Vue.js (Components)

Interface Pessoa-Máquina - 25/26 - LEI / UM

Hugo Pacheco
hpacheco@di.uminho.pt

Condicionais

- Nos templates HTML do Vue, podemos renderizar elementos condicionalmente

```
<h1 v-if="awesome">Vue is awesome!</h1>
```

- Suporte para condições encadeadas

```
<div v-if="type == 'A'"> A </div>
<div v-else-if="type == ‘B'"> B </div>
<div v-else-if="type == 'C'"> C </div>
<div v-else> Not A/B/C </div>
```

❗ Têm que ser consecutivas

❓ Qual a diferença entre `v-if` e `v-show`?

```
<h1 v-show="awesome">Vue is awesome!</h1>
```

💡 Apenas faz *toggle* da propriedade `display` no CSS

v-if: Adiciona ou remove o elemento do DOM (mudança estrutural).

v-show: Apenas altera a propriedade display no CSS (o elemento continua lá, mas fica invisível).

v-for: Usado para iterar sobre listas ou objetos e construir sequências de elementos repetitivos na interface.

Iteração

- Podemos iterar sobre uma lista e construir uma sequência de elementos

```
const items =
[{ message:'Foo'},
{ message:'Bar'}]

<li v-for="item in items">
  {{ item.message }}
</li>

<li v-for="(item,index) in
items”>
  {{ index }}:{{ item }}
</li>
```

- Também podemos iterar sobre um objeto

```
const myObject = {
  title: 'Test',
  author: 'Vue'
})

<li v-for="value in myObject”>
  {{ value }}
</li>

<li v-for="(value,key) in
myObject">
 {{ key }}:{{ value }}
</li>
```

Exemplo (InsideAirBnb Listings)

- Visualizar tabela de dados (simulados) do Inside AirBnb
  - Two-way data binding
  - Vários input types
- Editar células
- Inserir / remover linhas

- Mudámos de setup
- Projeto com `vite`
- Estrutura
  - `main.js`
  - `App.vue`

InsideAirBnb Listings

Action	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_
-	41339	Porto city fl	180050	Paula	PORTO	Lordelo do Ouro	41,1501	-8,66035	Entire home	69	2	33
-	55111	Fontielas H	259711	Isabel E Joá	PAREDES	Cete	41,17418	-8,35533	Entire home		5	21
-	73828	Fontielas H	259711	Isabel E Joá	PAREDES	Cete	41,17622	-8,35351	Entire home		5	20
+												

JSON server

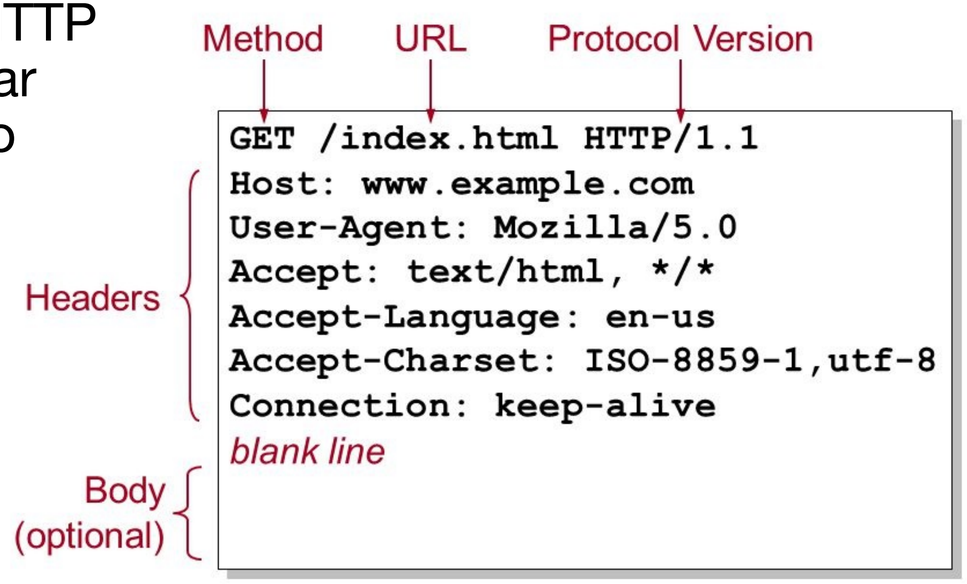
- Uma ferramenta que cria uma API de *mockup* a partir de um ficheiro JSON
  - Sem dependências, para rápido deployment
  - Minimalista, para prototipagem de front-ends
- Suporta operações CRUD (Create, Read, Update, Delete)
- Utilização:
  1. Criar um ficheiro `db.json` com os dados
  2. `json-server db.json`
  3. REST API à escuta em `localhost:3000`

JSON server (`db.json`)

- 💡 Para a 2ª fase do trabalho prático
  - Podem criar um único ficheiro `db.json` a partir dos dados disponibilizados pelo Inside AirBnb
    - Podem utilizar este script Python que cria um base de dados JSON a partir de uma hierarquia de pastas com ficheiros CSV e JSON
    - Podem acrescentar ao ficheiro JSON resultante toda a informação adicional que seja útil para o vosso protótipo
    - ! JSON é um formato bastante pouco compacto. Filtrem primeiro a informação relevante para evitar uma base de dados de muitos MB
    - ! O propósito dos dados é **sempre e apenas para *mockup***. Não se preocupem com ter listagens demasiado completas

JSON server (REST)

- REST (REpresentational State Transfer)
  - Acesso a serviços web de forma simples e flexível
  - Stateless: Cada operação é independente
- Protocol HTTP para efetuar pedidos ao servidor

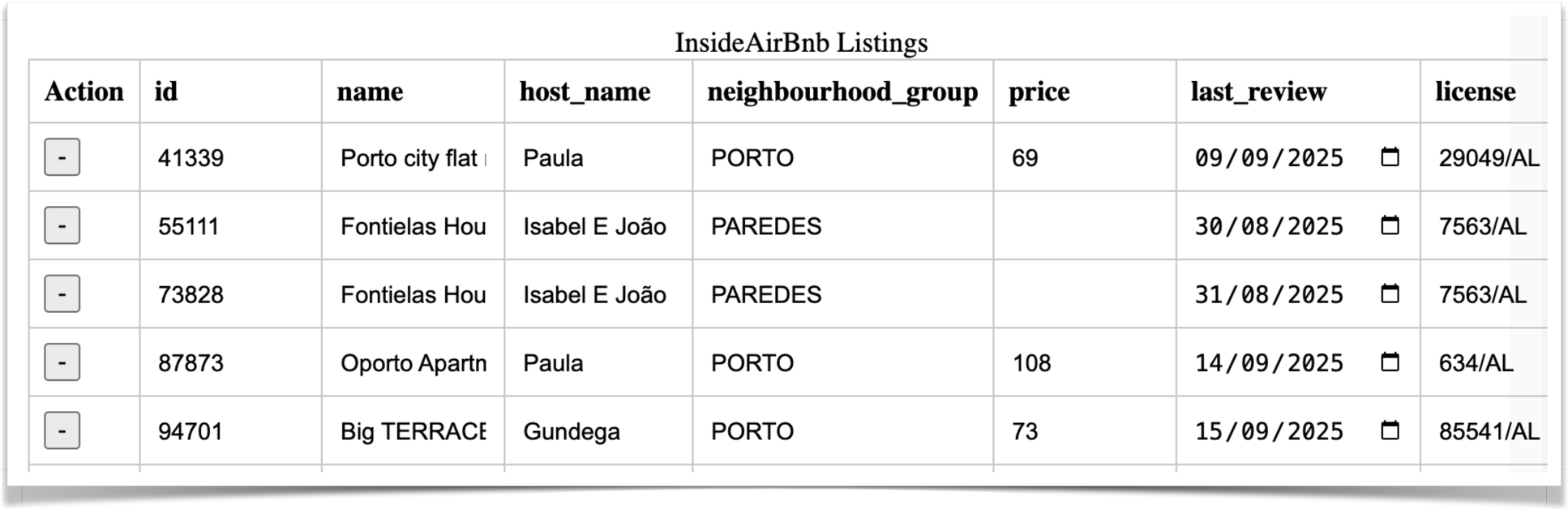


- Operações CRUD através de métodos HTTP

```
POST localhost:3000/listings
{…}

GET localhost:3000/listings/

GET localhost:3000/listings?
id=73828

PUT localhost:3000/listings
{…}

PATCH localhost:3000/listings
{…}

DELETE localhost:3000/
listings/aa6e
```

Exemplo (InsideAirBnb Listings + DB)

- Ações sincronizadas com DB `json-server`
  - Inserir / remover linha
  - Atualizar coluna
- Assíncrono: estado do front-end independente de estado da DB



InsideAirBnb Listings

Action	id	name	host_name	neighbourhood_group	price	last_review	license
-	41339	Porto city flat	Paula	PORTO	69	09/09/2025	29049/AL
-	55111	Fontielas Hou	Isabel E João	PAREDES		30/08/2025	7563/AL
-	73828	Fontielas Hou	Isabel E João	PAREDES		31/08/2025	7563/AL
-	87873	Oporto Apartn	Paula	PORTO	108	14/09/2025	634/AL
-	94701	Big TERRACE	Gundega	PORTO	73	15/09/2025	85541/AL

Fetch API (JS)

- Interface para fazer pedidos HTTP
- Sintaxe: `fetch(url, options)`
  - `url`: the URL of the resource being requested
  - `options`: an optional options object to configure the request

```
{ method: ‘POST',
   headers: {'Content-Type':
‘application/json’},
   body:
JSON.stringify({ key1:
‘value1', key2: 'value2' }),
   …
}
```

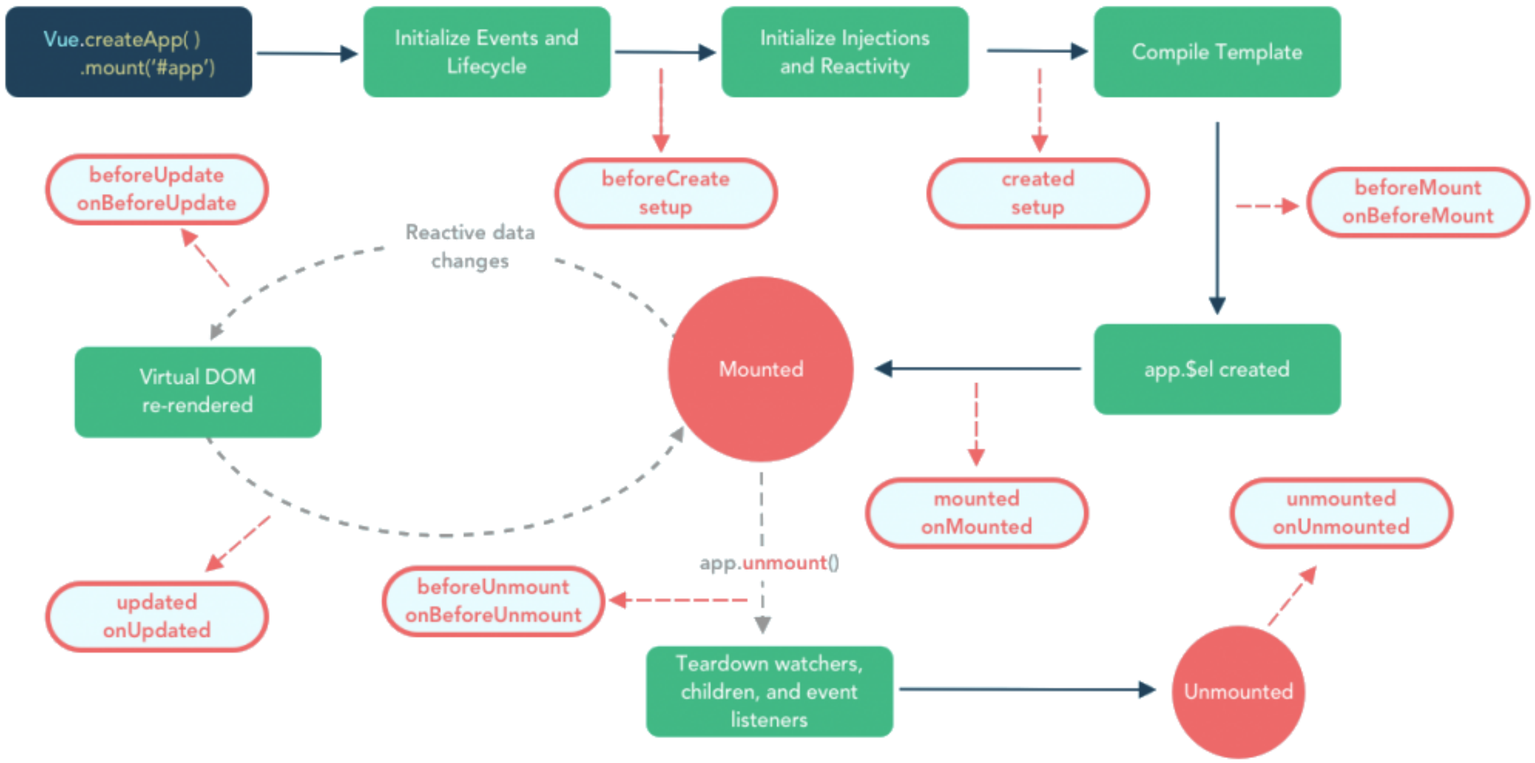
- Métodos:
  - GET: Retrieve data from a server (default method)
  - POST: Send data to a server to create a new resource
  - PUT: Send data to a server to update an existing resource
  - DELETE: Remove an existing resource from the server
  - PATCH: Send partial data to update parts of an existing resource

Promises (JS)

- A função `fetch` retorna uma *promise* (um objeto que encapsula uma computação a ser executada assincronamente)
- Em JS, podemos encadear promises
  - `then()` deals with fulfilled promises
  - `catch()` deals with rejected promises
  - `finally()` executes after the promise is resolved (regardless of its outcome)
- `await` espera pelo resultado de uma promise
- Podemos declarar uma `async function`, que retorna uma promise
  - 💡 Intuição: o browser não fica à espera do resultado

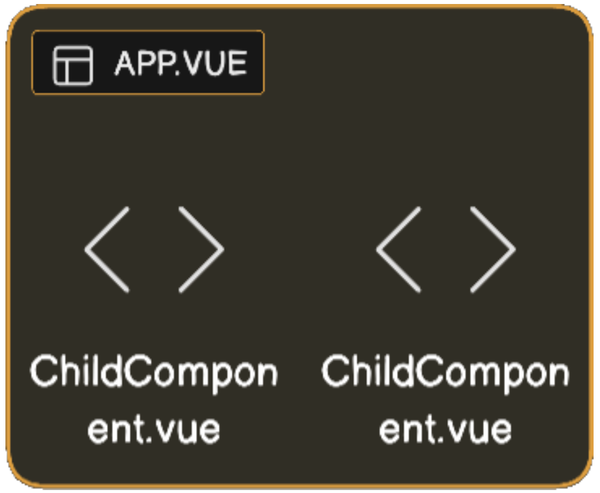
Lifecycle hooks

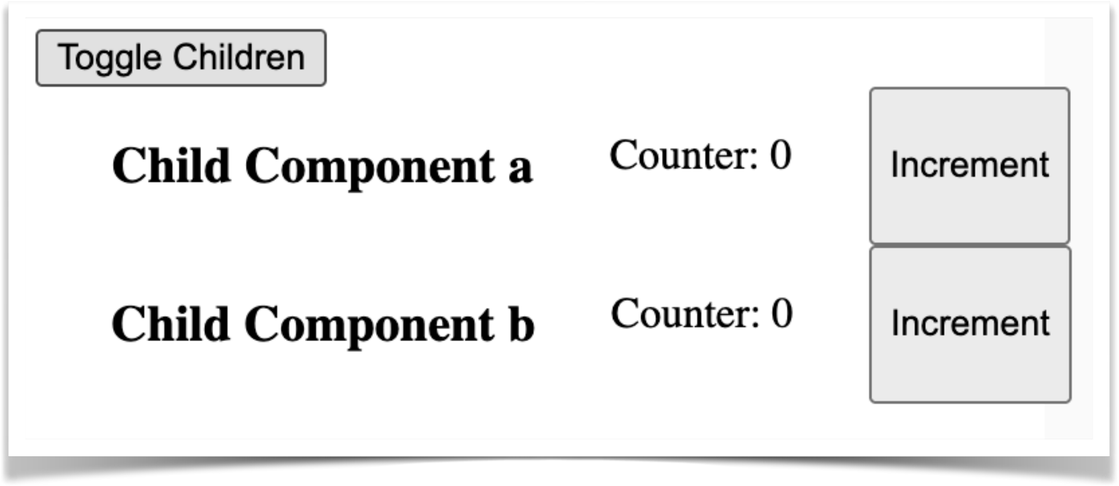
- Uma aplicação / componente Vue tem um ciclo de vida
- Podemos executar métodos Vue em diversas fases



Components

- Em Vue, cada componente é declarado num ficheiro independente
- Devem-se usar multi-word names
  - Em Vue JS: `ComponentName.vue`, `import ComponentName`
  - No template HTML: tag HTML `component-name` ou `ComponentName`
- Exemplo





Components (lifecycle)

- Bastante similar a Web Components
- `<style scoped>` = Shadow CSS
- Dynamic components:
  - Por defeito, components são destruídos quando `unmounted`
    - 💡 Ao contrário de Web Components, em que ficam no DOM
  - Tag HTML `keep-alive` preserva o component e o seu estado

❓ `onUpdated()` ≠ `attributeChangedCallback(name,oldV, newV)`?

- 💡 Virtual DOM ⇒ não queremos saber que valores no modelo mudaram!

Components (`props`)

- Componentes podem receber parâmetros

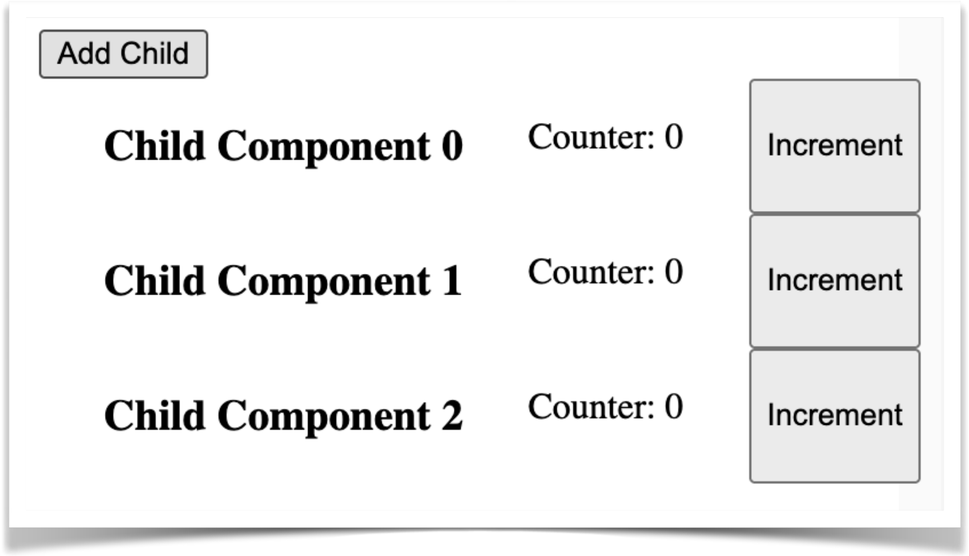
```
<template>
  <h1> Child {{ name }}
</h1>
</template>

<script>
const props =
defineProps({
  myName: String,
});
</script>
```

Pode ter anotações adicionais: `type`, `required`, `default`, ...

- Passados sob a forma de atributos HTML pelos pais

```
<child-component my-
name="{{ childName }}"/>
```

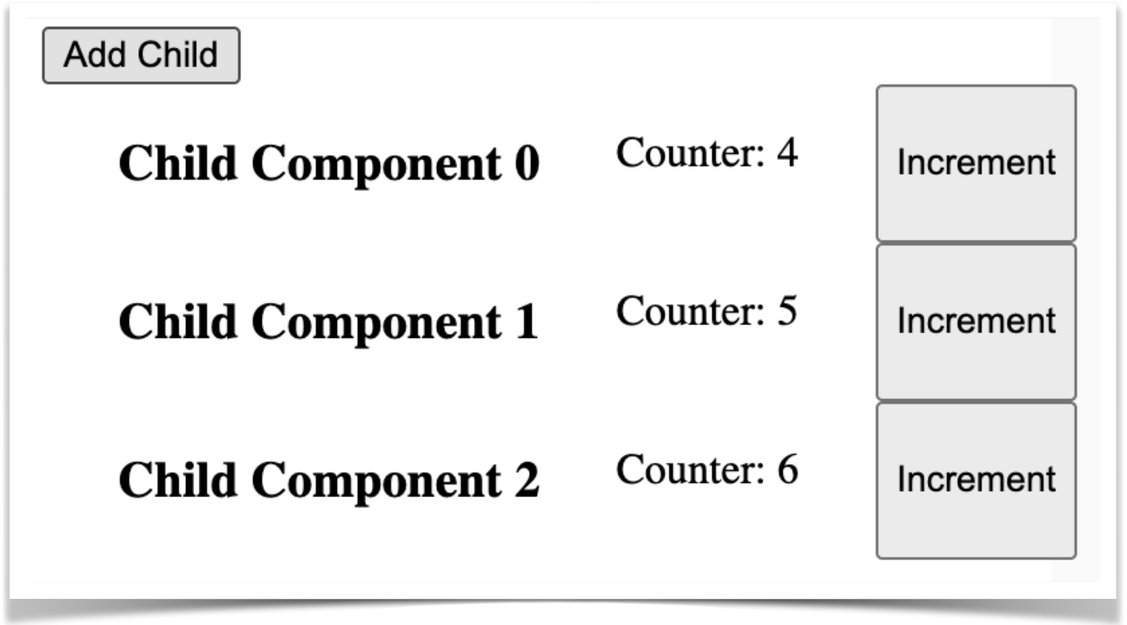
! Convenção de nomes diferente



Components (`props`)

- Valor de uma `prop` é imutável
  - Ao contrário de atributos HTML em Web Components
- Vue segue uma política de **unidirectional data flow**
  - Design pattern para parâmetros mutáveis é criar uma `ref` dentro do componente

- Para hierarquias de components, `props` **não** são passadas implicitamente de pais para filhos
- Comunicação direta de pai para filho



Components (`props`)

- ? Prop passada a child component pode depender de reactive data?
  - Sim, pode.
  - ? O que acontece se valor no parent component mudar?
    - Prop no child component é atualizada automaticamente.
    - ? O que significa então **unidirectional data flow**?
      - 💡 Child components **não podem** mutar props.
      - 💡 Dados e (modificações aos dados) **podem** fluir de parent para child.

Components (`emits`)

- Components podem emitir *custom events*

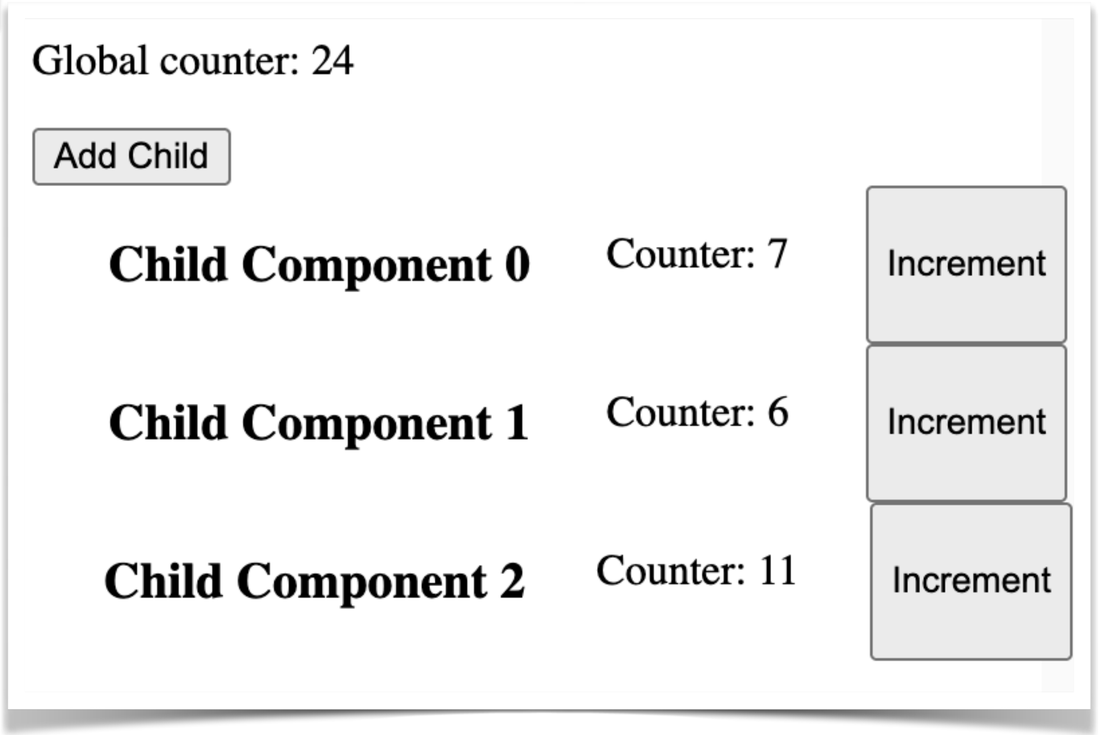
```
const emit =
defineEmits(['inc']);

emit('inc');
```

Pode ter anotações adicionais: parâmetros e tipo, etc

  - 💡 Forma de comunicar de filhos para pais
  - ❗ Ao contrário de JS, não existe *bubbling*

- ❗ Apenas o pai direto pode escutar o evento

```
<my-child
@inc="increment"/>
```



Exemplo (User + Session)

- Vue components são praticamente Web Components!
- Podemos migrar facilmente o exemplo anterior para Vue
- 💡 Pais podem invocar métodos de filhos
  (desde que expostos pelos filhos com `defineExpose`)

Components (slots)

- Uma vantagem de components Vue é maior integração
  - Pais podem passar código Vue a components filhos

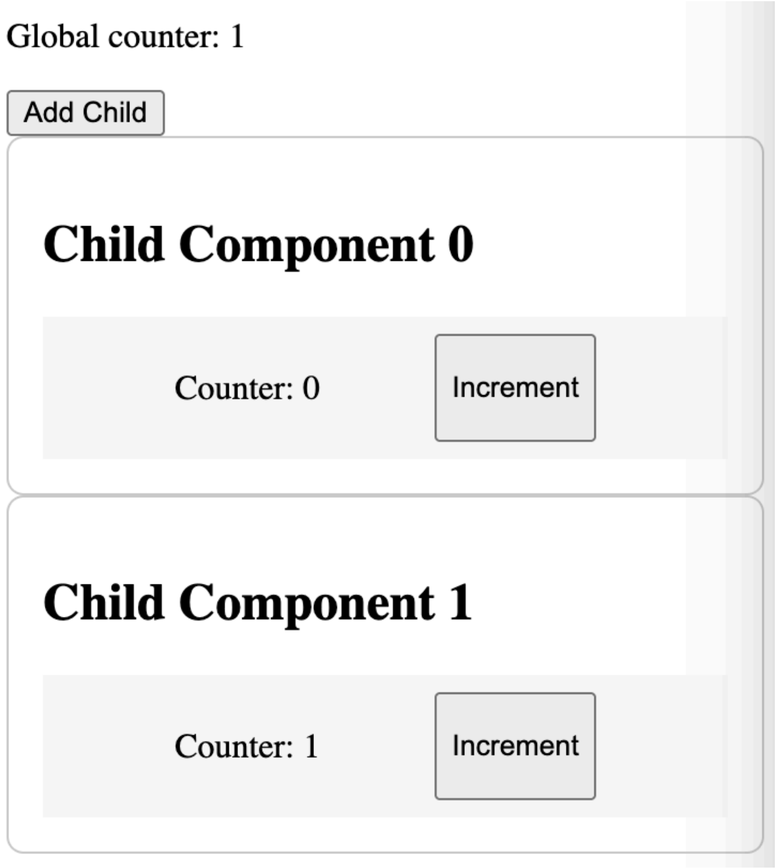
```
<card-component>
  <template #header>Child Component
{{ child }}</template>
  <child-component/>
</card-component>
```

  - Filhos definem uma ou mais slots

```
<template>
  <div>
    <h2><slot name="header"></slot></h2>
    <slot></slot>
  </div>
</template>
```

Template no pai para named slot

DOM do filho para Unnamed Slot



Components: Vue vs Angular

- Remember, remember…
  - Existe um preço a pagar por maior integração
  - Entrevista de Tim Berners-Lee ao jornal Público
    - ? **Não acha que o desenvolvimento web se tornou demasiado complexo para uma plataforma que se queria democrática?**

      *“Usei algumas ferramentas mais modernas, mas desconfio da dependência excessiva que geram. […] Se achamos a web de hoje complicada, porquê ensinar essas ferramentas aos estudantes? HTML e CSS são simples e **permitem copiar e adaptar código a partir de outra página facilmente**. **Essa é a beleza da web**.”*

https://www.publico.pt/2025/11/09/enter/entrevista/tim-bernerslee-web-maravilhosamente-libertadora-2153641