




環境構築

Inertia, Vue.js3



はじめに

Laravel 第4弾に向けて

これまでudemy 3本
PHP/Laravelの講座をリリースしてきました。
第4弾に向けて、
アンケートをとることにしました。

講座一覧

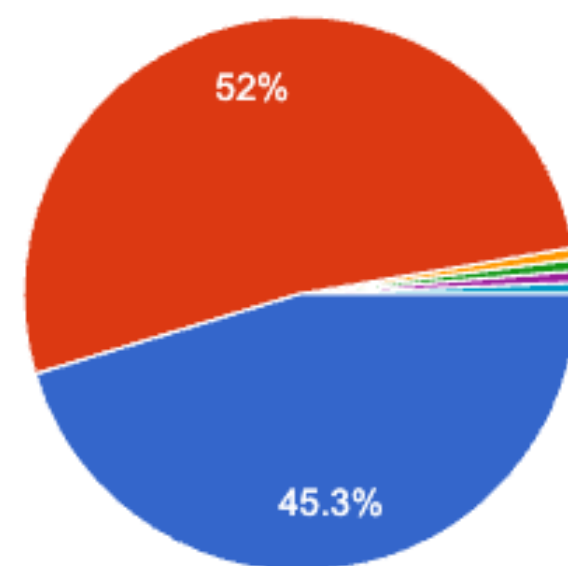
<https://coinbaby8.com/udemy-coupon-list.html>

アンケートご協力ありがとうございました。

2022年6月中旬頃
アンケート告知しまして、約150件のご回答をいただきました。

次の講座の希望システム

150 件の回答

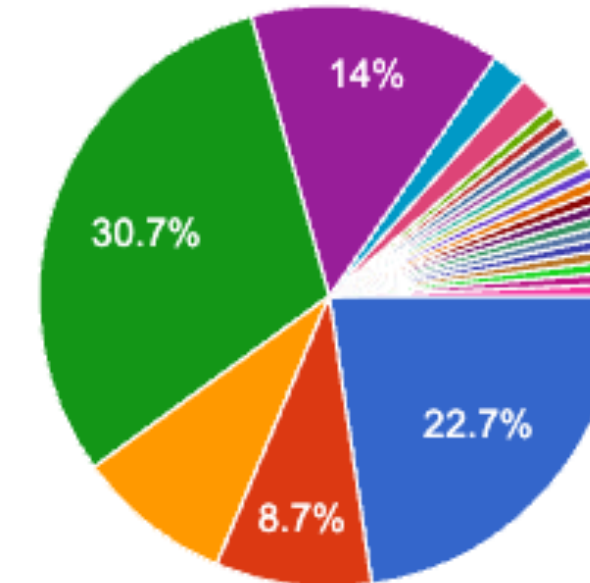


- 勤怠管理システム
- 顧客管理システム
- 両方とも
- 販売管理システム
- わからない
- Udemyやマイスビのようなコンテンツ配信ができるプラットフォームシステム



希望するフロント側

150 件の回答



- Laravel Breeze
- Laravel Jetstream + Livewire
- Laravel Jetstream + Inertia
- Vue3 + CompositionAPI
- Nuxt3
- react
- React
- React + React Hooks

▲ 1/3 ▼

詳細はブログにて

<https://coinbaby8.com/questionnaire-progress.html>

Laravel 第4弾



顧客管理システム (CRM)
Customer Relationship Management

FrontEnd


Laravel Breeze+Inertia

Vue.js3 (CompositionAPI)



環境構築

この講座で扱う環境



XAMPP/MAMP

-> PHP8.0以上 + サーバー(apache) +
mysql(mariaDB)

composer -> phpライブラリ管理ソフト

Git -> バージョン管理ソフト

GoogleChrome/Visual Studio Code

環境構築

```
# Laravel インストール  
composer create-project laravel/laravel:^9  
uCRM --prefer-dist
```

```
cd uCRM  
php artisan serve
```

Laravelマニュアル
<https://readouble.com/>

初期設定



Mysql (mariaDB) DB作成

.env 設定 (環境ファイル)

config/app.php

-> タイムゾーン、言語設定

バリデーションの言語ファイル

デバッグバーのインストール

もしDB接続できなかったら



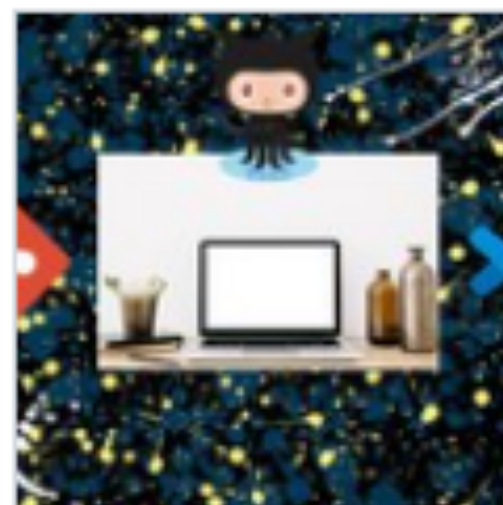
原因は様々あるようですので、
ブログを参照ください。

[https://coinbaby8.com/
access_denied.html](https://coinbaby8.com/access_denied.html)

gitHubでリポジトリ作成

この講座では
gitHubでブランチを分けながら
開発を進めていきます。

設定は必須ではありませんが
もしgit/gitHubも詳しく知りたい場合は
別講座も活用してみてください。



【Git/GitHub】【初心者向け】最短で実践力を身につけよう【VSCode】イメージ図たっぷり【わ...

ライブ

タイムゾーン、言語設定



タイムゾーン / 言語設定

config/app.php

‘timezone’ => ‘Asia/Tokyo’;

‘locale’ => ‘ja’;

デバグバ



デバグバのインストール

```
composer require barryvdh/laravel-  
debugbar
```

```
php artisan serve で確認
```

.envのDEBUGで表示切り替えできる

.envファイル更新反映しないなら



```
php artisan cache:clear
```

```
php artisan config:clear
```

これらのコマンドでキャッシュを消して
再度試してみてください。

言語ファイル



lang/jaを作成

auth.php

pagination.php

passwords.php

validation.php

をそれぞれマニュアルからコピー



Laravel Breeze (vue Inertia)

認証ライブラリの比較

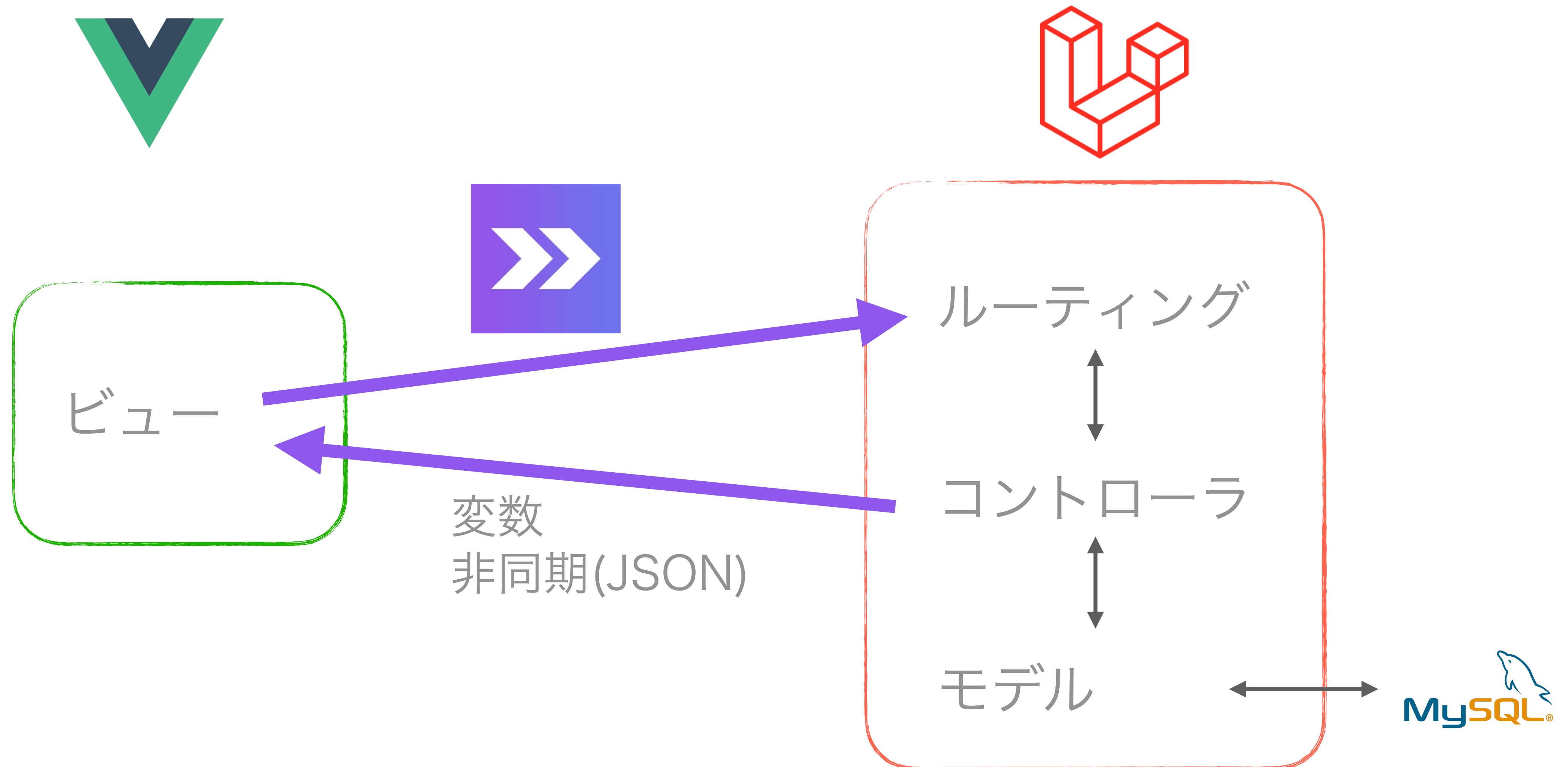
	Laravel / ui	Laravel Breeze	Fortify	Jetstream
Version	6.x～	8.x～	8.x～	8.x～
View (PHP)	Blade	Blade / Vue.js	-	Livewire + Blade
JS	Vue.js / React.js	Alpine.js / Inertia + Vue.js	-	Inertia.js + Vue.js
CSS	Bootstrap	Tailwindcss	-	Tailwindcss
追加ファイル	View/Controller/Route	View/Controller/Route	-	View/Controller/Route
機能1	ログイン、ユーザー登録、パスワードのリセット、 メール検証、パスワード確認			-
機能2	-	-	2要素認証、 プロフィール管理、チーム 管理	APIサポート (Sanctum)

Inertia(イナーシャ)の概要

Laravelなどサーバー側フレームワークの
機能を活かしながら、
SPAをつくるためのライブラリ
(SinglePageApplication)
Vue.js、React、Svelteなどにも対応。

Inertia公式ページ
<https://inertiajs.com/>

イメージ図



Breeze(vue)をインストール



```
composer require laravel/breeze:^1 --  
dev
```

```
php artisan breeze:install vue  
npm install && npm run dev
```

追記: 2023年1月以降

Inertia ver1.0に伴い
インストールされるライブラリが変更になりました。
動画と合わせるため
package.jsonに下記ライブラリを追記し、
npm install 実施をお願いします。

```
"@inertiajs/inertia": "^0.11.0",  
"@inertiajs/inertia-vue3": "^0.6.0",
```

マニュアル <https://legacy.inertiajs.com/>
アップグレードガイド <https://inertiajs.com/upgrade-guide>

追記: 2023年1月以降

app.jsファイルも変更必要になります。

resources/js/app.js

変更前

```
import { createInertiaApp } from '@inertiajs/vue3';
```

変更後

```
import { createInertiaApp } from '@inertiajs/inertia-vue3';
```


仮想サーバー

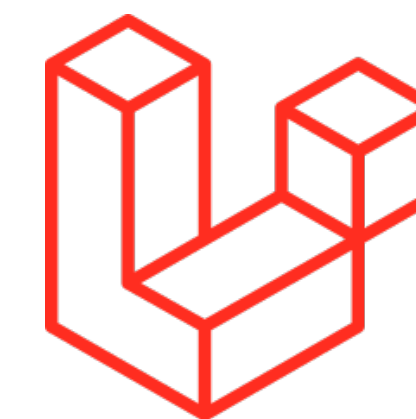
ターミナルのタブを3つ開いておく

1. フロント側の簡易サーバー用
(ホットリロード対応)

`npm run dev`



2. アプリ側の簡易サーバー用
`php artisan serve`



3. 各種コマンド実行用

開発環境+ α

GoogleChrome 拡張機能

Vue.js devtools 6.x

Visual Studio Code

Volar (Vue.js3 script setup対応)

Tailwind CSS Intellisense

後はお好きに

Dracula Theme, Highlight Matching Tag

Laravel Extra Intellisense, Git Graph

PHP Intelephense, Laravel Snippets

Material Icon Theme, Dot Env

ファイルを見してみる

routes/web.php

Inertia::render()

app/Http/Controllers/Auth

return Inertia::render()

app/Http/Middleware

HandleInertiaRequests.php

resources/js/

Components, Layouts, Pages

(Vue.js SFC多数 (Single File Component))

resources/views/app.blade.php

@inertia

resources/js/app.js



app.blade.phpの中で
@viteで読み込んでいるファイル

エントリーポイント(最初に読み込まれる
ファイル)

createInertiaApp // Inertia使用コード
createApp // Vue3使用コード



Inertia

動作確認

LaravelBladeとInertiaの違い



サーバーサイド

Laravel Blade

`view('viewファイル名', compact(変数名))`

Inertia

`Inertia::render('コンポーネント名', [変数名])`

LaravelBladeとInertiaの違い

クライアントサイド

Laravel Blade

`リンク`

ページ内全ての情報を再読み込みする HTML

Inertia

`<Link href="">`

部分的に読み込む JSON

読み込む量が少ない = 描画速度が早い

SPA (Single Page Application)

Linkを確認してみる

```
routes/web.php
Route::get('/inertia-test', function () {
    return Inertia::render('InertiaTest');
});
```

ルートの確認は `php artisan route:list`

resourcesにファイル作成
resources/js/Pages/InertiaTest.vue
(コンポーネント名はパスカルケース推奨)

resources/js/Pages/InertiaTest.vue

```
<script setup>
```

```
import { Link } from '@inertiajs/inertia-vue3';
```

```
</script>
```

```
<template>
```

```
  Inertiaテストです。 <br>
```

```
  <a href="/">aタグでWelcomeに移動</a><br>
```

```
  <Link href="/">LinkでWelcomeに移動</Link>
```

```
</template>
```


GoogleChrome開発ツール

Networkタブを開いた状態でリンクをクリック

localhostを確認 (Status Code : 200 OK)

通信は Request -> Response の順

aタグ経由

Content-Type: text/html

Linkコンポーネント経由

Content-Type: application/json

X-Inertia: true

X-Requested-With: XMLHttpRequest (XHR 非同期)

X-XSRF-TOKEN

名前付きルート その1

ziggyライブラリにより名前付きルートが使える

InertiaTest.vue

```
<Link :href="route('inertia.index')"></Link>
```

routes/web.php

```
use App\Http\Controllers\InertiaTestController;
```

略

```
Route::get('/inertia/index', [InertiaTestController::class,  
'index'])->name('inertia.index');
```

名前付きルート その2

```
php artisan make:controller InertiaTestController
```

```
use Inertia\Inertia;  
    public function index()  
    {  
        return Inertia::render('Inertia/Index');  
    }  
}
```

```
resources/js/Pages/Inertia/Index.vue
```

リソース (RestFul) コントローラ

CRUDのパターン

動詞	URI	アクション	ルート名
GET	/photos	index	photos.index
GET	/photos/create	create	photos.create
POST	/photos	store	photos.store
GET	/photos/{photo}	show	photos.show
GET	/photos/{photo}/edit	edit	photos.edit
PUT/PATCH	/photos/{photo}	update	photos.update
DELETE	/photos/{photo}	destroy	photos.destroy

一覧表示

詳細表示

ルートパラメータ その1

resources/js/Pages/InertiaTest.vue

```
<Link :href="route('inertia.show', { id : 1 })">button</Link><br>
```

routes/web.php

```
Route::get('/inertia/show/{id}', [InertiaTestController::class,  
'show'])->name('inertia.show');
```

InertiaTestController.php

```
public function show($id)  
{  
    // dd($id);  
    return Inertia::render('Inertia/Show', [  
        'id' => $id  
    ]);  
}
```

ルートパラメータ その2

resources/js/Pages/Inertia/Show.vue

```
<script setup>
defineProps({ // 変数の受け取り
  id: String
})
</script>

<template>
  {{ id }} // propsで渡ってきた変数
</template>
```

Linkコンポーネントの属性

Vue-routerのrouter-linkのようにいくつかの属性を設定できる

`type="button"` // buttonリンク

`as="button"` // button

`method="post"` // メソッド変更

`:data="{}"` // 送信リクエストにデータ追加

オブジェクトかFormDataインスタンス

`:headers="{}"` 追加するHTTPヘッダー指定

`replace` History履歴書き換え

`preserve-state` フォーム入力値を維持


`preserve-scroll` 移動前と同じスクロール位置を保持

部分リロード

`:only="{}"` // 特定のデータだけ読み込みたい場合
(通信量が節約される。

Laravel側でデータは取得しているので、
サーバー側の負荷は変わらない。

Laravel側も必要な時だけ取得するなら `fn()` を挟むとok
`return Inertia::render('コンポーネント', [key => fn() =>
value]);`



Linkコンポーネントで store処理

store処理 事前準備

```
php artisan make:model InertiaTest -m
```

```
database/migrations/xxx_create_inertia_tests_table.php
```

```
public function up()
{
    Schema::create('inertia_tests', function (Blueprint $table) {
        $table->id();
        $table->title();
        $table->content();
        $table->timestamps();
    });
}
```

```
php artisan migrate // DBに反映
```

ビュー側

resources/js/Pages/InertiaTest.vue

```
<script setup>
```

```
import { ref } from 'vue' // リアクティブ
```

```
const newTitle = ref("")
```

```
const newContent = ref("")
```

```
</script>
```

```
<template>
```

```
  <input type="text" name="newTitle" v-model="newTitle">{{ newTitle }}<br>
```

```
  <input type="text" name="newContent" v-model="newContent">{{ newContent }}
```

```
<br>
```

```
  <Link as="button" method="post" :href="route('inertia.store')" :data="{ title:  
newTitle, content: newContent}">保存テスト</Link>
```

```
</template>
```


ルート->コントローラ

routes/web.php

```
Route::post('/inertia', [InertiaTestController::class, 'store'])->name('inertia.store');
```

InertiaTestController.php

```
use App\Models\InertiaTest;
```

略

```
public function store(Request $request)
{
    $inertiaTest = new InertiaTest;
    $inertiaTest->title = $request->title;
    $inertiaTest->content = $request->content;
    $inertiaTest->save();

    return to_route('inertia.index');
}
```




Vue.jsの概要

Vue.js 年表



2014/02 Vue.js 0.8 リリース

2015/04 Laravelで採用

2016/10 Vue.js 2 リリース
OptionsAPI

2020/09 Vue.js 3 リリース
CompositionAPI (大規模対応+TypeScript対応)

2021/08 Vue.js 3.2 リリース
script setup (CompositionAPIを
シンプルに書ける)

Vue.js SingleFileComponent

HTML + JS + CSS を1つのファイル (*.vue)

<template> よく使うディレクティブ

v-if, v-else, v-else-if

v-for (v-for in), v-show

v-text, v-html

v-on (省略形は@) イベント

v-bind (省略形は:) 紐付け

v-model フォーム用

v-cloak

v-slot (省略形は#)

トランジション関連

Vue.js CompositionAPI(setup)



<script setup>

必要な機能はimportする

```
import { Link } from '@inertia/inertia-vue3' //
```

inertia側

```
import { ref } from 'vue' // vue側
```

コントローラからの受け渡し

```
defineProps({ id: String })
```

リアクティブな変数は refかreactiveで囲む

関数(メソッド)はJSと同じように書ける

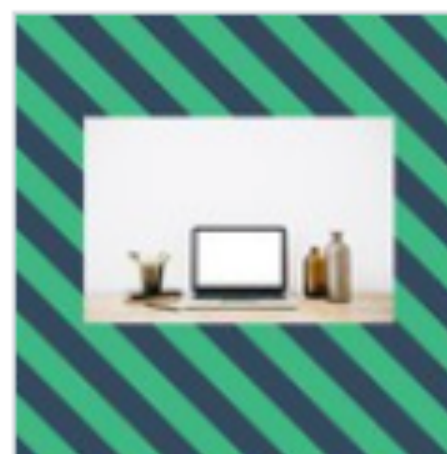
関連マニュアル・講座

Vue.js3 ドキュメント

<https://v3.ja.vuejs.org/guide/introduction.html>

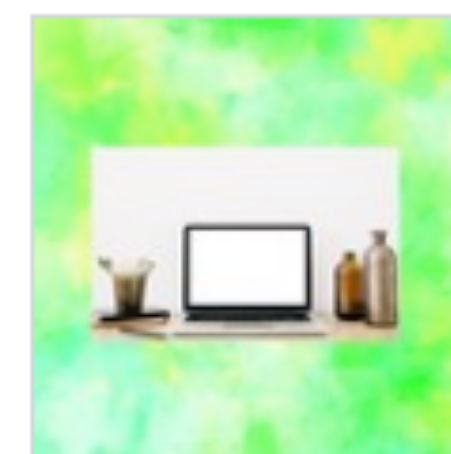
Vue.js3 APIリファレンス

<https://v3.ja.vuejs.org/api/>



【Vue.js2&Vue.js3対応】基礎から【Vuetify】を使った応用まで！超初心者から最短距離でレベル...

ライブ



JavaScriptをとことんやってみよう【超初心者から脱初心者へレベルアップ】【わかりやすさ重...

ライブ



フォーム create処理

create ルート->コントローラ

routes/web.php

```
Route::get('/inertia/create', [InertiaTestController::class,  
'create'])->name('inertia.create');
```


InertiaTestController.php

```
public function create()  
{  
    return Inertia::render('Inertia/Create');  
}
```

create ビュー

resources/js/Pages/Inertia/Create.vue

```
<script setup>
import { reactive } from 'vue'
import { Inertia } from '@inertiajs/inertia'
const form = reactive({
  title: null,
  content: null
})
const submitFunction = () => {
  Inertia.post('/inertia', form)
}
</script>
<template>
  <form @submit.prevent="submitFunction">
    <input type="text" name="title" v-model="form.title">
    <input type="text" name="content" v-model="form.content">
    <button>送信</button>
  </form>
</template>
```

バリデーション

バリデーション



クライアントサイド(ブラウザ側)

リアルタイムで検知できる

しっかり対応するならvee-validateなどの
ライブラリも要検討

開発ツールなどで無効化できる

サーバーサイド

Laravelのバリデーションがそのまま使える

バリデーション コントローラ側

InertiaTestController.php

```
public function store(Request $request)
{
    $request->validate([
        'title' => ['required', 'max:20'],
        'content' => ['required'],
    ]);
    略
}
```

View側に errors というオブジェクトが渡る


バリデーション ビュー側

Inertia/Create.vue

```
<script setup>
defineProps({
  errors: Object
})
略
</script>
<template>
  <div v-if="errors.title">{{ errors.title }}</div>
  <div v-if="errors.content">{{ errors.content }}</div>
</template>
```


GoogleChrome開発ツールのVueタブからも確認できる

エラーの日本語化対応



lang/ja/validation.php

```
'attributes' => [  
    'title' => '件名',  
    'content' => '本文'  
],
```



CSRF対策(実施済み)

CSRF

CSRF(シーサーフ)
(Cross-Site Request Forgeries(偽造) の略)

悪意のあるURLにアクセスし、
思わぬリクエスト(情報)を利用されてしまう

対策： 正しいページからアクセスがきているか
トークンを発行して確認する

Laravel @csrfで対応

Inertia 既に対応されている (X-XSRF-TOKEN)



フラッシュメッセージ

フラッシュメッセージ

登録・保存・削除した際に
1度だけ表示させるメッセージ

Inertia用のミドルウェアを使うと
コントローラ → ビュー に
データを渡す際に便利

readoubleマニユアルの レスポンス フラッシュデータ
readoubleマニユアルの セッション
InertiaマニユアルのShared Data

フラッシュメッセージ コントローラー

InertiaTestController.php

```
public function store(Request $request)
```

```
{  
  略
```

```
  return to_route('inertia.index')
```

```
    ->with([
```

```
      'message' => '登録しました。'
```

```
    ]);
```

```
}
```


HandleInertiaRequests

app/Http/Middleware/HandleInertiaRequests

```
public function share(Request $request)
{
    return array_merge(parent::share($request), [
        'flash' => [
            'message' => fn() => $request->session()->get('message')
        ],
    ]);
}
```


fn() => は PHP7.4からのアロー関数
関数を挟むと 必要な時だけ呼び出される

フラッシュメッセージ ビュー側



resources/js/Pages/Inertia/Index.vue

```
<div v-if="$page.props.flash.message">  
  {{ $page.props.flash.message }}  
</div>
```

データのリスト表示

コントローラ側

Laravelでモデル経由でデータ取得する場合
配列を拡張したコレクション型になる


InertiaTestController.php

```
use App\Models\InertiaTest;
```

```
public function index()
{
    return Inertia::render('Inertia/Index', [
        'blogs' => InertiaTest::all()
    ]);
}
```

Inertia/Index.vue

```
<script setup>
import { Link } from '@inertiajs/inertia-vue3';
defineProps({ blogs: Array // 配列 })
</script>
<template>
<ul>
  <li v-for="blog in blogs" :key="blog.id"> // 単数形 in 複数形 :keyもつける
    (ソートや削除などで順序変わっても状態を保持するため)
    件名: <Link class="text-blue-400"
      :href="route('inertia.show', {id: blog.id })">{{ blog.title }}</Link>,
    本文: {{ blog.content }}
  </li>
</ul>
</template>
```

イベントコールバック

イベントコールバック

処理の前後にフックさせて処理する仕組み

削除 -> 本当に削除しますか？ と表示させるなど

Inertiaマニュアル manual visitsの後半

```
Inertia.post('/users', data, {  
  onBefore: (visit) => {},  
  onStart: (visit) => {},  
  onProgress: (progress) => {},  
  onSuccess: (page) => {},  
  onError: (errors) => {},  
  onCancel: () => {},  
  onFinish: visit => {},  
})
```

ルート->コントローラ

routes/web.php

```
Route::delete('/inertia/{id}', [InertiaTestController::class, 'delete'])->name('inertia.delete');
```

InertiaTestController.php

```
public function show($id)
{
    return Inertia::render('Inertia/Show', [
        'id' => $id,
        'blog' => InertiaTest::findOrFail($id)
    ]);
}
```

ルート->コントローラ

InertiaTestController.php

```
public function delete($id)
{
    // 削除処理
    $blog = InertiaTest::findOrFail($id);
    $blog->delete();


    return to_route('inertia.index')
        ->with([ 'message' => '削除しました。' ]);
}
```

Inertia/Show.vue

```
<script setup>
import { Inertia } from '@inertiajs/inertia'
defineProps({ id: String, blog: Object })

const deleteConfirm = id => {
  // JSのテンプレート構文 バックスラッシュで囲む 変数箇所を${}で囲む
  Inertia.delete(`/inertia/${id}`, {
    onBefore: () => confirm('本当に削除しますか?')
  })
}
</script>

<template>
  {{ id }}<br> {{ blog.title }}<br>
  <button @click="deleteConfirm(blog.id)">削除</button>
</template>
```

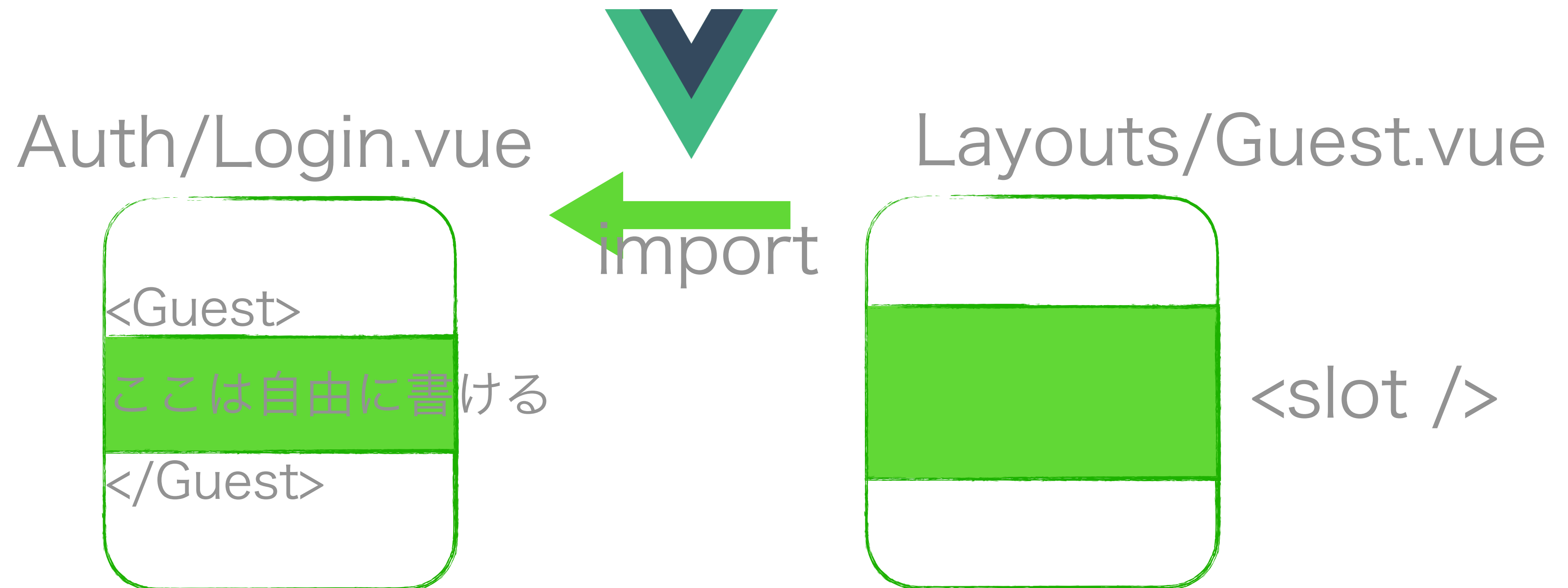



スロット

スロット Vue.jsの機能

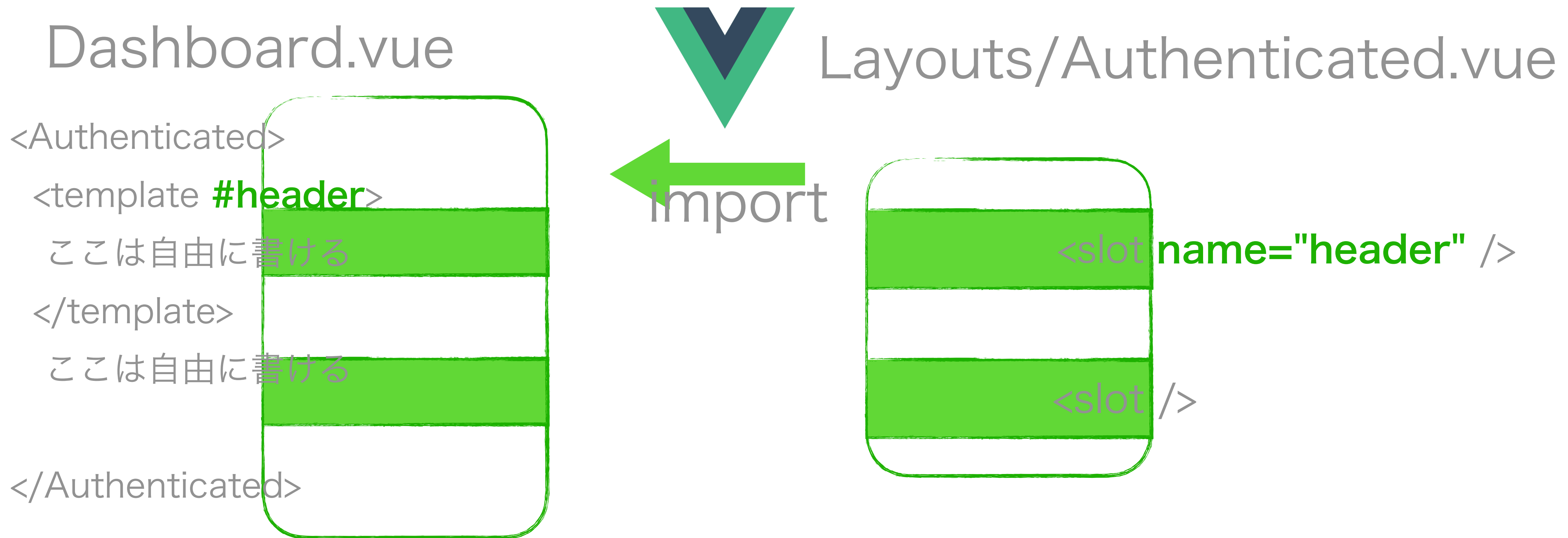
コンポーネントの中に


別のコンポーネントを差し込む



スロット Vue.jsの機能

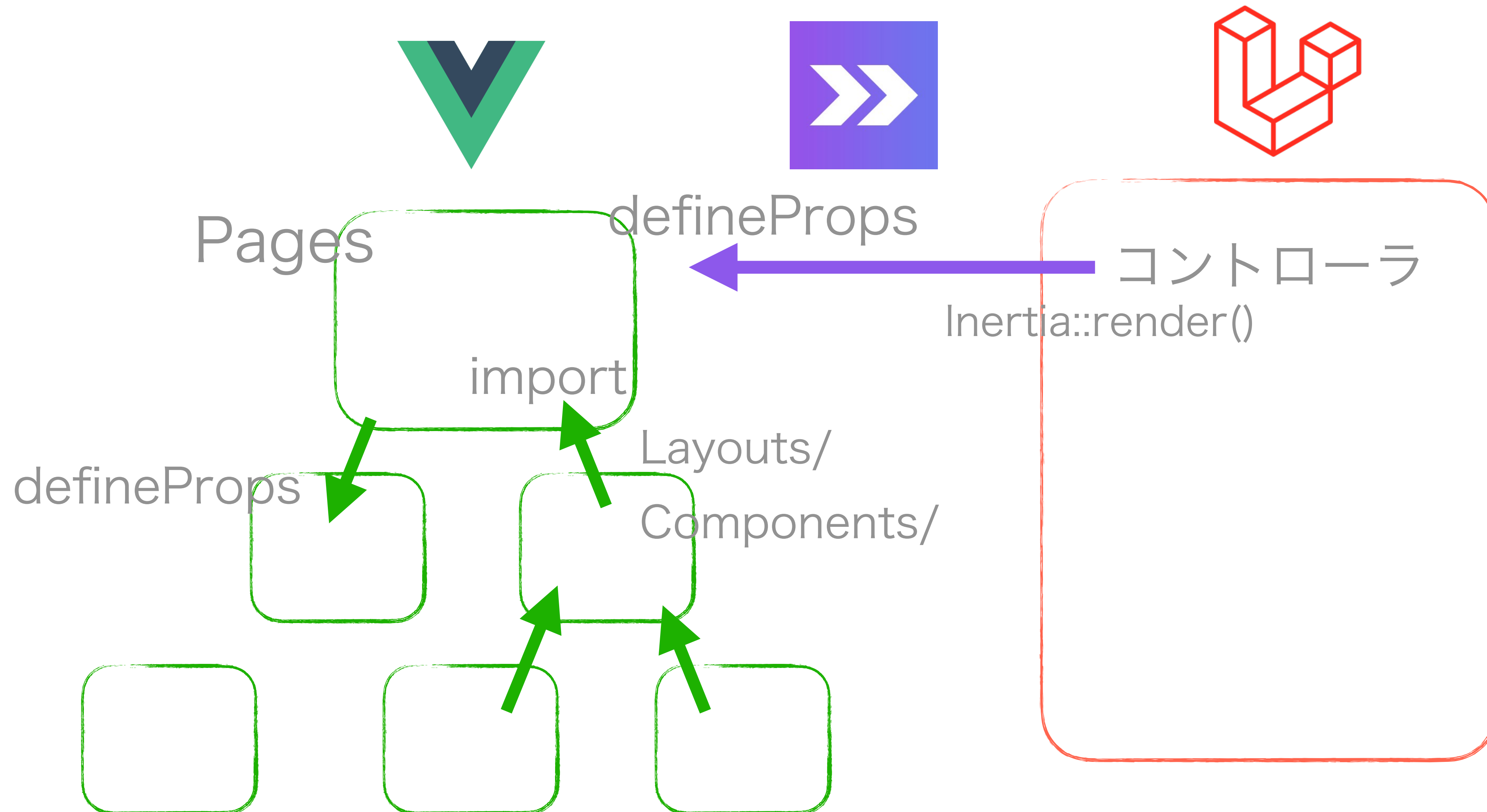
複数差し込む場合は名前付きスロットなど





他のコンポーネントを
使ってみる

他のコンポーネントを使ってみる



ルート



`routes/web.php`

```
Route::get('/component-test', function () {  
    return  
    Inertia::render('ComponentTest');  
});
```

ビュー側

resources/Pages/ComponentTest.vue

```
<script setup>
```

```
import Label from '@components/Label.vue'
```

```
import Input from '@components/Input.vue'
```

```
import GuestLayout from '@Layouts/Guest.vue'
```

```
</script>
```

```
<template>
```

```
  <GuestLayout>
```

```
    <Label>タイトル</Label>
```

```
    <Input modelValue="初期値です"></Input>
```

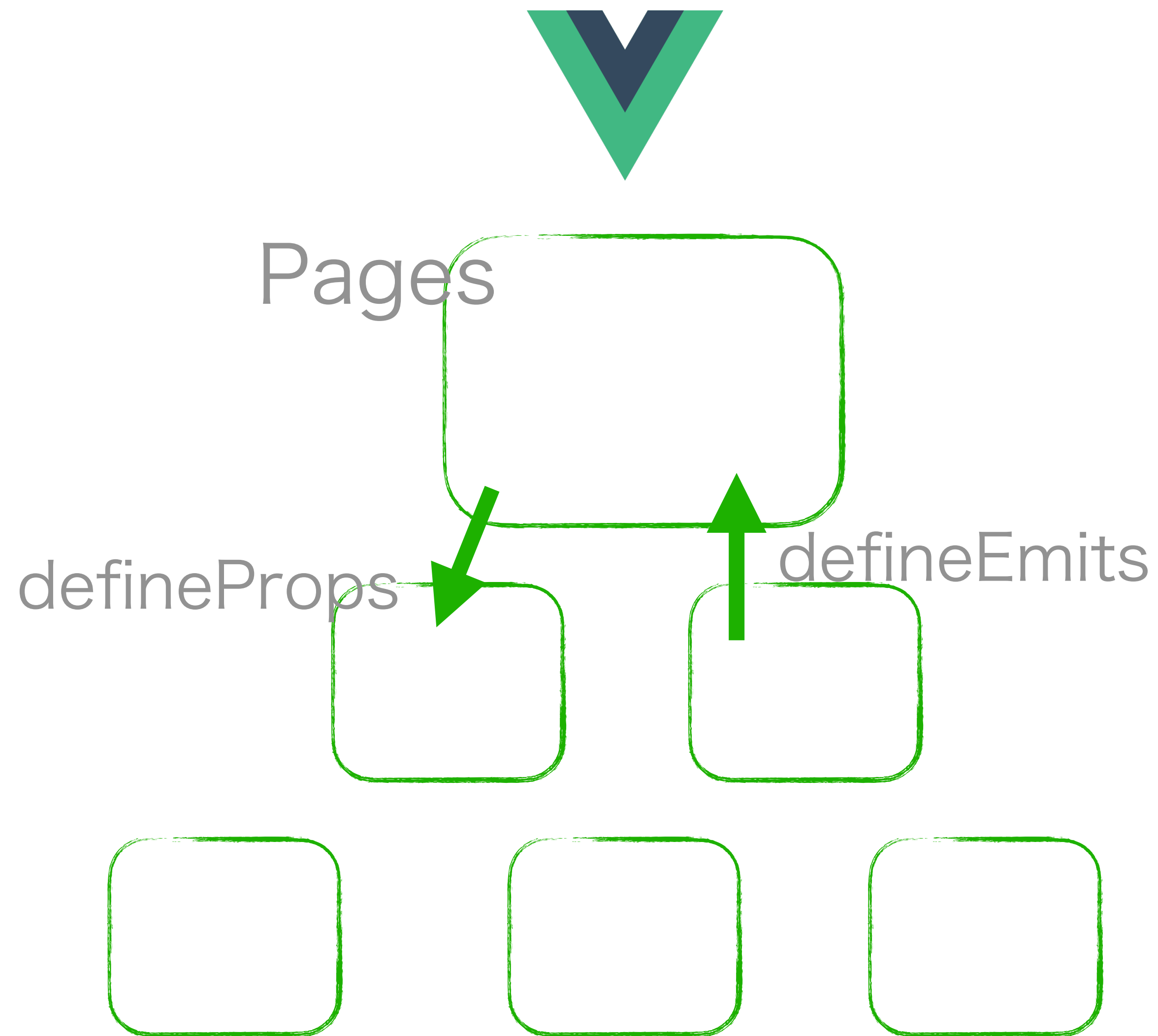
```
  </GuestLayout>
```

```
</template>
```


An abstract background on the left side of the slide, featuring bold, diagonal brushstrokes in various shades of pink, magenta, and purple. The strokes are layered and textured, creating a sense of movement and depth.

defineEmits

defineEmits



親->子 props

親<-子 emit

Props down Event up

とも呼ばれる

ビュー側

resources/Pages/ComponentTest.vue (親)

```
const emitTest = e => console.log(e)
```

```
<Input @update:modelValue="emitTest"></Input>
```

resources/Components/Input.vue (子)

```
defineEmits(['update:modelValue']); // カスタムイベント名
```

\$emit(カスタムイベント名、渡したい値)

```
<input @input="$emit('update:modelValue', $event.target.value)">
```