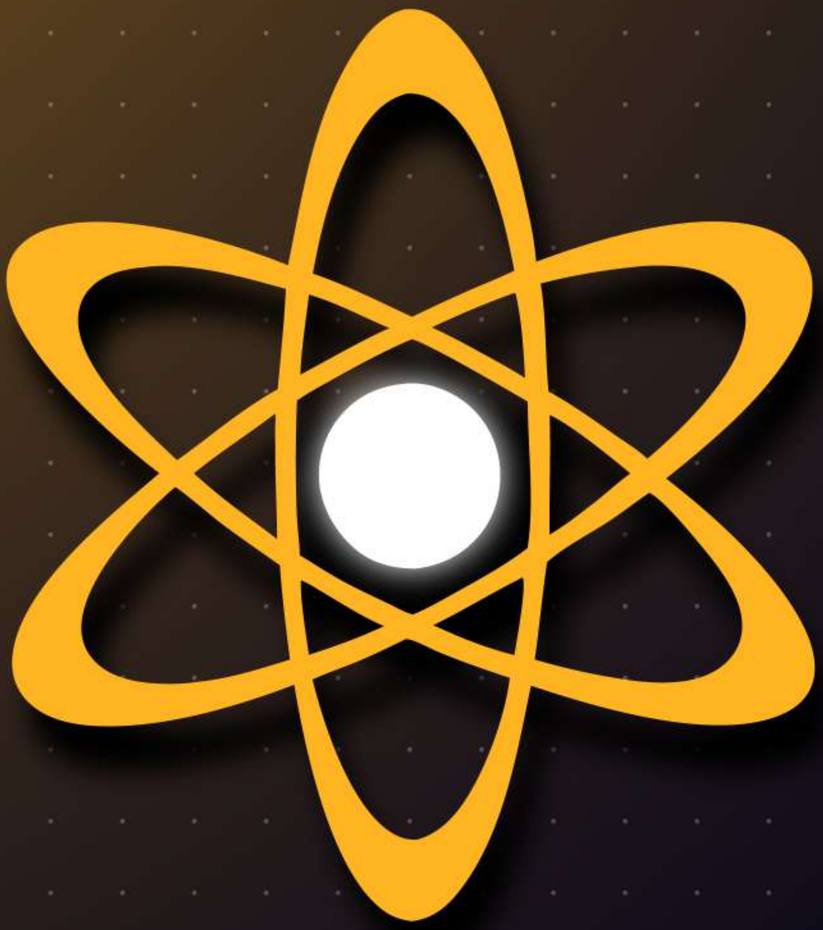
Speed Up Your App: Optimize with HTTP/2 in React & Next.js



SUMANTH M

Frontend developer

1

What is HTTP/2?

- Definition: HTTP/2 is the second major version of the HTTP protocol that powers the web. It improves the way data is sent between the client (browser) and server.
- Importance: HTTP/2 enhances performance by reducing latency, improving page load speeds, and making web applications more efficient, especially with large or complex resources.

SUMANTH M
Frontend developer

Why is HTTP/2 Important for Web Apps?

- Reduced Latency: HTTP/2 reduces latency by allowing multiple requests and responses to be sent over a single connection, eliminating the need for multiple round trips.
- Smarter Resource Loading: It optimizes the loading of assets like JavaScript, CSS, and images, which can speed up page rendering significantly.
- Improved Multiplexing: HTTP/2 can handle multiple r equests in parallel, reducing the need for connection multiplexing and minimizing delays in loading resources.

SUMANTH M
Frontend developer

1. Multiplexing: Faster Resource Loading

- What It Is: HTTP/2 allows multiple resources to be requested and delivered in parallel over a single connection, without blocking each other.
- Why It Matters: This reduces the time spent establishing connections and eliminates delays caused by the traditional serial request-response model in HTTP/1.1.
- How to Optimize: HTTP/2 handles multiplexing automatically, so you don't need to worry about requesting assets in parallel. Just make sure your server supports it.

Instead of making separate HTTP requests for each asset, HTTP/2 sends them all at once in a single connection, speeding up asset delivery.

SUMANTH M
Frontend developer

2. Server Push: Preload Resources Efficiently

- What It Is: Server Push is an HTTP/2 feature that allows the server to send resources to the client before they are requested.
- Why It Matters: By sending resources proactively, you can ensure that essential assets (like JavaScript, CSS, or fonts) are available immediately, reducing load times.
- How to Optimize: Use server push to send critical resources like stylesheets, scripts, and images to the browser without waiting for the client to request them.

```
// Example of pushing a CSS file in an HTTP/2 compatible server
res.setHeader('Link', '</styles.css>; rel=preload; as=style');
```

SUMANTH M
Frontend developer

5

React Js

3. Header Compression: Reduce Overhead

- What It Is: HTTP/2 uses HPACK compression for header fields, reducing the size of request and response headers sent over the network.
- Why It Matters: Header compression reduces unnecessary overhead, speeding up the process of sending requests and responses.
- How to Optimize: Enable HTTP/2 support on your server to take advantage of header compression automatically.

When a browser repeatedly requests the same headers (like cookies or authorization tokens), HTTP/2 compresses these headers to reduce data size.

SUMANTH M
Frontend developer

4. Streamlining JavaScript and CSS Delivery

What It Is: HTTP/2 enables more efficient delivery of JavaScript and CSS files, which often make up the majority of page load times in modern web apps.

Why It Matters: By allowing multiple assets to be delivered at once and reducing the overhead of opening new connections, HTTP/2 significantly improves the delivery of these critical files.

How to Optimize: Ensure your server and client use HTTP/2 to deliver JavaScript and CSS files more efficiently, enabling faster page rendering and responsiveness.

In a React or Next.js app, HTTP/2 can be used to preload critical assets such as JavaScript bundles and CSS files to minimize delays during initial page rendering.

5. Use HTTP/2 for Faster API Calls

What It Is: HTTP/2's ability to handle multiple requests concurrently can also speed up API calls, which are common in React and Next.js apps.

Why It Matters: If your app makes many API requests (e.g., for dynamic content), using HTTP/2 can drastically reduce the latency and time spent waiting for responses.

How to Optimize: Ensure API endpoints support HTTP/2 to reduce the number of connections and improve the responsiveness of your backend interactions.

```
// When making API calls in your React app, HTTP/2 will multiplex all calls
const response = await fetch('/api/data');
```

6. Server-Side Optimization for HTTP/2

What It Is: To benefit from HTTP/2, your server must be configured to support it.
Why It Matters: Servers like Nginx, Apache, and others need to be set up to serve HTTP/2 to clients. This allows the full potential of multiplexing, server push, and header compression.
How to Optimize: Ensure your hosting environment or server supports HTTP/2 and that it's enabled to take advantage of its benefits

```
server {
  listen 443 ssl http2;
  ...
}
```

SUMANTH M
Frontend developer

7. Compatibility and Fallbacks

What It Is: HTTP/2 is widely supported in modern browsers, but some older browsers or clients may not support it.

Why It Matters: Ensuring that your app works well for all users means that you need to have a fallback for HTTP/1.1 when HTTP/2 isn't available.

How to Optimize: Most modern web servers handle this automatically, falling back to HTTP/1.1 if HTTP/2 isn't supported. Ensure that your server is configured to gracefully handle such cases.

In your server setup, HTTP/2 is prioritized, but if the browser doesn't support it, HTTP/1.1 is used as a fallback.

SUMANTH M
Frontend developer

Conclusion

Enabling HTTP/2 on your React and Next.js applications is a powerful way to optimize performance, reduce latency, and improve the user experience.
By leveraging multiplexing, server push, header compression, and more, you can make your web apps faster, more efficient, and better suited for high-performance scenarios.

SUMANTH M
Frontend developer