



# Desenvolvimento de Software para WEB

Aula 30 - Vue.js - Components

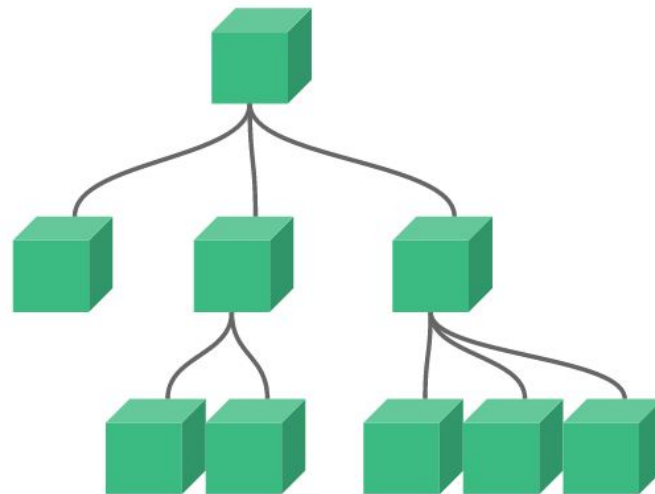
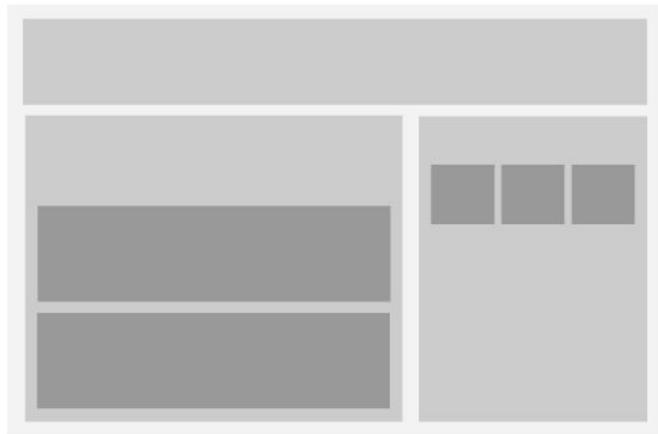
Professor: Anderson Almada

# Components

---

- São utilizados para o reúso de determinados elementos
- Eles podem compor uma aplicação inteira
- Eles podem ser integrados
- Eles recebem parâmetros

# Components - Organização



# Components - Estrutura

---

<template>

  <div class="hello">

    <h1>Bem-vindo a sua app com Vue.js</h1>

  </div>

</template>

<style>

</style>

<script>

</script>

# Components - Hands-On

---

- Criar um TodoList
- Que componentes devemos utilizar?

# Components - Hands-On

---

- Criar um TodoList
- Que componentes devemos utilizar?
  - Input (receber a atividade)
  - Lista, Tabela (mostrar as atividades)

# Components - Hands-On

---

- Realize as importações

- Vue.js

- `<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>`

- Bootstrap (opcional)

- `<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">`

# Components - Hands-On

---

- Crie uma **div** principal para receber todo o código

```
<div class="container">  
  <h2>Todo List</h2>  
</div>
```



# Components - Hands-On

---

- Na div principal, coloque a div que irá renderizar os elementos do Vue.js

```
<div class="container">  
  <h2>Todo List</h2>  
  <div id="app">  
  
    </div>  
  </div>
```

# Components - Hands-On

- Na seção de **script** do código principal, crie a instância do Vue.js

```
<script>
  var app = new Vue({
    el: "#app",
    data: {
      title: "Hello World",
    }
  });
</script>
```

**el:** elemento onde será realizado a renderização  
**data:** dados que podem ser na aplicação

# Components - Hands-On

---

- Agora, vamos criar o primeiro componente, no caso o input.
- O primeiro elemento a ser definido é o template. Defina um id para cada um dos componentes. O id server para referenciar que template deve compor o componente.

```
<template id="input-todo">
```

```
</template>
```

# Components - Hands-On

---

- Como o primeiro componente precisa de um input, então devemos inserir esse elemento dentro do template

```
<template id="input-todo">  
  <input type="text" placeholder="O que precisa ser feito?" >  
</template>
```

# Components - Hands-On

---

- Para essa aplicação, vamos inserir um evento no input, em que toda vez que for pressionado o **enter** do teclado\*, será disparado a ação de chamar o método **add**. Assim:

```
<template id="input-todo">  
  <input type="text" v-on:keyup.enter="add" placeholder="O que precisa  
    ser feito?">  
</template>
```

\*<https://br.vuejs.org/v2/guide/events.html#Key-Codes>

## Components - Hands-On

- Na parte de **script**, será definido o componente. Ele tem a estrutura similar a instância principal.

```
Vue.component('input-todo', {  
  template: "#input-todo",  
});
```

nome do componente



Id do template



# Components - Hands-On

---

- No componente também deve ser definido o método que será utilizado sempre que for pressionado o **enter**. Assim:

```
Vue.component('input-todo', {  
  template: "#input-todo",  
  
  methods: {  
    add($event) {  
      console.log($event.target.value);  
    },  
  },  
});
```

# Components - Hands-On

---

- Agora vamos implementar o segundo componente. Da mesma forma que o primeiro, vamos definir o template:

```
<template id="table-todo">
```

```
</template>
```



# Components - Hands-On

- Agora vamos implementar o segundo componente. Da mesma forma que o primeiro, vamos definir o template:

```
<template id="table-todo">
  <table class="table">
    <thead>
      <tr><th scope="col">Status</th>
      <th scope="col">#</th><th scope="col">Atividade</th>
    </tr>
    </thead>
    <tbody>
    </tbody>
  </table>
</template>
```

## Components - Hands-On

---

- Esse componente deve receber as atividades para renderizar na tabela. É o que chamados de envio de dados entre componentes (pai-filho). Assim, é preciso definir uma propriedade para esse componente.

```
Vue.component( 'table-todo', {  
  template: "#table-todo",  
  
  props: [ 'todos' ],  
});
```

# Components - Hands-On

- Agora vamos voltar ao template e alterar a tag **tbody**.

```
...  
<tbody>  
  <tr v-for="(todo, index) in todos" >  
    ...  
  </tr>  
</tbody>  
...
```

**v-for** é a diretiva que funciona como o for. Utilizado para realizar iteração em array.

**index** é a posição e **todo** é cada um dos elementos do **todos**, definido como propriedade

# Components - Hands-On

- Agora vamos voltar ao template e alterar a tag **tbody**.

```
...  
<tbody>  
  <tr v-for="(todo, index) in todos" >  
    <td><input type="checkbox" v-model="todo.checked"></td>  
    ...  
  </tr>  
</tbody>  
...
```

**v-model** é uma diretiva que é linkada ao dado, o qual serve para realizar mudanças

# Components - Hands-On

- Agora vamos voltar ao template e alterar a tag **tbody**.

```
...  
<tbody>  
  <tr v-for="(todo, index) in todos" >  
    <td><input type="checkbox" v-model="todo.checked"></td>  
    <th scope="row">{{index+1}}</th>  
    <td v-if="todo.checked" >  
      ...  
    </td>  
  </tr>  
</tbody>  
...
```

**v-if** é a diretiva que funciona como if. Se for verdade, realiza uma ação, caso contrário não realiza

# Components - Hands-On

- Agora vamos voltar ao template e alterar a tag **tbody**.

...

```
<tbody>
  <tr v-for="(todo, index) in todos" >
    <td><input type="checkbox" v-model="todo.checked"></td>
    <th scope="row">{{index+1}}</th>
    <td v-if="todo.checked" >
      <span style="text-decoration:line-through; color:
        red">{{todo.value}}</span>
    </td>
  </tr>
</tbody>
```

...

# Components - Hands-On

- Agora vamos voltar ao template e alterar a tag **tbody**.

```
...  
<tbody>  
  ...  
  <td v-if="todo.checked">  
    <span style="text-decoration:line-through; color:  
      red">{{todo.value}} </span>  
  </td>  
  <td v-else>  
    <span>{{todo.value}} </span>  
  </td>  
</tr>  
</tbody>
```

**v-else** é a diretiva que funciona caso a condição do v-if seja falsa

## Components - Hands-On

---

- Agora vamos criar uma variável global chamada de **tasks**, que é um array, para guardar as atividades inseridas pelo input. Além disso, na instância principal, vamos criar um campo no **data** chamado de **tasks**, também. Esse segundo recebe o primeiro. Feito isso, está linkado o array com um dado que é enxergado pelos componentes e pode ser utilizado como parâmetro para o componente de tabela. Assim:



# Components - Hands-On

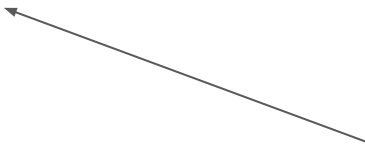
---

```
let tasks = [];  
  
var app = new Vue({  
  el: "#app",  
  data: {  
    tasks: tasks  
  }  
});
```

# Components - Hands-On

- Agora vamos chamar o segundo componente no código principal.

```
<div class="container">
  <h2>Todo List</h2>
  <div id="app">
    <input-todo></input-todo><br><br>
    <table-todo :todos="tasks"></table-todo>
  </div>
</div>
```



**:todos** é a propriedade definida no componente e **tasks** é recuperado da instância principal

## Components - Hands-On

- Agora só falta alterar o componente de input para ir armazenando cada atividade. Altera o método **add** para utilizar o array **tasks**. Assim:

```
Vue.component('input-todo', {  
  template: "#input-todo",  
  methods: {  
    add($event) {  
      let task = {};  
      task.value = $event.target.value;  
      task.checked = false;  
      tasks.push(task);  
    },  
  },  
});
```

# Components - Hands-On

- Enjoy :)

## Todo List

Fazer as aulas da próxima s

Status	#	Atividade
<input checked="" type="checkbox"/>	1	<del>Aula de WEB</del>
<input type="checkbox"/>	2	Terminar o Doutorado
<input type="checkbox"/>	3	Fazer as aulas da próxima semana

# Links importantes

---

- <https://br.vuejs.org/>
- <https://pastebin.com/caE6hUKL>
- <https://www.youtube.com/watch?v=07-TvnH7XNo&list=PLcoYAcR89n-qg1vGRbaUiV6Q9puy0qigW>



# Dúvidas??

E-mail: [almada@crateus.ufc.br](mailto:almada@crateus.ufc.br)