



Client-Side vs. Server-Side Rendering

In this lesson, you will learn about the client side and the server-side rendering and the use cases for both the approaches.

We'll cover the following



- Client-side rendering - How does a browser render a web page?
- Server-side rendering
- Use cases for server-side & client-side rendering

Client-side rendering - How does a browser render a web page?#

When a user requests a web page from the server and the browser receives the response, it has to render the response on the window in the form of an HTML page.

For this, the browser has several components, such as the:

- *Browser engine*
- *Rendering engine*
- *JavaScript interpreter*
- *Networking and the UI backend*

- *Data storage etc.*



I won't go into much detail but the browser has to do a lot of work to convert the response from the server into an HTML page.

The rendering engine constructs the *DOM* tree and renders and paints the construction. Naturally, all this activity needs a bit of time.

Server-side rendering#

To avoid all this rendering time on the client, developers often render the UI on the server, generate HTML there and directly send the HTML page to the UI.

This technique is known as the *server-side rendering*. It ensures faster rendering of the UI, averting the UI loading time in the browser window because the page is already created and the browser doesn't have to do much assembling or rendering work.

Use cases for server-side & client-side rendering#

The server-side rendering approach is perfect for delivering static content, such as WordPress blogs. It's also good for SEO because the crawlers can easily read the generated content.

However, modern websites are highly dependent on AJAX. On such websites, content for a particular module or a section of a page has to be fetched and rendered on the fly.

Therefore, server-side rendering doesn't help much. For every AJAX-

request, instead of sending just the required content to the client, the approach generates the entire page on the server. This process consumes

unnecessary bandwidth and also fails to provide a smooth user experience.

A big downside to this is that, once the number of concurrent users on the website rises, it puts an unnecessary load on the server.

Client-side rendering works best for modern dynamic AJAX-based websites.

However, we can leverage a hybrid approach, to get the most out of both techniques. We can use server-side rendering for the home page and for the other static content on our website and use client-side rendering for the dynamic pages.

Alright, before moving down to the database, message queue and the caching components, it's important for us to understand a few concepts such as:


- *Monolithic architecture*
- *Micro-services*
- *Scalability*
- *High availability*
- *Distributed systems*
- *What are nodes in distributed systems? Why are they important to software design?*

Understanding these concepts will help you understand the rest of the web components better. Let's have a look one by one.

[← Back](#)[Next →](#)


Mark as


Completed

 Report an Issue