**What is HTTP?**

Hypertext Transfer Protocol, commonly known as HTTP, serves as the foundation for nearly all web applications. In essence, it functions as the communication protocol between computers and servers, facilitating the request and transmission of information.

**What is HTTP/1.1?**

HTTP, or Hypertext Transfer Protocol, saw its inaugural functional iteration emerge in 1997. Evolving through multiple developmental phases, this inaugural rendition earned the moniker HTTP/1.1. Remarkably, this particular version continues to be actively employed across the web to this day.

**What is HTTP/2?**

Back in 2015, a fresh iteration of HTTP emerged, known as HTTP/2. This newer version addressed various challenges that were unforeseen by the architects of HTTP/1.1. Notably, HTTP/2 boasts significantly enhanced speed and efficiency compared to its predecessor. One of the key factors contributing to this swifter performance lies in its innovative approach to prioritizing content during the loading process.

**What is prioritization?**

Within the realm of web performance, prioritization delineates the sequence in which various content elements are loaded. Consider a user exploring a news website and navigating to an article. Questions arise: Should the photo atop the article take precedence in loading? Or should the text of the article or banner ads be prioritized first?

The process of prioritization wields a notable impact on a webpage's load time. For instance, certain resources, like sizable JavaScript files, may impede the loading of the entire page if they are required to load first. Optimal loading efficiency occurs when these render-blocking resources are deferred to load last, enabling more concurrent loading of the rest of the page.

Moreover, the sequence in which these page resources load influences the user's perception of page load time. If inconspicuous content (such as a CSS file) or content not immediately visible to the user (like banner ads at the page's bottom) loads initially, the user might perceive a delay or assume the page isn't loading at all. On the contrary, when pivotal content, such as the image at the page's zenith, loads first, the user experiences a swifter perception of the page loading.

**How does prioritization in HTTP/2 affect performance?**

Within the framework of HTTP/2, developers wield hands-on, intricate control over prioritization, presenting an unprecedented ability to optimize both perceived and actual page load speed, a feat unattainable in the HTTP/1.1 era.

A distinctive feature of HTTP/2 is its provision of weighted prioritization. This empowers developers to dictate the loading sequence of page resources consistently. When a client initiates a request for a webpage in HTTP/2, the server dispatches multiple data streams simultaneously to the client, departing from the sequential approach of HTTP/1.1. This simultaneous data delivery method is known as multiplexing. Developers assign distinct weighted values to each data stream, guiding the client on the priority order for rendering.

To illustrate, consider Alice and Bob communicating via traditional mail. In an HTTP/1.1 scenario, Bob sends one chapter of his novel at a time, awaiting confirmation from Alice before dispatching the next. This elongated process extends the time it takes for Alice to read the entire novel over many weeks. However, in the HTTP/2 model, Bob adopts a more efficient strategy. He sends each chapter separately but concurrently, numbering them for reference. Alice receives the entire novel at once, assembling it in the correct order at her convenience. If a chapter is missing, she can swiftly request it, expediting the overall reading process to just a few days.

Similarly, in HTTP/2, data is transmitted simultaneously, analogous to Bob sending multiple novel chapters concurrently. Developers, akin to numbering chapters, exercise the authority to decide the loading order of webpage elements—be it the text, CSS files, JavaScript, or any components deemed pivotal for an enhanced user experience.

**What are the other differences between HTTP/2 and HTTP/1.1 that impact performance?**

Multiplexing, a stark departure from HTTP/1.1's sequential loading of resources, is a hallmark of HTTP/2. While HTTP/1.1 loads resources one after the other, causing a blockade if one resource encounters loading issues, HTTP/2 capitalizes on a single TCP connection to simultaneously dispatch multiple streams of data. This method ensures that the failure to load one resource does not impede the loading of other resources. HTTP/2 achieves this by segmenting data into binary-code messages and assigning numerical values to each message, enabling the client to identify the respective stream to which each binary message belongs.

Another innovation in HTTP/2 is the concept of server push. Traditionally, a server provides content to a client only upon request. However, with modern webpages often entailing numerous resources, the client must request each one individually. HTTP/2 addresses this challenge by enabling servers to "push" content to clients preemptively, even before the client requests it. In this scenario, the server also communicates a message to the client detailing the pushed content to expect, analogous to Bob sending Alice a Table of Contents before delivering his entire novel.

Furthermore, header compression plays a pivotal role in enhancing web performance. Recognizing that smaller files load more rapidly than larger ones, both HTTP/1.1 and HTTP/2 employ compression techniques to reduce the size of HTTP messages. However, HTTP/2 introduces a more sophisticated compression method named HPACK. This method excels at eliminating redundant information within HTTP header packets, shaving off a few bytes from each packet. Given the substantial volume of HTTP packets involved in loading a single webpage, these byte savings accumulate swiftly, resulting in expedited loading times.