# SSR

Server Side Rendering SSR

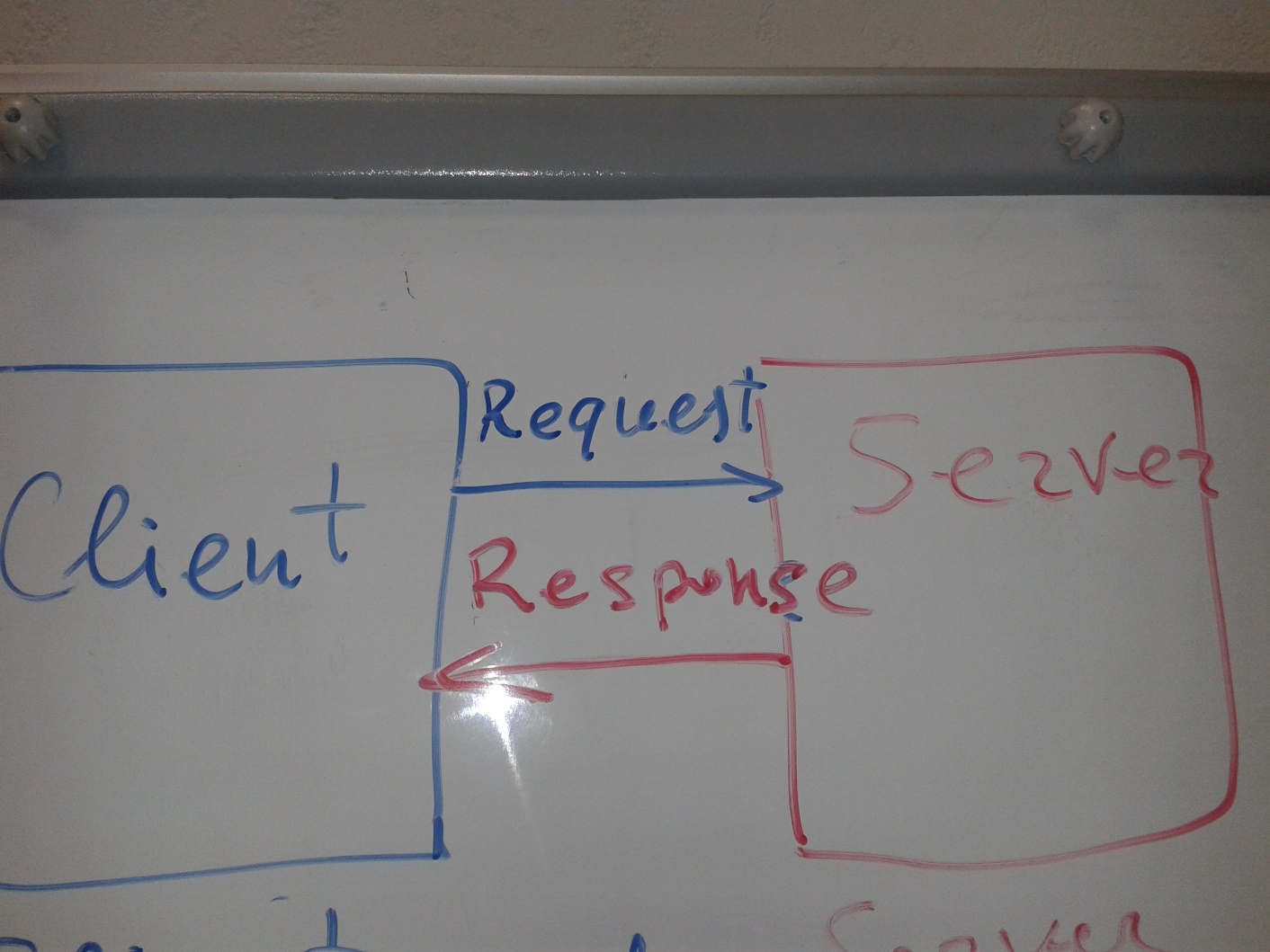
The client–server model is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients.

So, we have client, server and connection between client and server.

Historical first we had Server side rendering, but client became thicker and thicker and the connection was the thickest place in this architecture.

The client side rendering was the most popular and the most performance.

But client side is not good approach for weakness browsers.

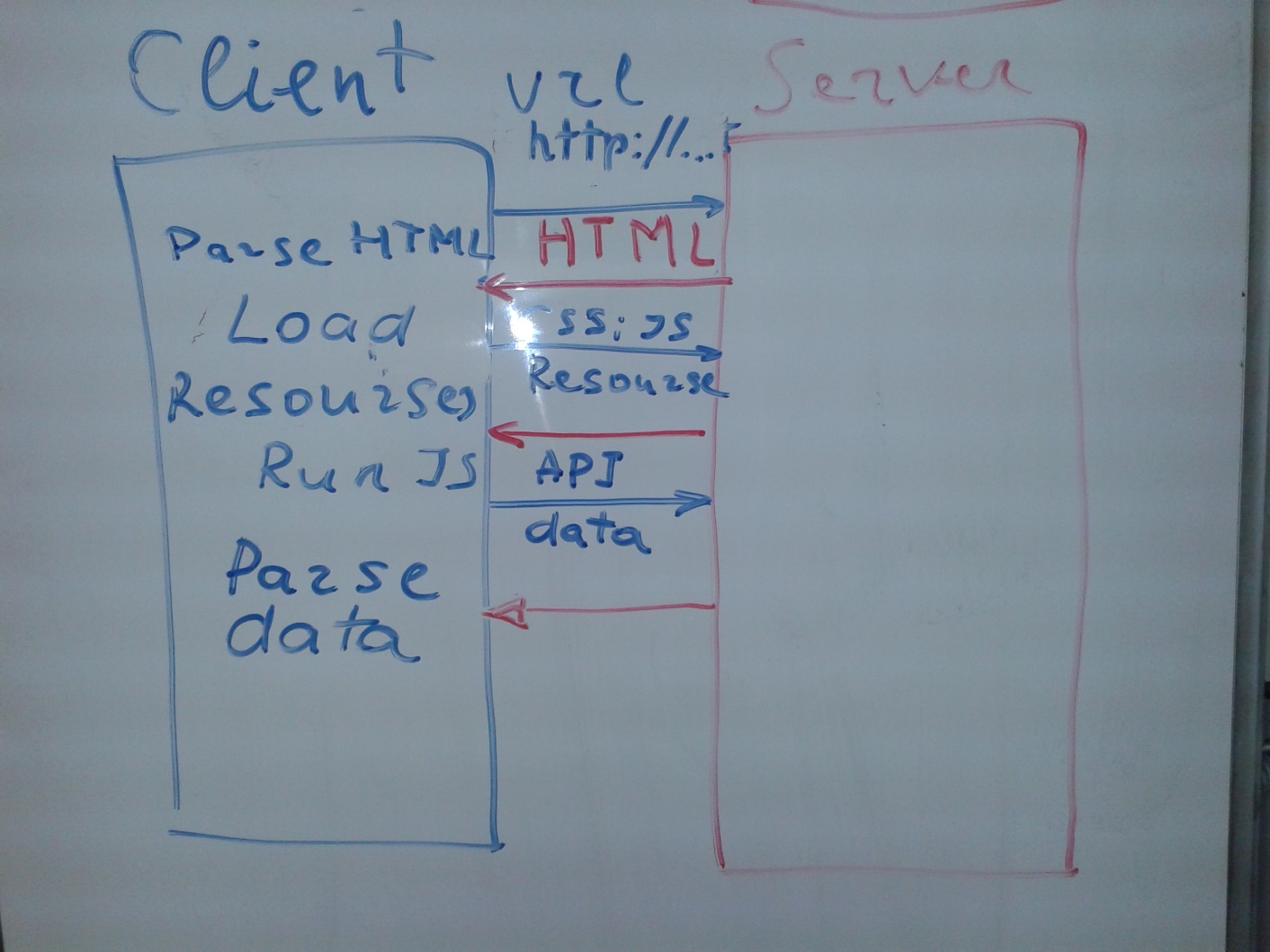


Which one is better client side or server side rendering

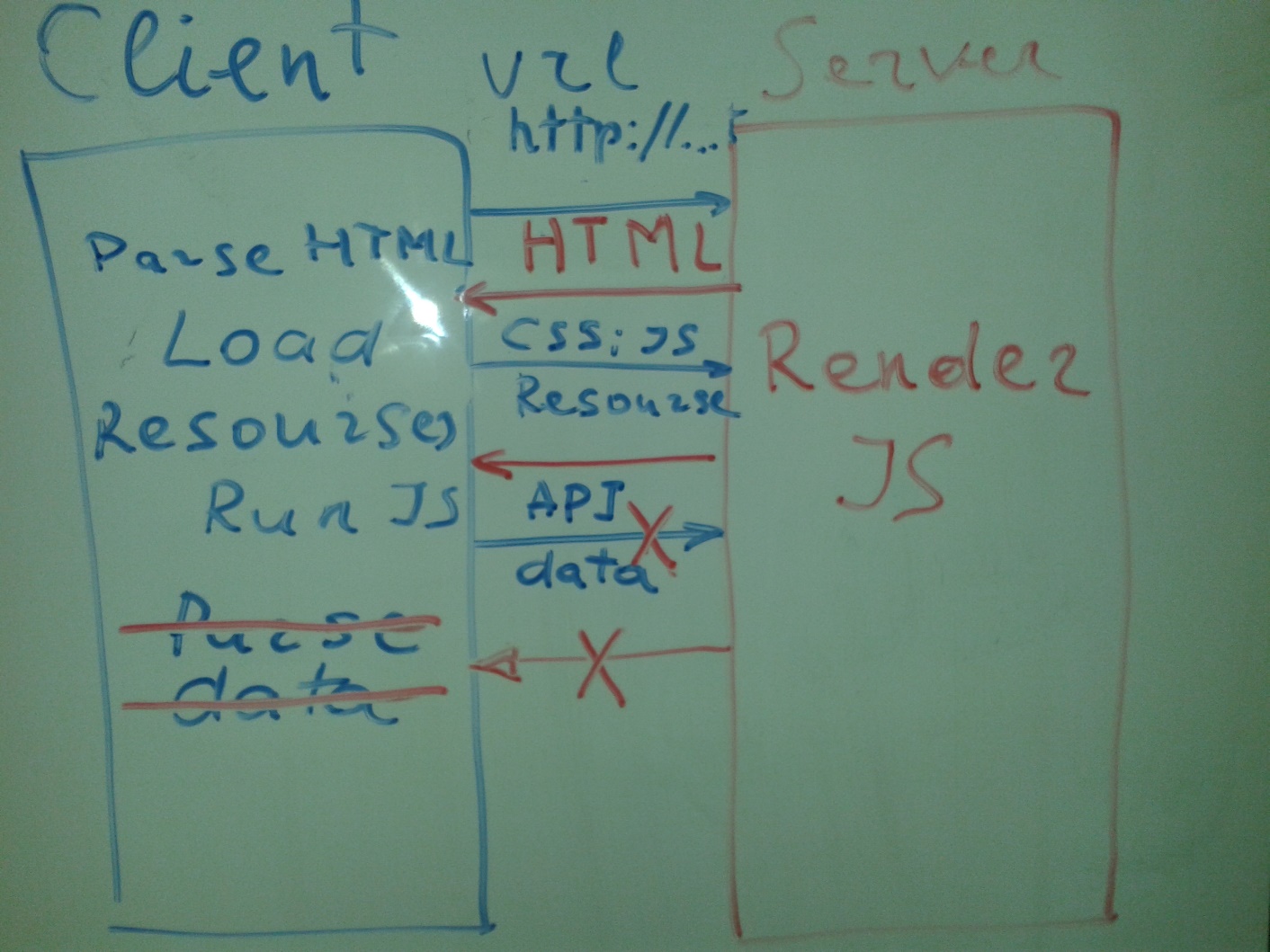
With client-side rendering, your initial request loads the page layout, CSS and JavaScript. It's all common except that some or all of the content isn't included. Instead, the JavaScript makes another request, gets a response (likely in JSON), and generates the appropriate HTML (likely using a templating library).

With server-side rendering, your initial request loads the page, layout, CSS, JavaScript and content.

Client side rendering:



Server side rendering:



Server-side vs client-side rendering

In the old days almost all rendering was done server-side, either:

• dynamically on the fly when a request is received (in PHP, ASP, Perl, Java,...),

• at scheduled update times,

• or whenever a modification is done on the server.

Since JavaScript and DOM are available on the client, it has been made possible to modify the rendering on client-side, and Ajax made it even easier.

Server-side rendering is very much recommended for content oriented website to be easily parsable by search engines. It can also be more mobile friendly at the computing level and then potentially preserve the battery, it has a cost at the communication level as all refresh need to reload a lot of HTML.

With client-side rendering, it becomes possible to make almost all the HTML layout cacheable, as usually are the scripts and CSS files. The client, at the extreme level will have to only load the HTML once, and all the data can be loaded in a lighter JSON format to be rendered client-side in JavaScript with libs like Mustache.js, Handlebar, Underscore, AngularJS, or almost any framework. It is more appropriate for Web Applications that doesn't need to be parsable by search engines. The Wakanda Ajax Framework use this approach at the extreme level for either application HTML "pages" or "components". This approach is often even more adapted when building hybrid mobile Web Apps as all these static HTML layouts can easily be integrated in the mobile application package.

Some solutions propose rendering on both sides. Usually, the client-side rendering is then the first used option, while a Google compliant alternative URL will provide the same content from server-side rendering. Such solution has become a bit easier to do consistently as with Server-Side JavaScript, the same templates and template engines may be used on both sides for an exact same result.

Server-side can also be used to provide alternative formats than HTML like PDF, Excel, zip archives, images. Most of these formats were hard if not impossible to generate on client-side in JavaScript, but since the new HTML5 features came in, it started to become possible.

The main questions today becomes then:

• Do I want my app to be as standalone as possible on client-side, even in offline mode?

• Or do I prefer to keep it light and defer some tasks to the server?

Your choice will probably be driven by:

• your server capabilities to handle your amount of connected clients

• the capacity / power of your target clients

• the expected network speed

Modern Smartphones and tablets are more and more powerful and the bottleneck is often at the network level. In this case client-side rendering can be great (even more for offline mode)

Some users are still limited to lighter phones in which case server-side rendering can help. Advanced apps can also detect the network speed and choose server-side rendering when it is big enough compared to the cost of the client-side rendering (like when generating dynamic images).

There are three examples which show Server Side Rendering itself, SSR in React-Router, SSR in React-Redux. Each example has read.me how to run tis example.

Happy coding! :)