#Follow private channel docs #1 first

**##listening to multiple event**

Step 1: Making a event first, **php artisan make:event OrderDelivered**,

<?php  
  
namespace App\Events;  
  
use App\Models\Order;  
use App\Models\User;  
use Illuminate\Broadcasting\Channel;  
use Illuminate\Broadcasting\InteractsWithSockets;  
use Illuminate\Broadcasting\PresenceChannel;  
use Illuminate\Broadcasting\PrivateChannel;  
use Illuminate\Contracts\Broadcasting\ShouldBroadcast;  
use Illuminate\Contracts\Broadcasting\ShouldBroadcastNow;  
use Illuminate\Foundation\Events\Dispatchable;  
use Illuminate\Queue\SerializesModels;  
  
class OrderDelivered implements ShouldBroadcastNow  
{  
 use Dispatchable, InteractsWithSockets, SerializesModels;  
  
 */\*\*  
 \* Create a new event instance.  
 \*/* public function \_\_construct(public User $user,public Order $order)  
 {  
 //  
 }  
  
 */\*\*  
 \* Get the channels the event should broadcast on.  
 \*  
 \* @return array<int, \Illuminate\Broadcasting\Channel>  
 \*/* public function broadcastOn(): array  
 {  
 return [  
 new PrivateChannel('users.'.$this->user->id),  
 ];  
 }  
}

Step 2: From channels.php,

Broadcast::channel('users.{id}', function (User $user, $id) {  
 return (int) $user->id === (int) $id;  
});

Step 3: From **web.php**,

Route::get('/broadcast', function () {  
 sleep(3);  
 broadcast(new \App\Events\OrderDispatched(User::*find*(1), \App\Models\Order::*find*(1)));  
 sleep(5);  
 broadcast(new \App\Events\OrderDelivered(User::*find*(1), \App\Models\Order::*find*(1)));  
  
});

* Ekhane sleep er kaj ta hocche kotokkhon poe route ta execute hobe,

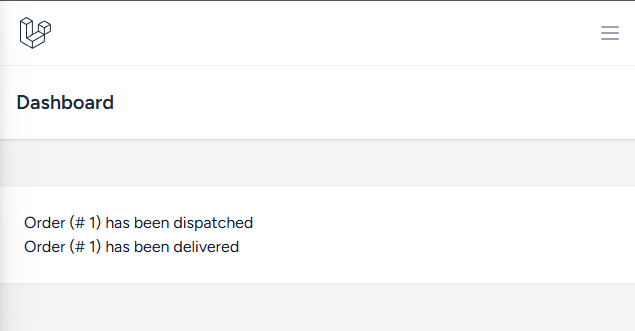
Step 4: from dashboard.php, **.listen** function ke chain kora jay,

<script>  
 function privateClass() {  
 Echo.private('users.{{ auth()->id() }}').listen('OrderDispatched', (event) => {  
  
 Alpine.store('dispatchedStore').dispatched = true;  
 Alpine.store('dispatchedStore').order = event.order  
 })  
  
 .listen('OrderDelivered', (event) => {  
  
 Alpine.store('dispatchedStore').delivered = true;  
 Alpine.store('dispatchedStore').order = event.order  
 })  
 }  
</script>

Full **dashboard.php** code,

<x-app-layout>  
 <x-slot name="header">  
 <h2 class="font-semibold text-xl text-gray-800 leading-tight">  
 {{ \_\_('Dashboard') }}  
 </h2>  
 </x-slot>  
  
  
 <div class="py-12">  
 <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">  
 <div class="bg-white overflow-hidden shadow-sm sm:rounded-lg">  
 <div class="p-6 text-gray-900" x-init="privateClass()">  
  
 <template x-if="$store.dispatchedStore.dispatched">  
 <div>  
 Order (# <span x-text="$store.dispatchedStore.order.id"></span>) has been dispatched  
 </div>  
 </template>  
  
 <template x-if="$store.dispatchedStore.delivered">  
 <div>  
 Order (# <span x-text="$store.dispatchedStore.order.id"></span>) has been delivered  
 </div>  
 </template>  
 </div>  
 </div>  
 </div>  
 </div>  
  
  
 <script>  
 function privateClass() {  
 Echo.private('users.{{ auth()->id() }}').listen('OrderDispatched', (event) => {  
  
 Alpine.store('dispatchedStore').dispatched = true;  
 Alpine.store('dispatchedStore').order = event.order  
 })  
  
 .listen('OrderDelivered', (event) => {  
  
 Alpine.store('dispatchedStore').delivered = true;  
 Alpine.store('dispatchedStore').order = event.order  
 })  
 }  
 </script>  
  
 {{-- Alpine js store script --}}  
  
 <script>  
 *document*.addEventListener('alpine:init', () => {  
 Alpine.store('dispatchedStore', {dispatched: false, delivered: false, order: null});  
 });  
 </script>  
  
  
</x-app-layout>

After hitting the “**/broadcast**” route, this output will occur,



##Note : Ekta event er moddhe multiple channel thaka possible,

public function broadcastOn(): array  
{  
 return [  
 new PrivateChannel('users.'.$this->order->user\_id),  
 new PrivateChannel('orders.'. $this->order->id),  
 ];  
}