


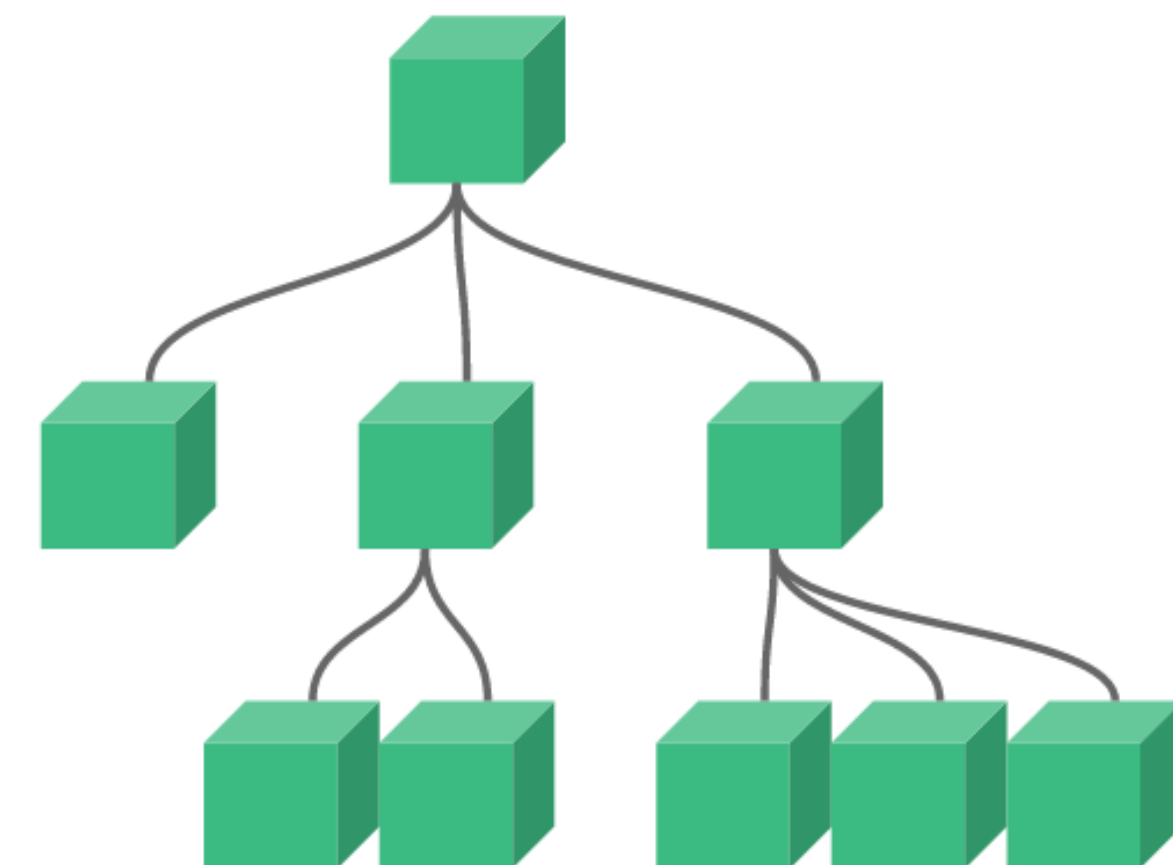
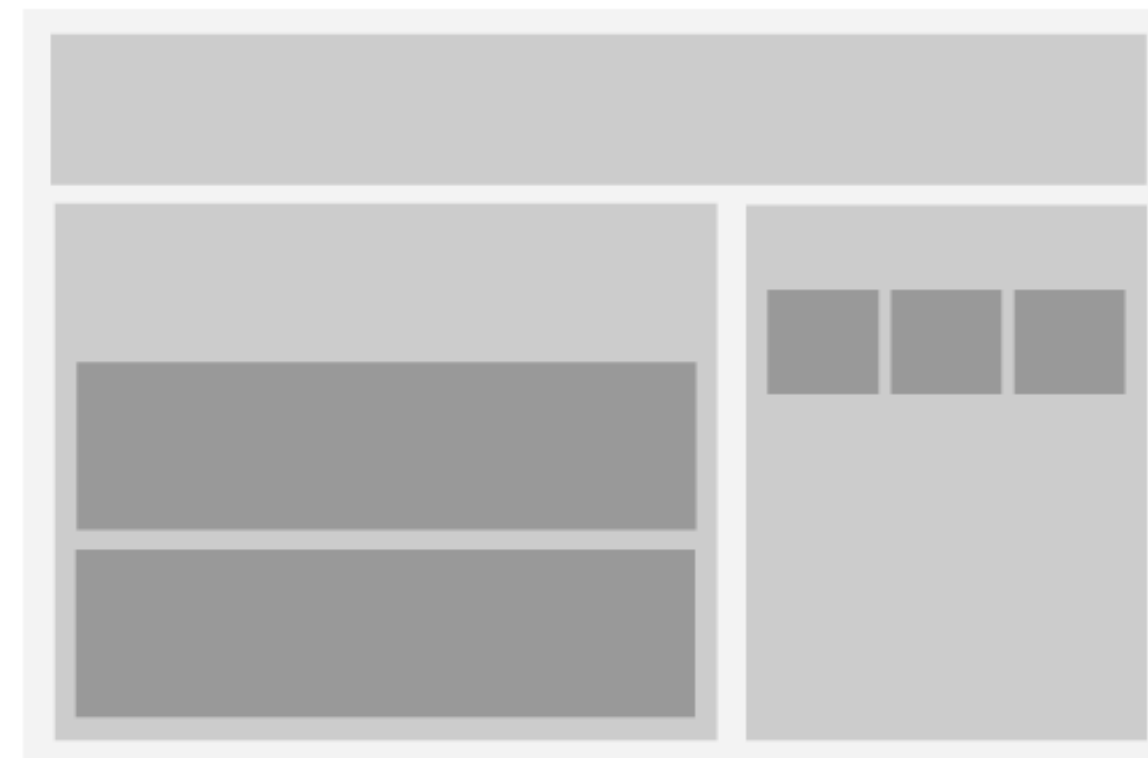
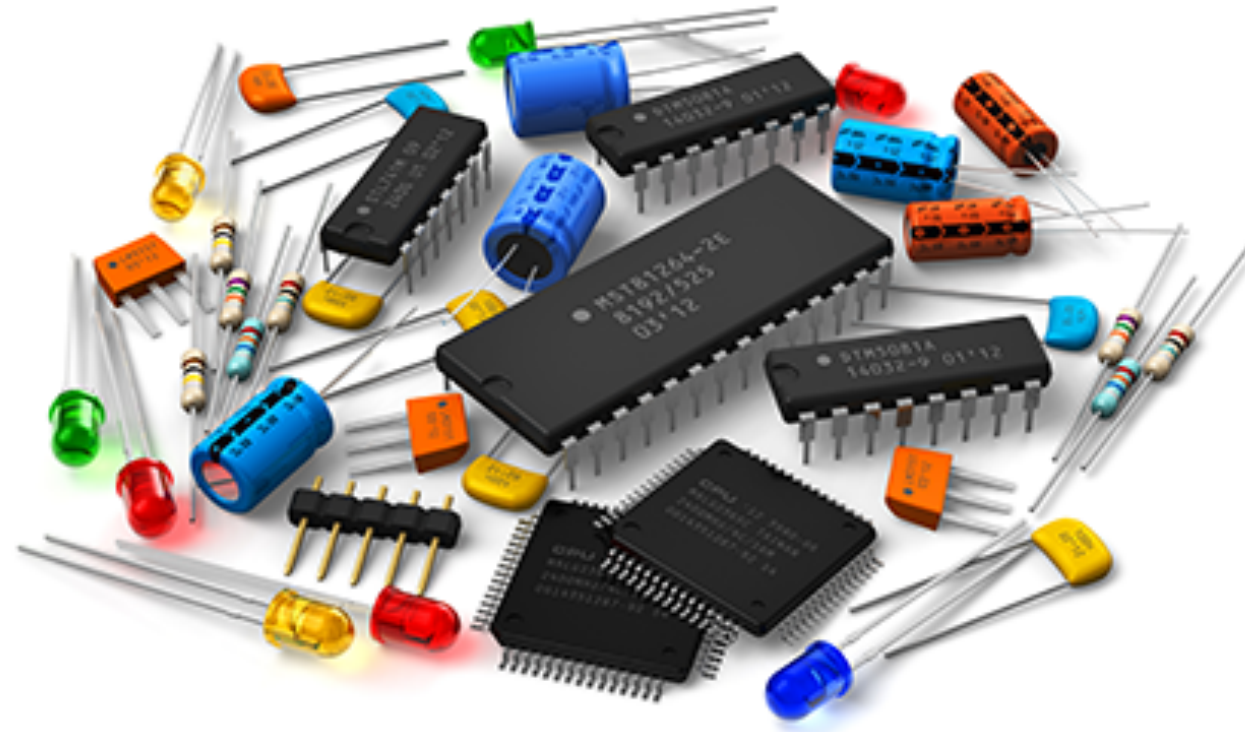


Component



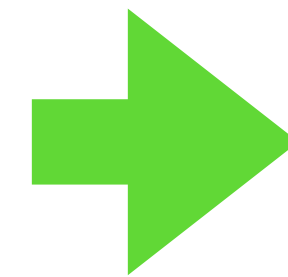
コンポーネント

コンポーネント (部品・構成要素)

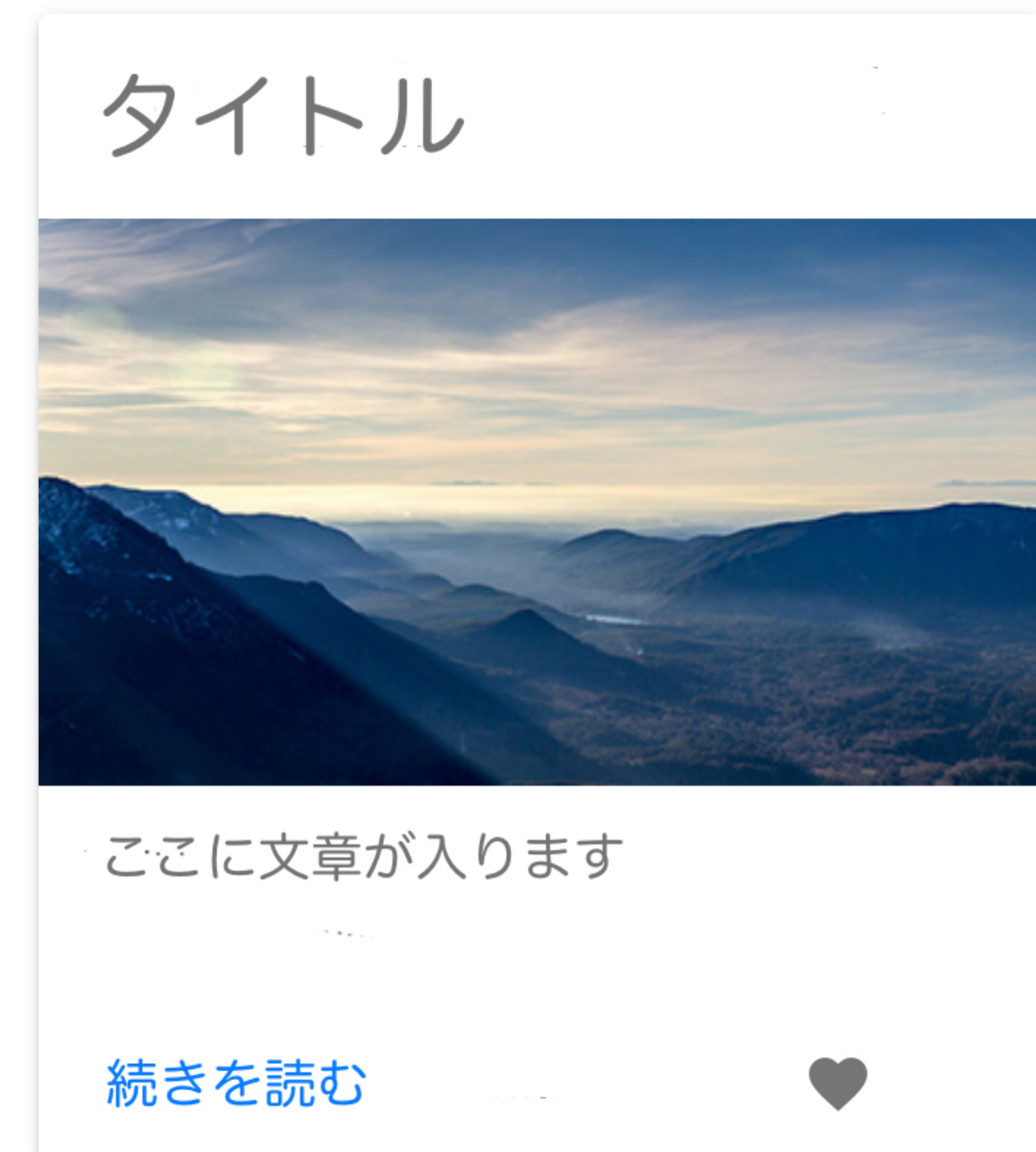


コンポーネント イメージ

```
<div class="card">
  <h3 class="card__title">タイトル</h3>
  <div class="card__body">
    <div class="card__image">
      </div>
    <div class="card__content">
      <p class="card__summary">要約</p>
      <div class="card__controls">
        <a class="button" href="/articleXXX">続
        きを読む</a>
        <a class="button__heart">お気に入り</a>
      </div>
    </div>
  </div>
</div>
```



```
<ArticleCard
  title="タイトル"
  summary="要約"
  url="/articleXXX"
  image-url="/images/xxx.jpg"/>
```



コンポーネント メリット

- ・ 役割分担(責務の分離)
- ・ 使い回しができる
- ・ メンテナンスしやすい

コンポーネント お約束

```
<my-component title=""></my-component>
```

HTMLのタグのように作成

HTMLタグと重ならないよう、


名前は2語以上、ケバブケース(*vueファイル
使用時はパスカルケースも可)

HTMLの属性のように値を設定できる(props)

コンポーネント 簡易表

スコープ	グローバル	ローカル	
書き方 読み込み方	<pre>Vue.component('c-name', { template: `` }) new Vue({})</pre>	<pre>let comA = {} new Vue({ components: { 'com-a': ComA }) })</pre>	<pre>import comA from './ComA.vue' export default { components: { ComA } }}</pre>
拡張子	.html, .js		.vue
ファイル内	<pre><script></script> (script内に template)</pre>		<pre><template></template> <script></script> <style></style></pre>
環境	CDN		Vue-CLI (webpack/babel)
特徴	使ってなくても呼び出される (ほぼ使わない)	責務の分離(役割分担)	責務の分離(役割分担)

グローバルコンポーネント



インスタンス化の前に書く
template内はバッククォート(`)
単一ルートが必須(divタグなど)

```
Vue.component('my-component', {  
  template: `<div>あああ</div>`,  
})
```

```
let app = new Vue({})
```


ローカルコンポーネント

変数で作成

インスタンス内にcomponentsで追加する

```
let myComponent = {}
```

```
let app = new Vue({  
  components: {  
    'my-component': myComponent  
    myComponent // この書き方でもOK  
  })  
})
```



props

props プロパティ



HTMLタグの属性のように自由に設定できる

```
<a href="https://google.com" target="">
```

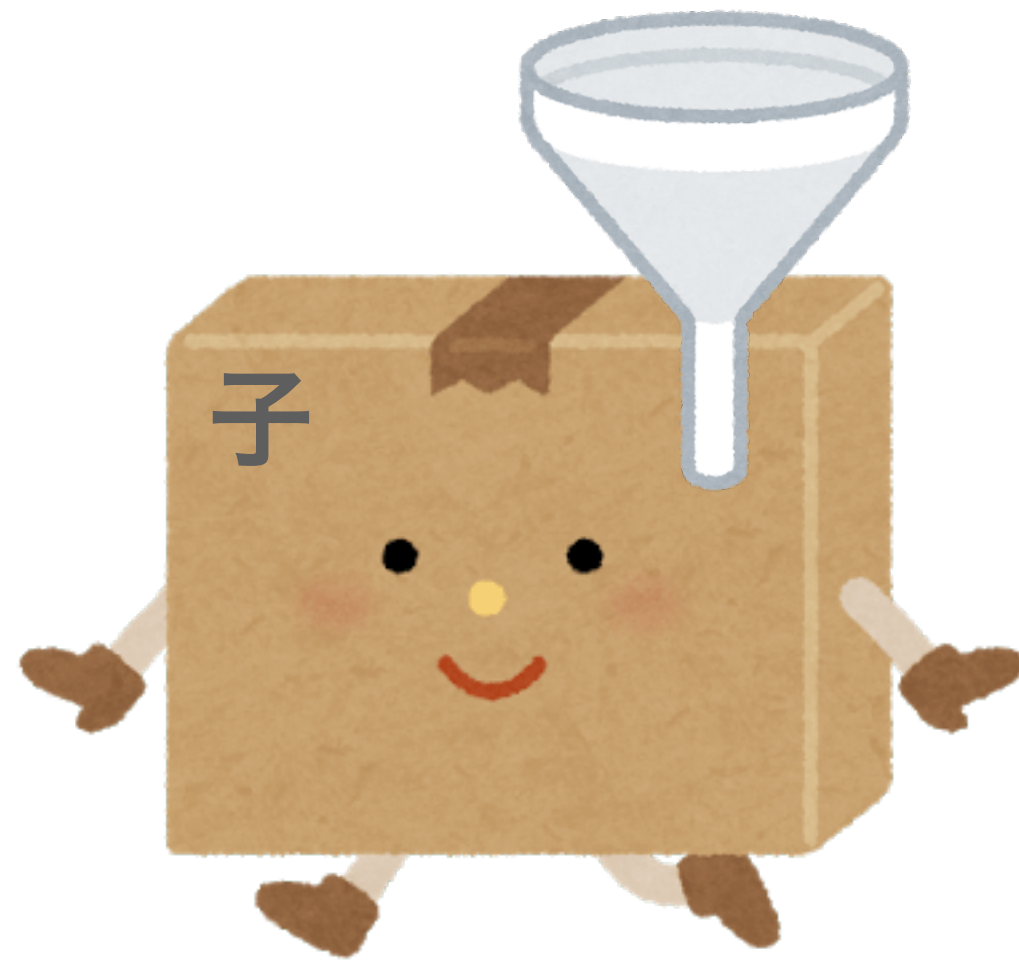
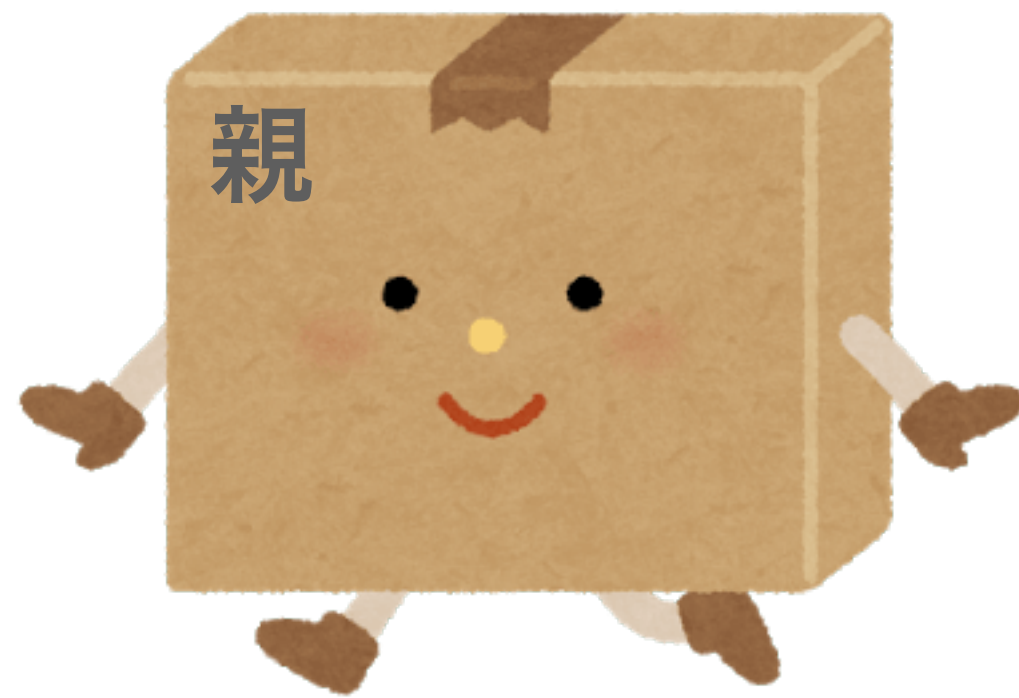
```
<a href="https://yahoo.co.jp">
```

(例) Vuetify のv-btnの場合

```
<v-btn text small color="primary">
```

```
<v-btn depressed large color="error">
```

props (プロパティ)

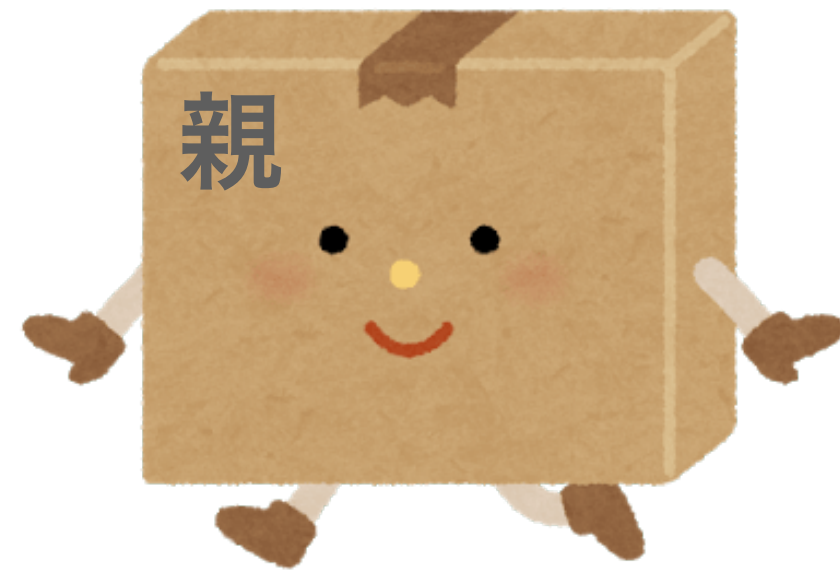


じょうろ(ロート)
コンポーネントが受け取れる情報を指定
親側<v-btn title="テスト">

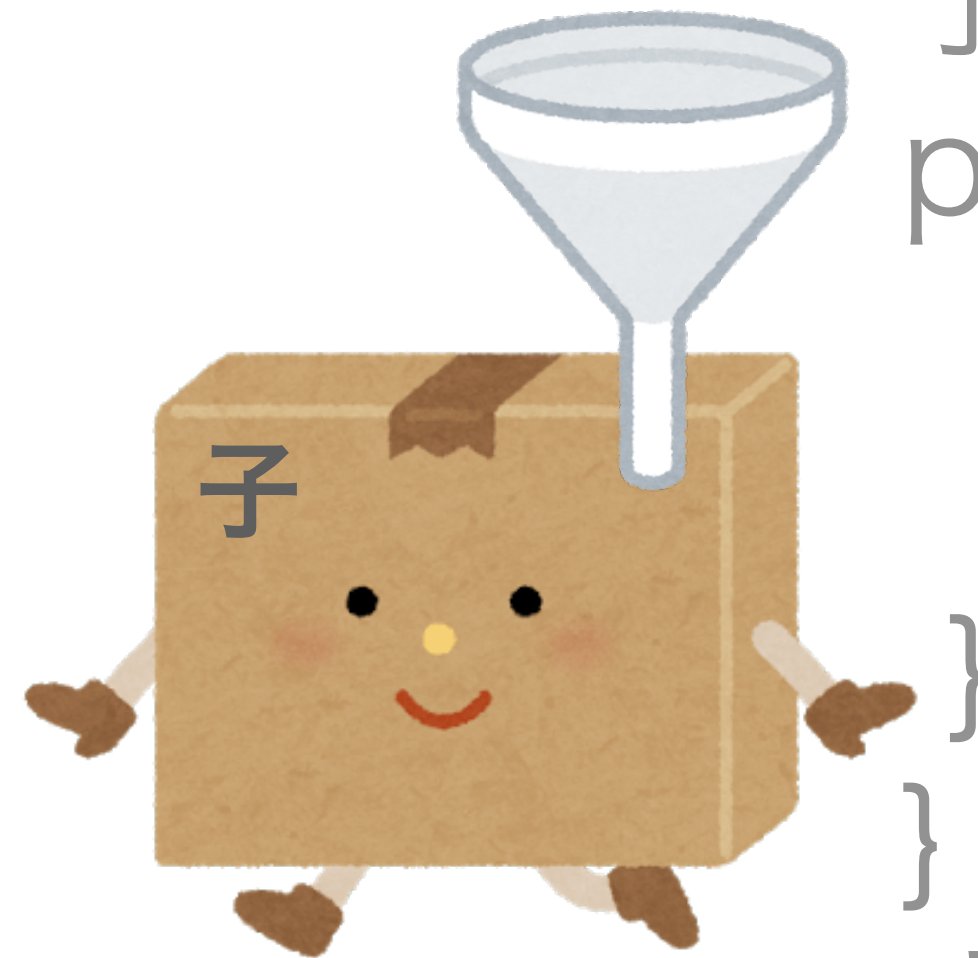
子側への通り道

```
props: {  
  title: {  
    type: String,  
  }  
}
```


props (プロパティ)



propsは親のデータ 親の状況で変更される
子側で変更するのはNG
一旦data(computed)に渡してtemplateで表示

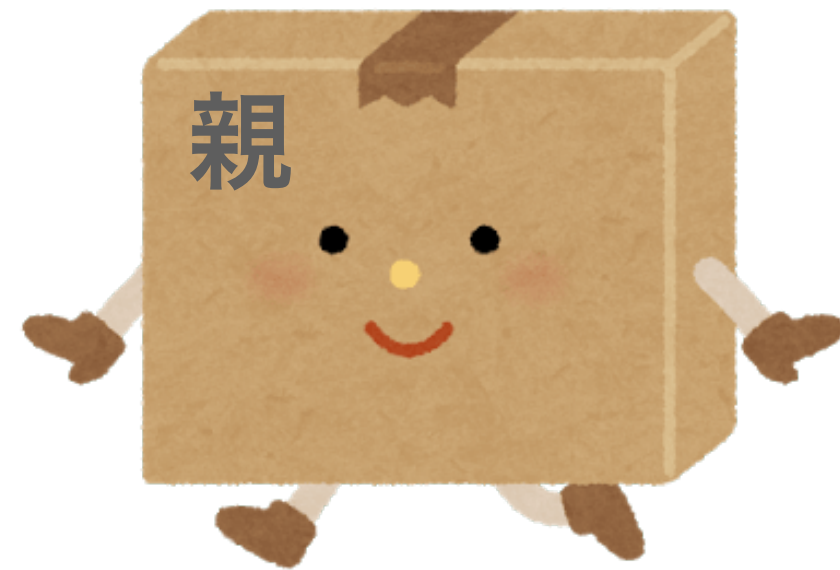


子側への通り道

```
props: {  
  title: {  
    type: String,  
  }  
}
```

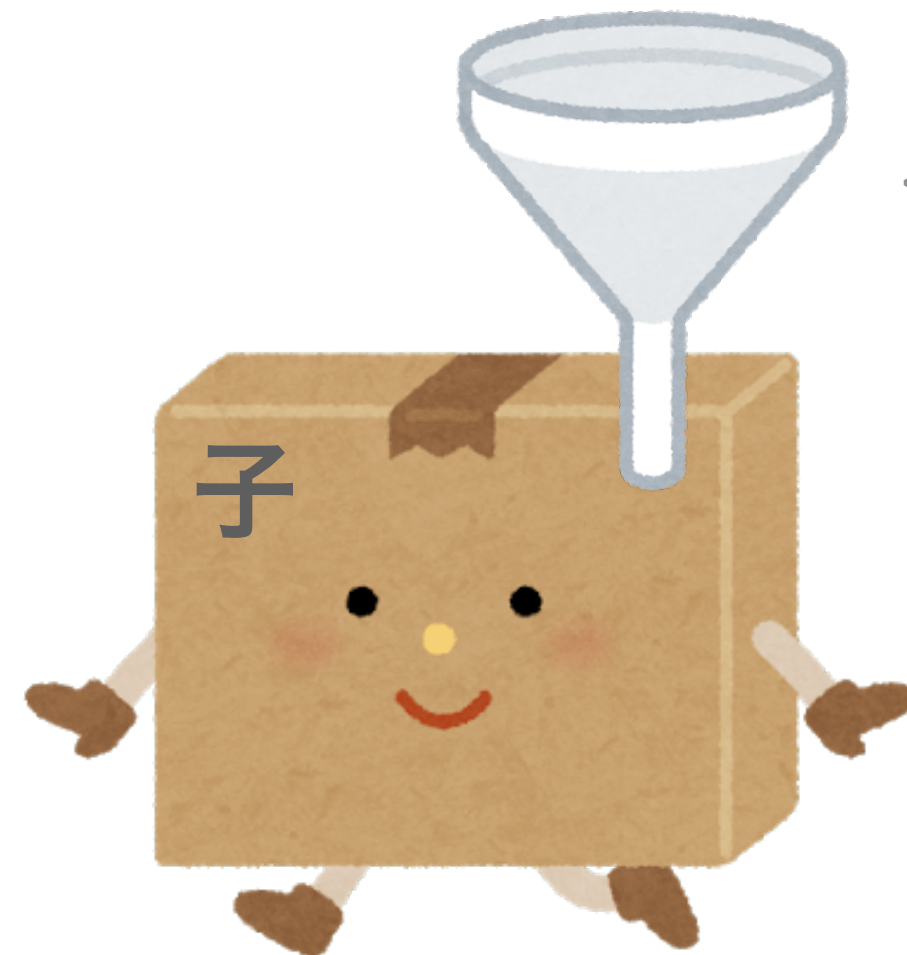
```
data(){ return { getTitle: this.title }}
```

props (プロパティ) と v-bind



親側でdataなどに設定

```
data(){ return { parentTitle: '親側のタイトル' } }
```

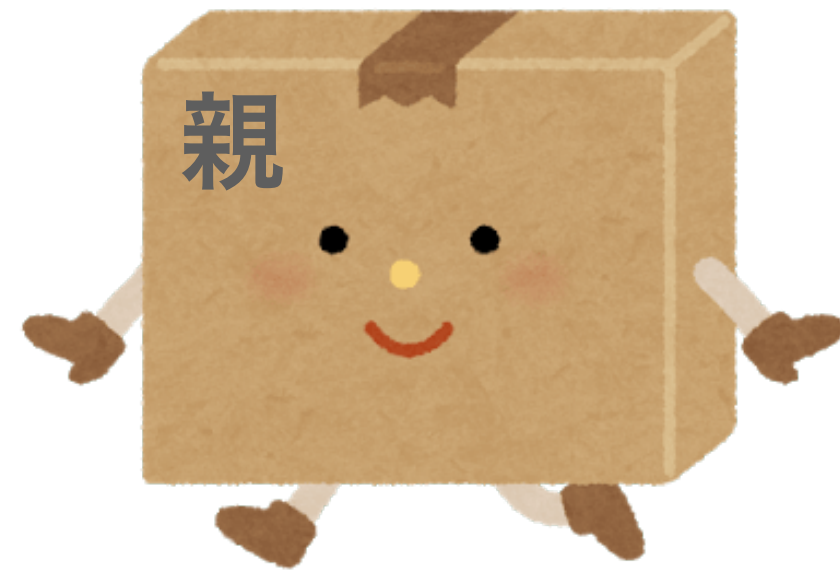


```
<my-component :title="parentTitle"></my-component>
```

子側への通り道

```
props: { title: { type: String, } }
```

props (プロパティ) と v-for



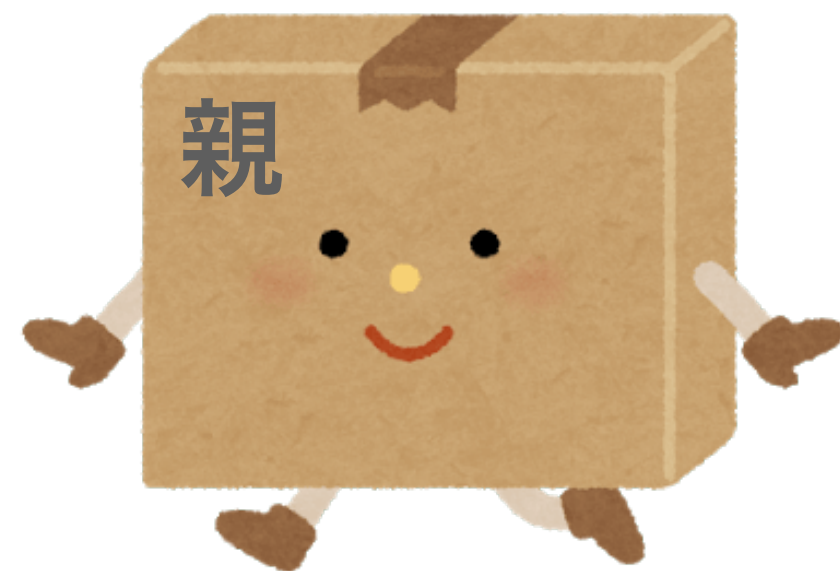
親側で配列を作成 `data(){ return { members:[{}, {}, {}] }`

```
<array-test  
  v-for="member in members"  
  :key="member.id" // keyは必須  
  :item="member"> // propsのv-bind  
</array-test>
```



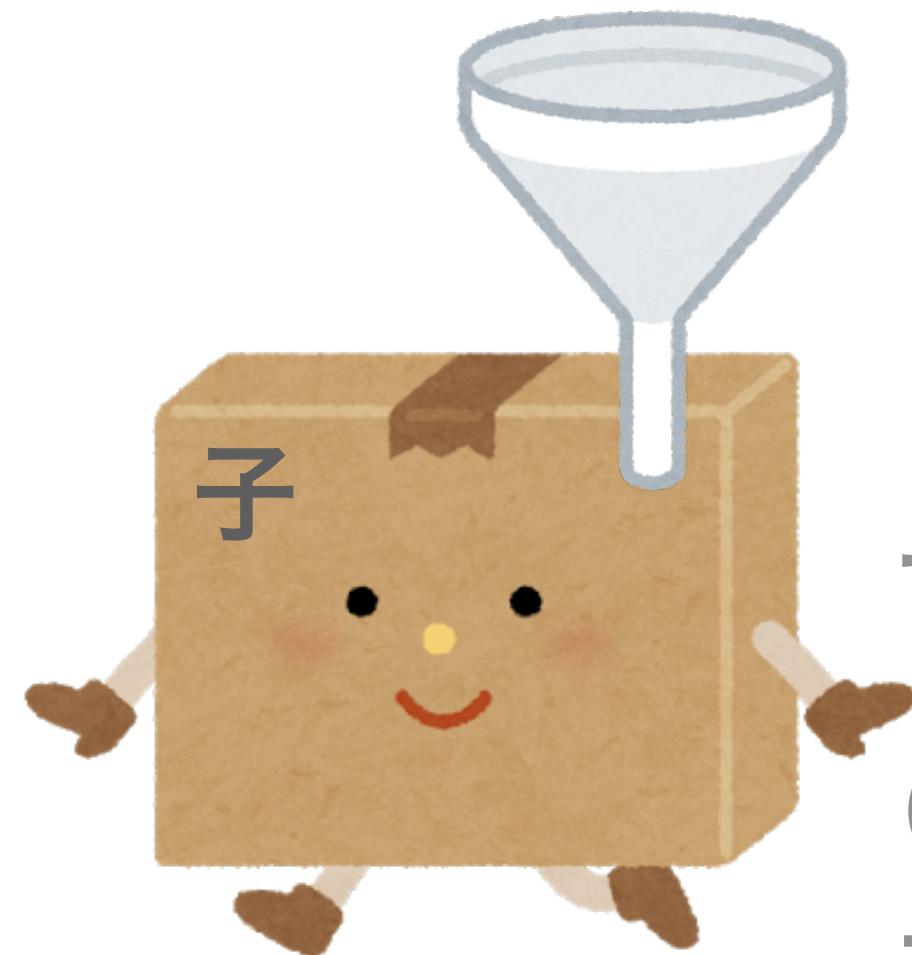
子側 `props: { item: { type: Object } }` // 配列内がオブジェクトなら Object を指定
template内は `{{ item.name }}` // props名.

props (プロパティ)の補足



props名もケバブケース

タイプがObjectかArrayで
default設定するなら関数で



```
props: { item: {  
  type: Object  
  default: () => ({ count: 0 })  
}}
```




\$emit

(カスタムイベント)

\$emit(発射・放出) カスタムイベント



```
@custom-event="親のメソッド名"  
methods:{  
  親のメソッド名(e){  
    console.log(e)  
  }  
}}
```

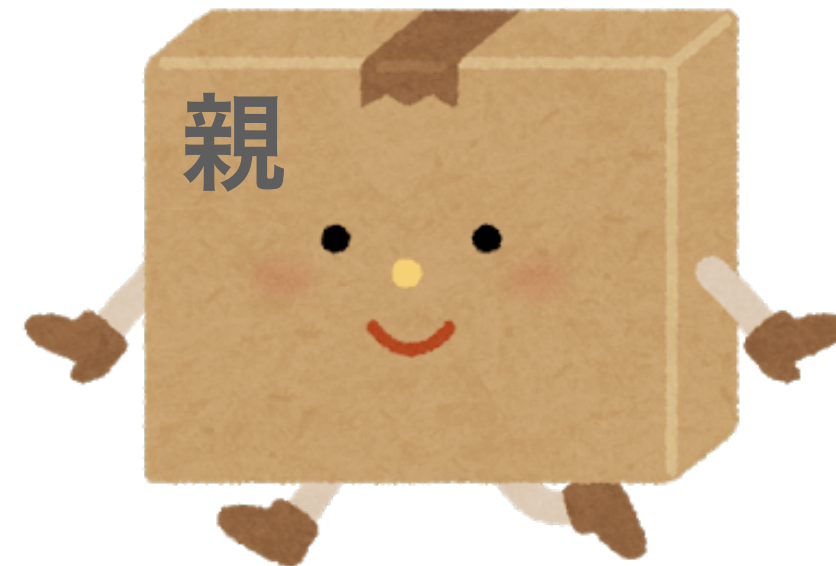
```
@click="子のメソッド名"  
methods:{  
  子のメソッド名(){
```

```
    this.$emit('custom-event', 値)}}}
```

Props Down Event Up

親側

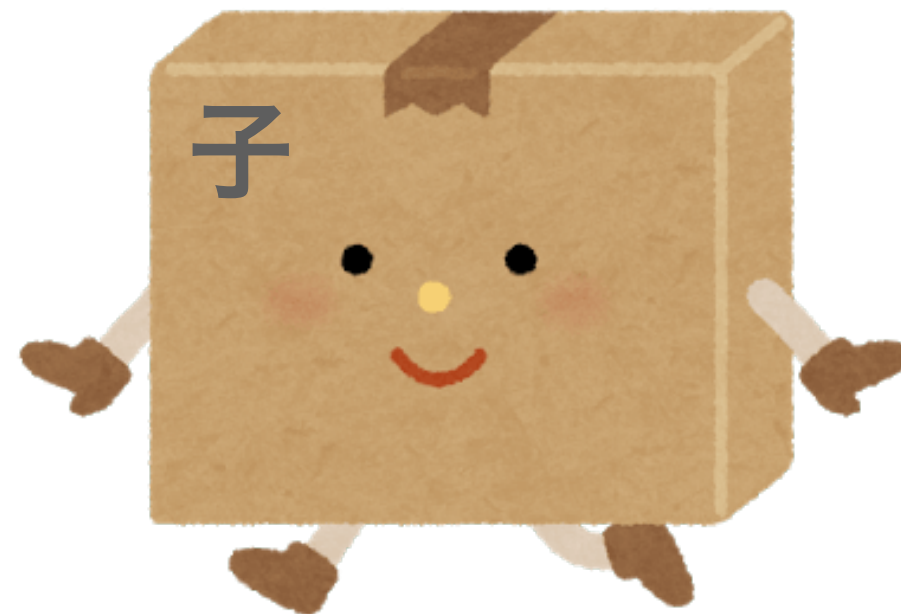
```
<v-btn :title=parent>  
data(){ return  
{ parent : '親データ' } }
```




```
<v-test @custom-event="親のメソッ  
ド名">  
methods:{  
  親のメソッド名(e){  
    console.log(e)  
  }  
}
```

子側への通り道

```
props: {  
  title: {  
    type: String,
```



```
<div @click="子のメソッド名">  
methods:{  
  子のメソッド名(){  
    this.$emit('custom-event', 値)}
```



コンポーネント間の フォーム

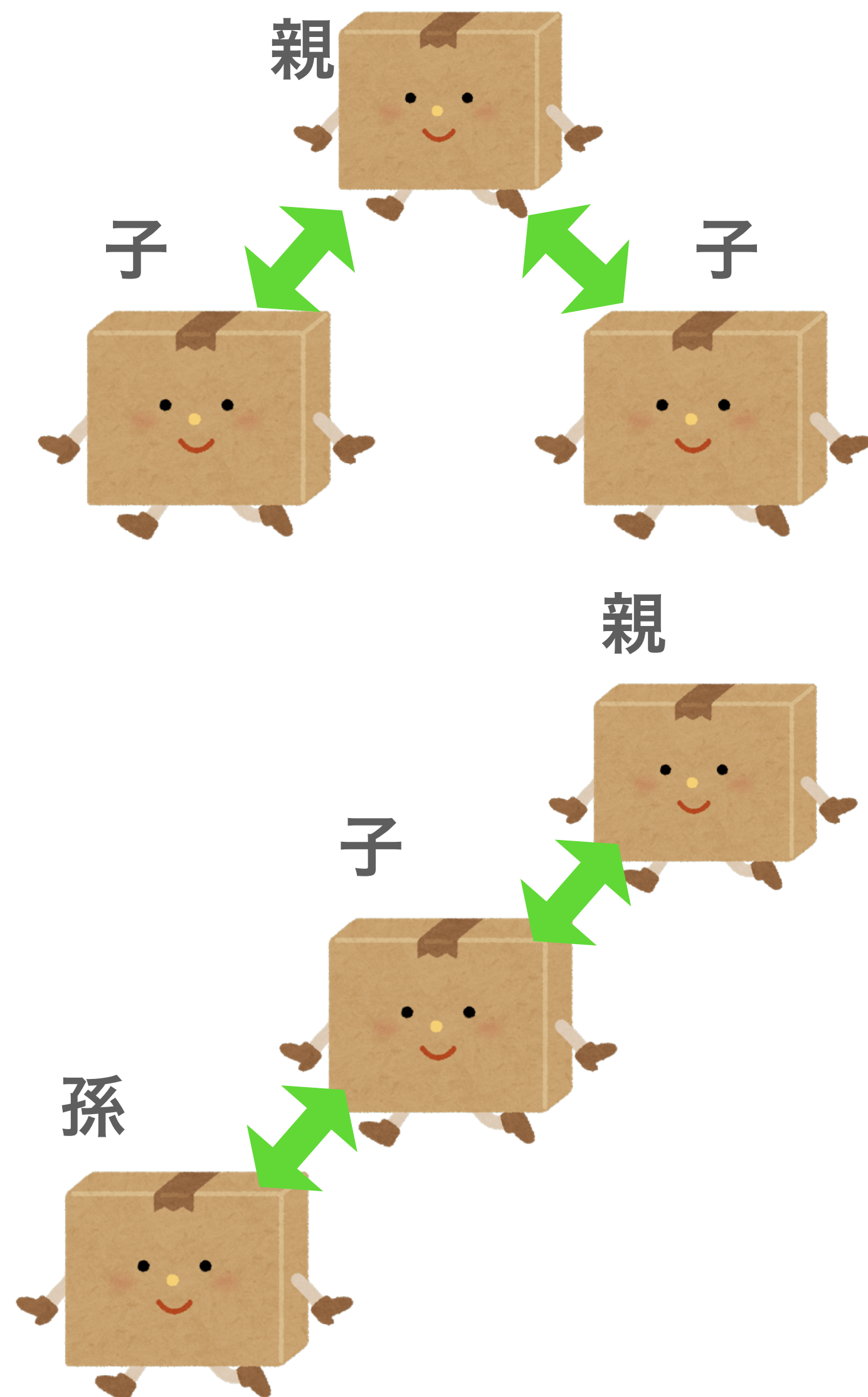
コンポーネント間のフォーム

	v-model (子でv-modelなら computed(get/set))	v-bind(:) と v-on(@)
親	<pre><custom-input v-model="parentValue" </custom-input></pre> <pre>data({ return { parentValue: '' } })</pre>	<pre><custom-input :value="parentValue" @input="parentValue = \$event"> </custom-input></pre> <pre>data({ return { parentValue: '' } })</pre>
子	<pre><input :value="value" @input="childEvent"></pre> <pre>props:{ value: { type: String } }</pre> <pre>methods:{ childEvent(e){ this.\$emit('input', e.target.value) } }</pre>	<pre><input :value="value" @input="childEvent"/></pre> <pre>props:{ value:{ type: String } }</pre> <pre>methods:{ childEvent(e){ this.\$emit('input', e.target.value) } }</pre>



親子以外のやりとり

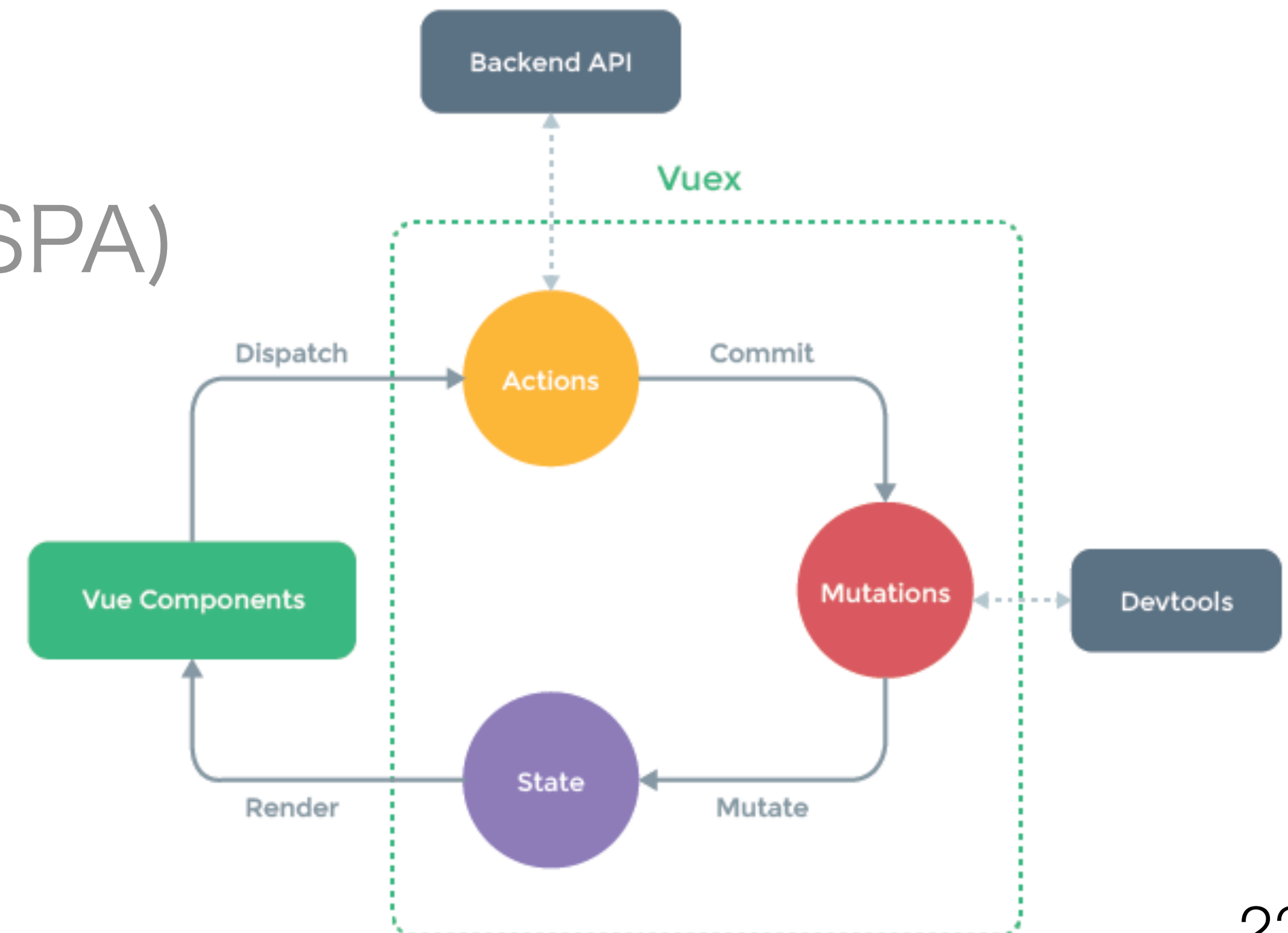
子<->子 親<->孫 など



EventBus



Vuex
(大規模SPA)



EventBus



```
let eventBus = new Vue()
```

イベント監視設定

```
mounted(){  
  eventBus.$on('event-name', callback)  
}
```

イベント発行

```
methods : {  
  eventBus.$emit('event-name' value)  
}
```

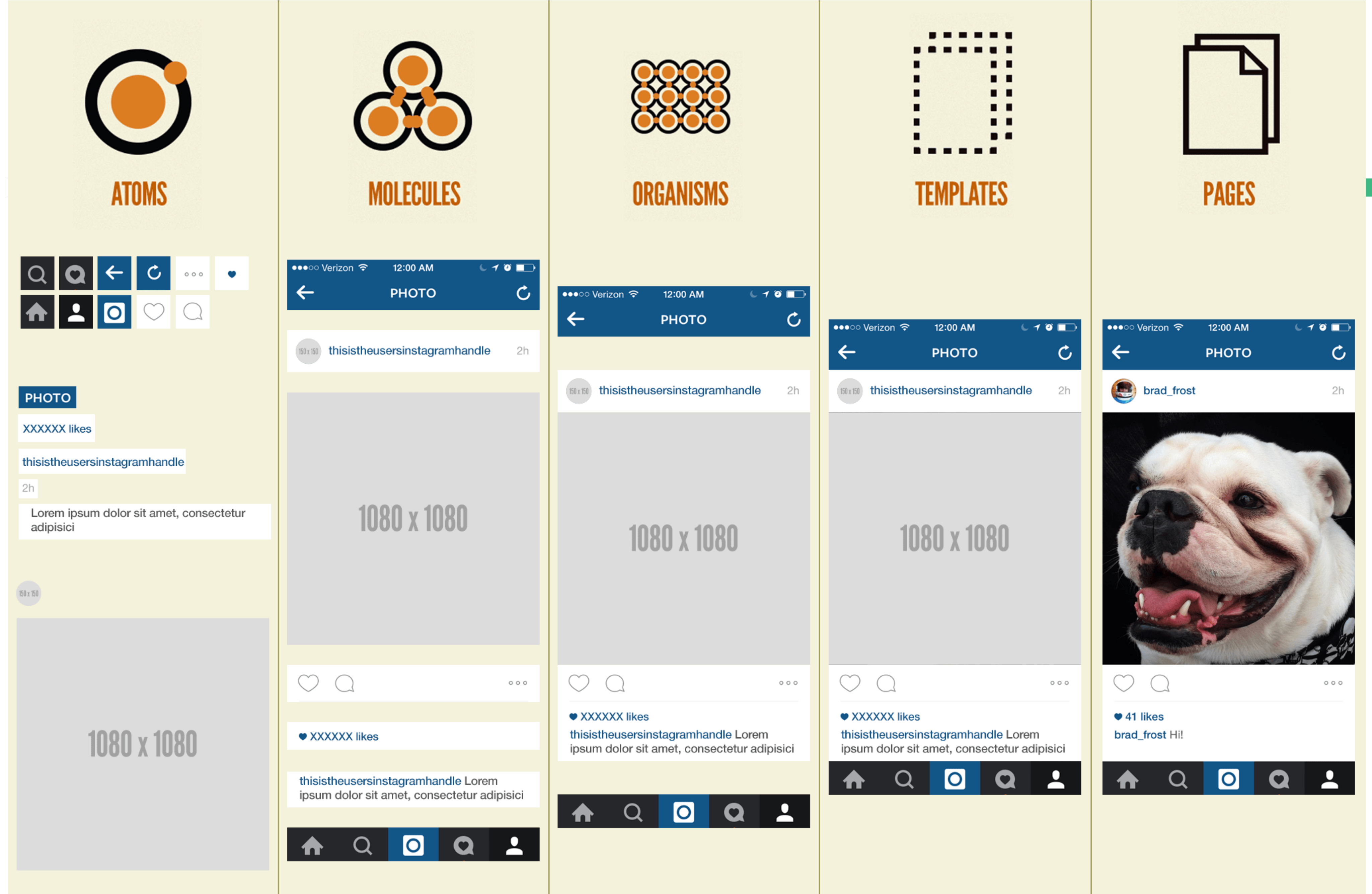



Atomic Design

AtomicDesign



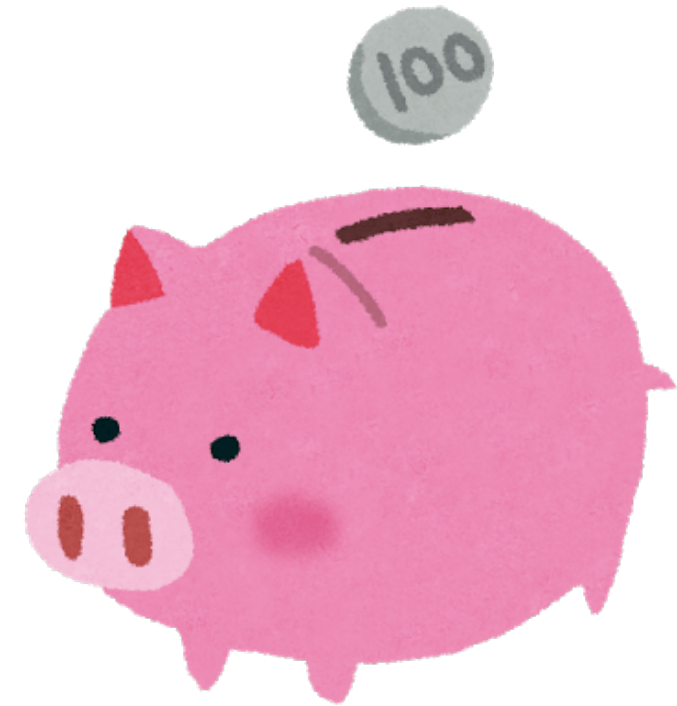
Atoms (原子) ・ ・ ボタン ・ 文字など
Molecules (分子)
Organisms (有機体)
Templates (ワイヤーフレーム)
Pages (ページ)





V-slot

スロット (差し込む)



```
<a href="xxx">google</a>  
<a href="xxx">yahoo</a>
```

親

```
<child-com>親側で書いた文字が入ります</child-com>
```

子

```
template:`<div>  
久保 <slot>南野</slot> 堂安  
</div>`
```

Slotの簡易表

	slot	名前付きslot v-slot:(#)スロット名	スコープ付きスロット v-slot:default="" #default=""
特徴	子のslot1つ	子のslot複数 (複数slot時は templateタグ)	子のdataを親で表示できる (スロットプロパティ)
親	<pre><my-com> タグ内の文章が置き換わる </my-com></pre>	<pre><my-com> <template v-slot:header> ヘッダ </template> mainに入ります。 <template #footer>フッタ </template> ここの文章もmainに入ります。 </my-com></pre>	<pre><my-com> <template v-slot="player"> {{ player.member.name }} </template> <template v-slot="{ member }"> {{ member.name }} {{ member.height }} </template> </my-com></pre>
子	<pre>テスト<slot>差し込み口</slot> テスト</pre>	<pre><slot name="header">header </slot> <slot>main</slot> <slot name="footer">footer </footer></pre>	<pre><slot :member="member"> </slot> data(){ return{ member:{name:'堂 安', height:170} }}</pre>

スコープ付きスロット



スコープ(有効範囲・使える範囲)
子のdataやmethodを親側で設定

子側 dataをslotとv-bindで紐付け(スロットプロパティ)

親側 v-slotの値として名前を指定し受け取る

v-slot:スロット名="名前" v-slot:default="player"

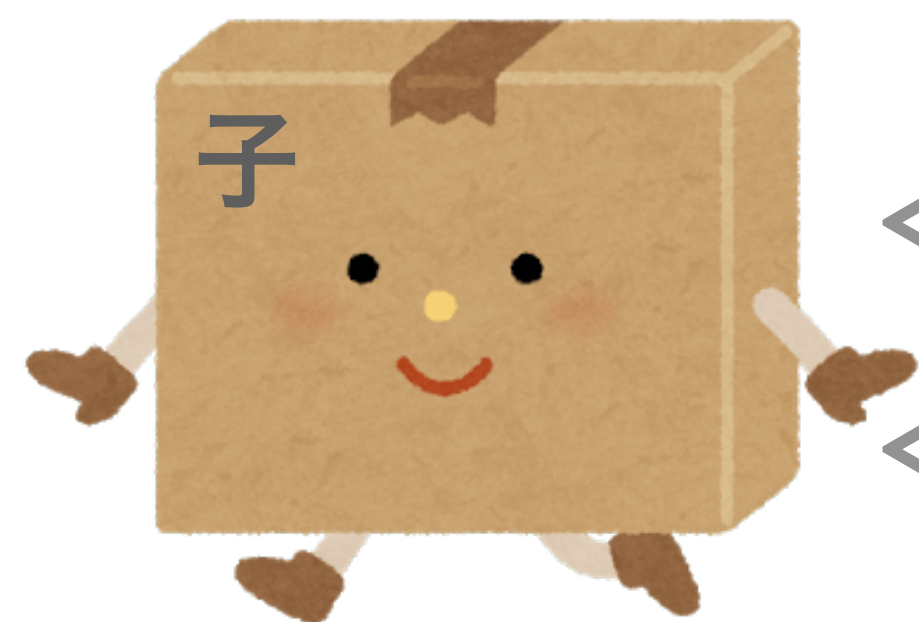
親で指定した名前.スロットプロパティ名 で表示

<https://reffect.co.jp/vue/vue-js-slot-scoped-slot>

スコープ付きスロット



```
<my-com>                                     //v-slotに値を設定
  <template v-slot="player">
    {{ player.member.name }}
  </template>
</my-com>
```



```
<slot :member="member"> //スロットプロパティ
</slot>
data(){ return { member:
{ name: 'xxx' } }}
```