Inertia.js

Introduction to Inertia.js: Inertia.js is a javascript package designed for developing single-page applications (SPAs) without the need for building a separate API. It enables the creation of client-side rendered apps using popular front-end frameworks such as Vue.js, React, or Svelte, while leveraging existing server-side frameworks like Laravel or other backend frameworks. In contrast to traditional approaches, Inertia.js follows a server-driven model, referring to it as the "modern monolith."

Unlike methods involving client-side routing and API development, Inertia simplifies the process by allowing developers to write controllers and page views, eliminating the necessity for a separate API. This approach enables seamless rendering the code of the backend framework directly on the client-side, reducing development time and providing an efficient solution for building SPAs.

Comparison of SSR and CSR: Server-side rendering (SSR) and client-side rendering (CSR) represent two distinct approaches to rendering web applications. SSR involves generating the HTML content of a page on the server, while CSR defers this task to the client-side JavaScript. Server-side rendering pre-renders JavaScript pages on the server, allowing the visitors to receive fully rendered HTML when they visit the application. Overall, SSR is a valuable technique that can help build a more efficient and effective web presence.

Server-side rendering (SSR) has several advantages in web development.

SSR allows for faster load times, as the server generates the HTML for a web page and sends it to the client side to display on the browser. This results in a smoother user experience and avoids the blank page flicker typical of client-side rendering (CSR).
SSR provides better SEO performance, as search engines can easily crawl and index the content of the website. This can result in higher search engine rankings and more organic traffic.
SSR is efficient for loading on slow internet connections and can handle large-scale applications without affecting performance.

SSR has also some disadvantages.

While there are some disadvantages to server-side rendering (SSR) Such as increased expenses and higher complexity in development and hosting. Server-Side Rendering (SSR) offers several benefits, including enhanced page loading times by pre-rendering pages on the server, resulting in faster rendering on the client side. This not only improves user experience but also boosts search engine rankings. SSR facilitates better search engine crawling and indexing since it provides fully formed HTML pages for search engines to analyze. Additionally, SSR enhances the mobile experience by minimizing the JavaScript needed for page rendering, ensuring quick loading on mobile devices. Furthermore, SSR contributes to better accessibility by making web content available to users with disabilities, irrespective of their device or browser.

In terms of CSR; it is a technique where all the page resources are rendered on the client's browser rather than the server. This is done using the Javascript framework that compiles and generates code at the browser end. CSR can be incorporated with the help of Javascript frameworks and libraries like Angular, VueJS, and React SSR.

The CSR approach has many advantages like:

Renders fast – The page renders quickly after the initial page load time.
Offers quicker navigation of the website – This is possible because placeholders are loaded first.
Puts less load on server – The Javascript is executed on the client's browser, putting less load on the server.
Works amazing with PWA (Progressive Web Apps) – Again, this is because the code can be entirely rendered on the browser.
Is interactive – The rendered page is interactive.

The CSR approach also has some disadvantages:

Loads slowly – The HTML, CSS and Javascript have to be rendered first and then displayed to the user, increasing the time taken to load the initial page.
Hampers SEO if not implemented correctly – This can increase your page speed and can be unresponsive for the user.
Delays crawling and indexing– The search engine bot has to wait for all the page resources to load.
Relies on external libraries – To achieve the best results, it has to be combined with external libraries and frameworks.

Inertia.js Features:

Inertia.js orchestrates a harmonious blend of features that enhance the development experience and optimize application performance.

Data-driven UI: Inertia.js facilitates the creation of data-driven UIs by seamlessly passing data from the server to the client-side components.

Client-side routing: Inertia.js enables client-side routing without the complexity of managing an API. It utilizes single-page components and server-side routing to provide a seamless navigation experience.

Shared controllers: Inertia.js promotes code reuse by allowing the sharing of controllers between server-side routes and client-side components.

Integration with Laravel:

To use Inertia.js with Laravel framework following steps are done.

Installing the Inertia server-side adapter using the Composer package manager.

```
composer require inertiajs/inertia-laravel
```

Next, setup the root template that will be loaded on the first page visit to your application.

```
<html>
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0" />
@vite('resources/js/app.js')
@inertiaHead
</head>
<body>
      @inertia
</body>
</html>
```

Next we need to setup the Inertia middleware with the following command.

```
php artisan inertia:middleware
```

Once the middleware has been published, we have to register the `HandleInertiaRequests` middleware in our `App\Http\Kernel` as the last item in the **web** middleware group.

```
'web' => [
// ...
\App\Http\Middleware\HandleInertiaRequests::class,
 ],
```

Now you're ready to start creating Inertia pages and rendering them via responses.

```
use Inertia\Inertia;
class EventsController extends Controller
{
      public function show(Event $event)
      {
            return Inertia::render('Event/Show', [
                  'event' => $event->only(
                        'id',
                        'title',
                        'start_date',
                        'description'
                  ),
            ]);
      }
}
```

Client-Side Components: Vue.js, a popular JavaScript framework, pairs beautifully with Inertia.js for creating dynamic and interactive client-side components. It is a progressive JavaScript framework for building user interfaces. Client-side components are the fundamental building blocks of Vue.js applications. They are responsible for rendering HTML markup and handling user interactions. Components can be nested within each other to create complex UIs. Inertia.js complements frontend frameworks like Vue.js to create dynamic client-side components. These components enhance the user interface by enabling interactive elements without compromising on performance. The communication between the server and client is facilitated through the exchange of data, ensuring that the application remains reactive and responsive.