allow-Origin para a url: `http://localhost:porta/`

# Exemplo

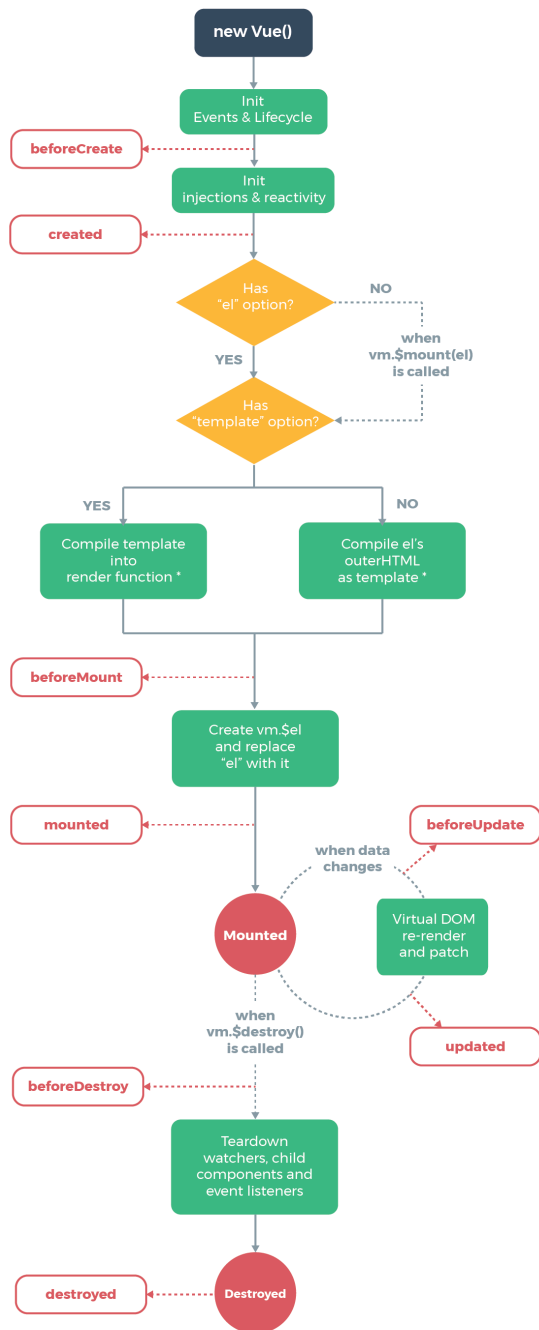
- get:

```
axios.get('http://localhost/posts')  
  .then(function (response) {  
    app.posts = response.data  
  })  
  .catch(e => {  
    app.errors.push(e)  
  })
```

# Estudo de caso

- Iremos utilizar o exemplo de uma atividade anterior: post, user e comments;
- No primeiro exemplo, recuperaremos todos os posts;
- Post possui os campos: id, title, body e user
- Não utilizaremos paginação ainda;
- Para recuperar todos os posts, faremos uso de um método evento chamado mounted;
- Evento representa o momento em que a página já está definida.

# Eventos de uma app vue



<https://vuejs.org/v2/guide/instance.html>

# Listando os posts

- HTML:

...

```
<div id="app4">
  <div v-for = "post in posts">
    <h1>{{post.title}}</h1>
    <p>{{post.body}}</p>
    <p class = "user">{{post.user}}</p>
  </div>

  <span v-if = "errors">
    {{ errors.response.data }}
  </span>
</div>
</div>
```

...

# Listando os posts

- JavaScript:

```
<script>
```

```
var app4 = new Vue({  
  el : '#app4',  
  data: {  
    posts : [],  
    errors : ''  
  },  
  mounted () {  
    axios.get('http://localhost/posts')  
      .then(function (response) {  
        app4.posts = response.data  
      })  
      .catch(e => {  
        app4.errors = e  
      })  
  },  
})
```

```
})
```

```
</script>
```



# Incluindo usuários

- HTML:

...

```
<div id="app5">
  <form>
    <input v-model = "user.name" placeholder="Username">
    <br/>
    <input v-model = "user.email" placeholder="email">
    <button @click.prevent= "inserirUsuario">
      Inserir Usuário
    </button>
  </form>
  <p v-if = "response">
    {{ response.status }} / {{ response.statusText }}
  </p>
  <p v-if = "errors" class = "erros">
    {{ errors.response.data }}
  </p>
</div>
```

...

# Incluindo usuários

- JavaScript:

```
<script>
var app5 = new Vue({
  el : '#app5',
  data: {
    user :{ name: '', email : ''} ,
    response : '',
    errors : ''
  },
  methods : {
    inserirUsuario : function(){
      axios.post('http://localhost/users/', app5.user)
        .then(function (response) {
          app5.response = response
        })
        .catch(e => {
          app5.errors = e
        })
    }
  }
})
</script>
```

# Atividade

- Crie uma implementação com Vue que:
  - Liste um post pelo id;
  - Inclua um post;
- Crie uma listagem de usuários. Em cada item da lista deve ter um link para remover o usuário mediante confirmação



# Vue.js 2.0

**Aplicações, diretivas e comunicação com APIs**

**Ely – [elydasilvamiranda@gmail.com](mailto:elydasilvamiranda@gmail.com)**