

# Inertia 3 cheat sheet for Laravel and Vue

Responses, prop types (deferred, optional, once, merge), useForm, Link, partial reloads, shared data, validation errors and SSR with the @inertiajs/vite plugin.



Inertia 3 · Laravel · Vue 3

## Install and boot

### Packages

```
composer require inertiajs/inertia-laravel:^3.0
npm install @inertiajs/vue3@3.0 @inertiajs/vite@3.0
```

### resources/js/app.ts

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

createInertiaApp({
  pages: './Pages', // resolved by the
  @inertiajs/vite plugin
  progress: { color: '#6936D3' },
})
```

## Responses

### Render a page

```
use Inertia\Inertia;

return Inertia::render('Event/Show', [
  'event' => $event->only('id', 'title', 'start_date'),
]);

Route::inertia('/about', 'About'); // no controller
needed
return to_route('users.index'); // redirects after
POST/PUT/DELETE
```

### Flash data (not kept in history)

```
Inertia::flash('message', 'User created.');
```

```
return Inertia::flash('newUserId', $user->id)->back();
```

## Props: lazy, optional, deferred, once, merge

### Prop types

```
return Inertia::render('Users/Index', [
  'users' => fn () => User::all(),
  // lazy: only if requested
  'companies' => Inertia::optional(fn () =>
Company::all()), // never on first load
  'roles' => Inertia::always(Role::all()),
  // even on partial reloads
  'permissions' => Inertia::defer(fn () =>
Permission::all()), // fetched after render
  'teams' => Inertia::defer(fn () => Team::all()),
  'sidebar', // grouped request
  'plans' => Inertia::once(fn () => Plan::all()),
  // cached across visits
  'posts' => Inertia::merge(fn () => Post::paginate())-
>append('data', matchOn: 'id'),
  'chat' => Inertia::deepMerge($chatData),
]);
```

*Inertia::lazy() is gone in v3: use Inertia::optional(). Prop types now work inside nested arrays and closures.*

### <Deferred> in Vue

```
<Deferred data="permissions">
  <template #fallback><p>Loading...</p></template>
  <template #default="{ reloading }">
    <PermissionList :items="permissions" :class="{ dim:
reloading }" />
  </template>
</Deferred>
<WhenVisible data="comments"><template #fallback>...</template>
</WhenVisible>
```

## Forms with useForm

### Submit, errors, progress

```
<script setup lang="ts">
import { useForm } from '@inertiajs/vue3'

const form = useForm({ email: '', password: '', avatar: null as
File | null })
const submit = () => form.post('/login', {
  preserveScroll: true,
  onSuccess: () => form.reset('password'),
})
</script>

<template>
  <form @submit.prevent="submit">
    <input v-model="form.email" type="email">
    <p v-if="form.errors.email">{{ form.errors.email }}</p>
    <progress v-if="form.progress"
:value="form.progress.percentage" max="100" />
    <button :disabled="form.processing">Log in</button>
    <span v-if="form.recentlySuccessful">Saved.</span>
  </form>
</template>
```

### Helpers

```
form.put(url); form.patch(url); form.delete(url);
form.submit('post', url)
form.transform(data => ({ ...data, remember: true
})).post('/login')
form.reset(); form.reset('field'); form.defaults();
form.cancel()
form.clearErrors('email'); form.setError('email', 'Taken')
form.isDirty; form.hasErrors; form.wasSuccessful
useForm('EditUser:' + user.id, data) // remembered in history
state
```

## Links and manual visits

### <Link>

```
<Link href="/users">Users</Link>
<Link href="/logout" method="post" as="button">Log out</Link>
<Link href="/search" :data="{ query }" preserve-state preserve-
scroll>Search</Link>
<Link href="/users" prefetch>Users</Link>
<Link href="/users" :class="{ active:
$page.url.startsWith('/users') }">Users</Link>
```

### router

```
import { router } from '@inertiajs/vue3'
router.visit('/users', { method: 'get', preserveState: true })
router.post('/users', data, { onHttpException: () => false })
// stay on page
router.cancelAll()
// v3: replaces cancel()
```

## Partial reloads

### Ask for some props only

```
router.reload({ only: ['users'] })
router.reload({ except: ['companies'] })
router.reload({ only: ['auth.notifications'] }) // dot
notation in v3
<Link href="/users?active=true" :only="['users']">Active</Link>
```

*Wrap expensive props in closures (or Inertia::optional) so a partial reload skips them.*

## Shared data

### HandleInertiaRequests middleware

```
public function share(Request $request): array
{
    return [
        ...parent::share($request),
        'appName' => config('app.name'),
        'auth.user' => fn () => $request->user()?->only('id',
            'name', 'email'),
    ];
}

public function shareOnce(Request $request): array
{
    return [...parent::shareOnce($request), 'countries' => fn
        () => Country::all()];
}
```

### Read it in Vue

```
import { usePage } from '@inertiajs/vue3'
const page = usePage()
page.props.auth.user // or $page.props in templates
```

## Validation errors

### Server validates, client reads errors

```
$request->validate(['email' => ['required', 'email']]); //
redirects back with errors
// Inertia puts them in props.errors (first message per field
by default)
protected $withAllErrors = true; // in HandleInertiaRequests:
all messages
```

Several forms on one page? Name an error bag: `form.post('/companies', { errorBag: 'createCompany' })`, then read `form.errors` as usual.

## SSR

### With the @inertiajs/vite plugin

```
// vite.config.js
plugins: [laravel({ input: ['resources/js/app.ts'], refresh:
true }), inertia()]
```

```
// package.json: "build": "vite build && vite build --ssr"
// dev: npm run dev (SSR runs in Vite, no Node server)
// prod: npm run build && php artisan inertia:start-ssr
```

Turn SSR off for tests with `Inertia::disableSsr(app())→runningUnitTests()`. Head tags use `data-inertia` (was `inertia` in v2).

Checked 2026-10-15 against: <https://inertiajs.com/docs/v3/getting-started/upgrade-guide> · <https://inertiajs.com/docs/v3/data-props/deferred-props> ·

<https://inertiajs.com/docs/v3/the-basics/forms> · <https://inertiajs.com/docs/v3/advanced/server-side-rendering>

Online version with copy buttons: [laravel-vuejs.space/cheatsheets/inertia](https://laravel-vuejs.space/cheatsheets/inertia) · More guides and the monthly newsletter at [laravel-vuejs.space](https://laravel-vuejs.space)