**HTTP messages**

Mensagens HTTP são o mecanismo usado para trocar dados entre um servidor e um cliente no protocolo HTTP. Existem dois tipos de mensagens: solicitações enviadas pelo cliente para acionar uma ação no servidor e respostas, a resposta que o servidor envia em resposta a uma solicitação.

Desenvolvedores raramente, ou nunca, criam mensagens HTTP do zero. Aplicativos como navegadores, proxies ou servidores web utilizam softwares projetados para criar mensagens HTTP de forma confiável e eficiente. A forma como as mensagens são criadas ou transformadas é controlada por APIs em navegadores, arquivos de configuração para proxies ou servidores, ou outras interfaces.

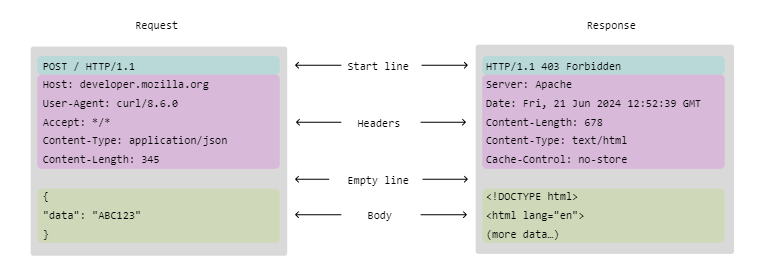
Nas versões do protocolo HTTP até o HTTP/2, as mensagens são baseadas em texto e são relativamente fáceis de ler e entender depois que você se familiarizar com o formato. No HTTP/2, as mensagens são encapsuladas em quadros binários, o que as torna um pouco mais difíceis de ler sem certas ferramentas. No entanto, a semântica subjacente do protocolo é a mesma, então você pode aprender a estrutura e o significado das mensagens HTTP com base no formato baseado em texto das mensagens HTTP/1.x e aplicar esse conhecimento ao HTTP/2 e versões posteriores.

Este guia utiliza mensagens HTTP/1.1 para facilitar a leitura e explica a estrutura das mensagens HTTP usando o formato HTTP/1.1. Na seção final, destacamos algumas diferenças que você pode precisar para descrever o HTTP/2.

Observação: você pode ver mensagens HTTP na guia Rede do navegador nas ferramentas do desenvolvedor ou se imprimir mensagens HTTP no console usando ferramentas CLI, como curl, por exemplo.

**Anatomia de uma mensagem HTTP**

Para entender como as mensagens HTTP funcionam, vamos analisar as mensagens HTTP/1.1 e analisar sua estrutura. A ilustração a seguir mostra a aparência das mensagens em HTTP/1.1:



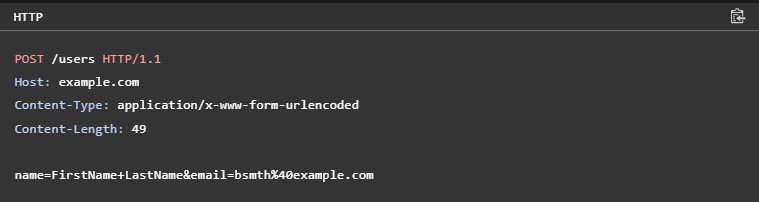
Tanto solicitações quanto respostas compartilham uma estrutura semelhante:

1. Uma linha inicial é uma única linha que descreve a versão HTTP, juntamente com o método de solicitação ou o resultado da solicitação.
2. Um conjunto opcional de cabeçalhos HTTP contendo metadados que descrevem a mensagem. Por exemplo, uma solicitação de um recurso pode incluir os formatos permitidos desse recurso, enquanto a resposta pode incluir cabeçalhos para indicar o formato real retornado.
3. Uma linha vazia indicando que os metadados da mensagem estão completos.
4. Um corpo opcional contendo dados associados à mensagem. Podem ser dados POST a serem enviados ao servidor em uma solicitação ou algum recurso retornado ao cliente em uma resposta. Se uma mensagem contém um corpo ou não é determinado pela linha inicial e pelos cabeçalhos HTTP.

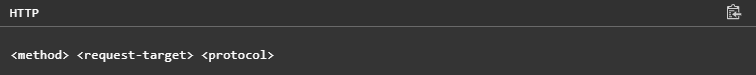
A linha inicial e os cabeçalhos da mensagem HTTP são conhecidos coletivamente como o cabeçalho das solicitações, e a parte posterior que contém seu conteúdo é conhecida como corpo.

**HTTP requests**

Vejamos o seguinte exemplo de solicitação HTTP POST enviada depois que um usuário envia um formulário em uma página da web:



A linha de início em solicitações HTTP/1.x (POST /users HTTP/1.1 no exemplo acima) é chamada de "linha de solicitação" e é composta de três partes:



<method>

O método HTTP (também conhecido como verbo HTTP) é um de um conjunto de palavras definidas que descrevem o significado da solicitação e o resultado desejado. Por exemplo, GET indica que o cliente gostaria de receber um recurso em troca, e POST significa que o cliente está enviando dados para um servidor.

<request-target>

O destino da solicitação geralmente é uma URL absoluta ou relativa e é caracterizado pelo contexto da solicitação. O formato do destino da solicitação depende do método HTTP utilizado e do contexto da solicitação. Ele é descrito com mais detalhes na seção Destinos da solicitação abaixo.

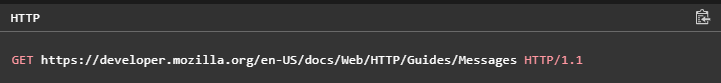
<protocol>

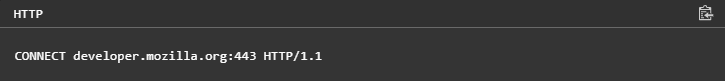
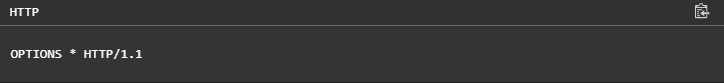
A versão HTTP, que define a estrutura da mensagem restante, atuando como um indicador da versão esperada a ser usada para a resposta. Quase sempre é HTTP/1.1, pois HTTP/0.9 e HTTP/1.0 estão obsoletos. No HTTP/2 e versões superiores, a versão do protocolo não é incluída nas mensagens, pois é compreendida a partir da configuração da conexão.

**Request targets**

Existem algumas maneiras de descrever um alvo de solicitação, mas de longe a mais comum é o "formato de origem". Aqui está uma lista dos tipos de alvos e quando eles são usados:

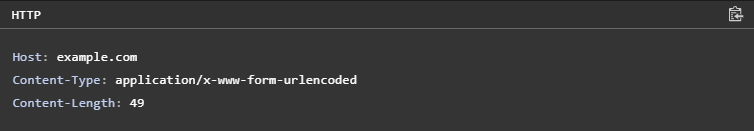
1. No formato de origem, o destinatário combina um caminho absoluto com as informações do cabeçalho do Host. Uma string de consulta pode ser anexada ao caminho para obter informações adicionais (geralmente no formato chave=valor). Isso é usado com os métodos GET, POST, HEAD e OPTIONS:
2. O formato absoluto é uma URL completa, incluindo a autoridade, e é usada com GET ao conectar-se a um proxy:



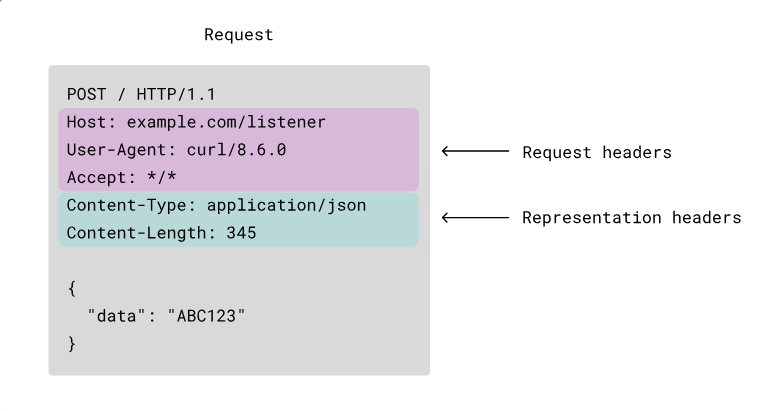
1. O formato de autoridade é a autoridade e a porta separadas por dois pontos (:). Ele é usado apenas com o método CONNECT ao configurar um túnel HTTP:
2. O formato asterisco só é usado com OPTIONS quando você deseja representar o servidor como um todo (\*) em oposição a um recurso nomeado:****

**Request headers**

Cabeçalhos são metadados enviados com uma solicitação após a linha inicial e antes do corpo. No exemplo de envio de formulário acima, eles são as seguintes linhas da mensagem:



No HTTP/1.x, cada cabeçalho é uma string que não diferencia maiúsculas de minúsculas, seguida por dois pontos (:) e um valor cujo formato depende do cabeçalho. O cabeçalho inteiro, incluindo o valor, consiste em uma única linha. Essa linha pode ser bem longa em alguns casos, como no cabeçalho Cookie.

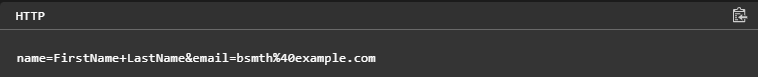


Alguns cabeçalhos são usados ​​exclusivamente em solicitações, enquanto outros podem ser enviados tanto em solicitações quanto em respostas, ou podem ter uma categorização mais específica:

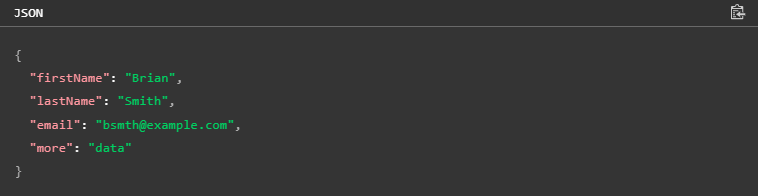
* Cabeçalhos de solicitação fornecem contexto adicional a uma solicitação ou adicionam lógica extra à forma como ela deve ser tratada por um servidor (por exemplo, solicitações condicionais).
* Cabeçalhos de representação são enviados em uma solicitação se a mensagem tiver um corpo e descrevem a forma original dos dados da mensagem e qualquer codificação aplicada. Isso permite que o destinatário entenda como reconstruir o recurso como ele era antes de ser transmitido pela rede.

**Request body**

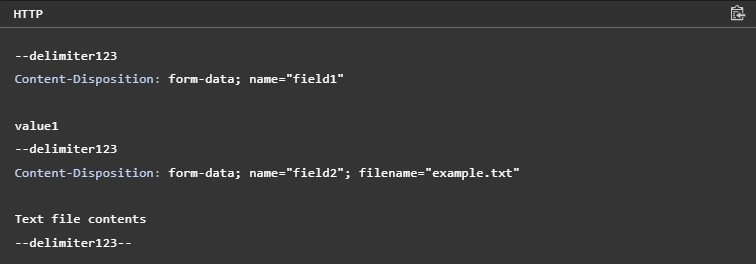
O corpo da solicitação é a parte da solicitação que transporta informações para o servidor. Somente solicitações PATCH, POST e PUT possuem um corpo. No exemplo de envio do formulário, esta parte é o corpo:



O corpo da solicitação de envio do formulário contém uma quantidade relativamente pequena de informações como pares chave=valor, mas o corpo da solicitação pode conter outros tipos de dados que o servidor espera:



ou dados em várias partes:

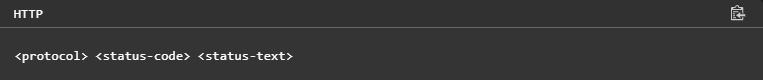


**HTTP responses**

Respostas são as mensagens HTTP que um servidor envia em resposta a uma solicitação. A resposta permite que o cliente saiba qual foi o resultado da solicitação. Aqui está um exemplo de resposta HTTP/1.1 para uma solicitação POST que criou um novo usuário:



A linha de início (HTTP/1.1 201 criada acima) é chamada de "linha de status" nas respostas e tem três partes:



<protocol>

A versão HTTP da mensagem restante.

<status-code>

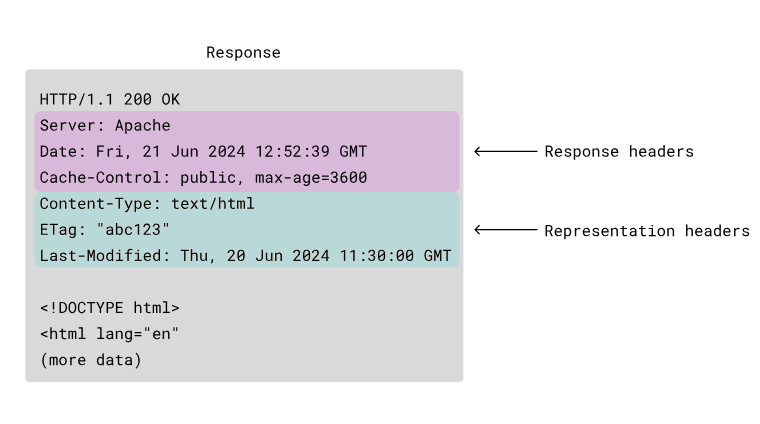
Um código de status numérico que indica se a solicitação foi bem-sucedida ou não. Os códigos de status comuns são 200, 404 ou 302.

<status-text>

O texto de status é uma descrição textual breve e puramente informativa do código de status para ajudar um humano a entender a mensagem HTTP.

**Response headers**

Cabeçalhos de resposta são os metadados enviados com uma resposta. Em HTTP/1.x, cada cabeçalho é uma string que não diferencia maiúsculas de minúsculas, seguida por dois pontos (:) e um valor cujo formato depende do cabeçalho usado.



Assim como os cabeçalhos de solicitação, há muitos cabeçalhos diferentes que podem aparecer nas respostas, e eles são categorizados como:

* Response headers que fornecem contexto adicional sobre a mensagem ou adicionam lógica extra à forma como o cliente deve fazer solicitações subsequentes. Por exemplo, cabeçalhos como "Server" incluem informações sobre o software do servidor, enquanto "Date" inclui quando a resposta foi gerada. Há também informações sobre o recurso retornado, como seu tipo de conteúdo (Content-Type) ou como ele deve ser armazenado em cache (Cache-Control).
* Representation headers: se a mensagem tiver um corpo, eles descrevem o formato dos dados da mensagem e qualquer codificação aplicada. Por exemplo, o mesmo recurso pode ser formatado em um tipo de mídia específico, como XML ou JSON, localizado para uma linguagem escrita ou região geográfica específica e/ou compactado ou codificado de outra forma para transmissão. Isso permite que o destinatário entenda como reconstruir o recurso como ele era antes de ser transmitido pela rede.

**Response body**

Um corpo de resposta é incluído na maioria das mensagens ao responder a um cliente. Em solicitações bem-sucedidas, o corpo da resposta contém os dados que o cliente solicitou em uma solicitação GET. Se houver problemas com a solicitação do cliente, é comum que o corpo da resposta descreva o motivo da falha e indique se a falha é permanente ou temporária.

Response bodies may be:

* Single-resource bodies definidos pelos dois cabeçalhos: Content-Type e Content-Length, ou de comprimento desconhecido e codificados em blocos com Transfer-Encoding definido como chunked.
* Multiple-resource bodies, consistindo em um corpo que contém várias partes, cada uma contendo uma informação diferente. Corpos multipartes são normalmente associados a formulários HTML, mas também podem ser enviados em resposta a solicitações de intervalo.

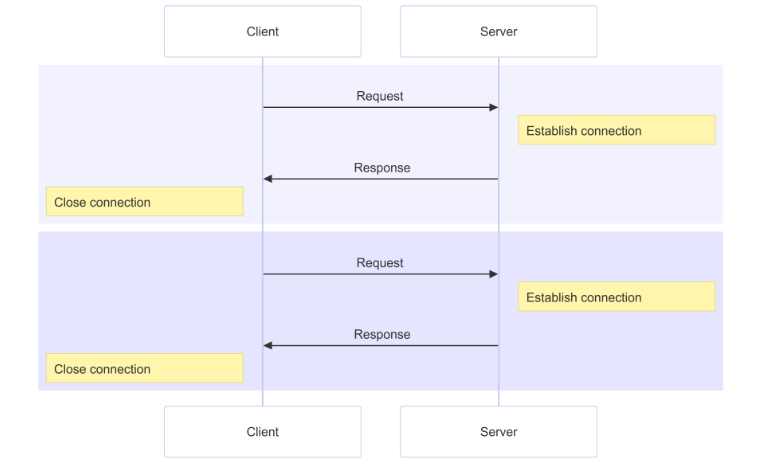
Respostas com um código de status que responde à solicitação sem a necessidade de incluir o conteúdo da mensagem, como 201 Criado ou 204 Sem Conteúdo, não têm um corpo.

**HTTP/2 messages**

O HTTP/1.x utiliza mensagens baseadas em texto que são simples de ler e construir, mas, como resultado, apresentam algumas desvantagens. É possível compactar o corpo das mensagens usando gzip ou outros algoritmos de compactação, mas não os cabeçalhos. Os cabeçalhos costumam ser semelhantes ou idênticos em uma interação cliente-servidor, mas são repetidos em mensagens sucessivas em uma conexão. Existem muitos métodos conhecidos para compactar texto repetitivo que são muito eficientes, o que deixa uma grande economia de largura de banda não utilizada.

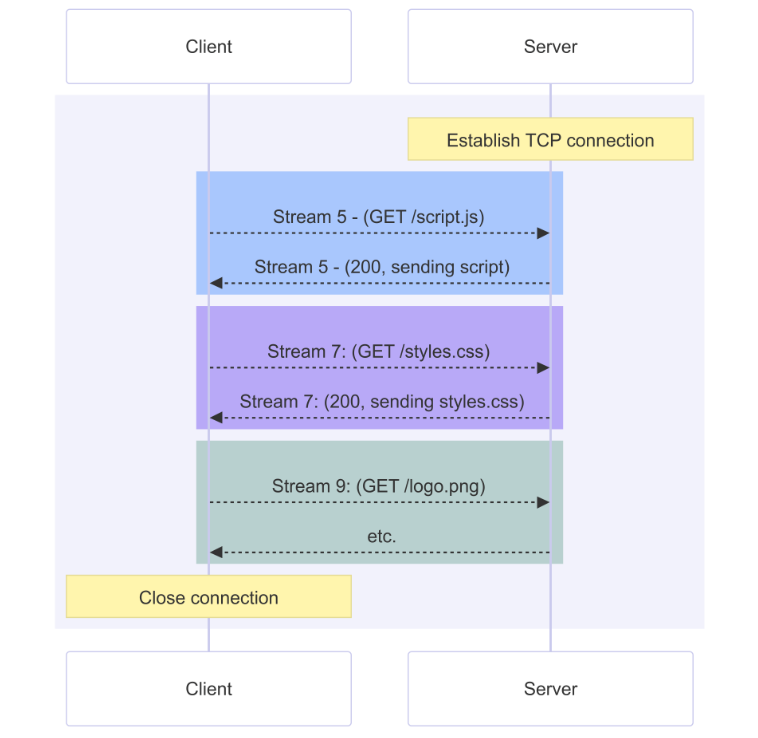
O HTTP/1.x também apresenta um problema chamado bloqueio de  head-of-line (HOL), em que um cliente precisa aguardar uma resposta do servidor antes de enviar a próxima requisição. O pipeline HTTP tentou contornar isso, mas o suporte deficiente e a complexidade fazem com que seja raramente utilizado e difícil de acertar. Várias conexões precisam ser abertas para enviar requisições simultaneamente; e conexões quentes (estabelecidas e ocupadas) são mais eficientes do que frias devido à inicialização lenta do TCP.

No HTTP/1.1, se você quiser fazer duas solicitações em paralelo, precisará abrir duas conexões:



Isso significa que os navegadores são limitados no número de recursos que podem baixar e renderizar ao mesmo tempo, o que normalmente é limitado a 6 conexões paralelas.

O HTTP/2 permite que você use uma única conexão TCP para múltiplas requisições e respostas simultaneamente. Isso é feito encapsulando mensagens em um quadro binário e enviando as requisições e respostas em um fluxo numerado em uma conexão. Os quadros de dados e cabeçalhos são tratados separadamente, o que permite a compactação dos cabeçalhos por meio de um algoritmo chamado HPACK. Usar a mesma conexão TCP para lidar com múltiplas requisições simultaneamente é chamado de multiplexação.



As solicitações não são necessariamente sequenciais: o fluxo 9 não precisa esperar o fluxo 7 terminar, por exemplo. Os dados de vários fluxos geralmente são intercalados na conexão, permitindo que os fluxos 9 e 7 sejam recebidos pelo cliente simultaneamente. Existe um mecanismo no protocolo para definir uma prioridade para cada fluxo ou recurso. Recursos de baixa prioridade ocupam menos largura de banda do que recursos de alta prioridade quando enviados por fluxos diferentes, ou podem ser enviados sequencialmente na mesma conexão se houver recursos críticos que devam ser tratados primeiro.

Em geral, apesar de todas as melhorias e abstrações adicionadas ao HTTP/1.x, praticamente não são necessárias alterações nas APIs usadas pelos desenvolvedores para usar HTTP/2 sobre HTTP/1.x. Quando o HTTP/2 está disponível tanto no navegador quanto no servidor, ele é ativado e usado automaticamente.

**Pseudo-headers**

Uma mudança notável nas mensagens em HTTP/2 é o uso de pseudocabeçalhos. Enquanto o HTTP/1.x usava a linha inicial da mensagem, o HTTP/2 usa campos especiais de pseudocabeçalho que começam com :. Nas solicitações, existem os seguintes pseudocabeçalhos:

:method - o método HTTP.

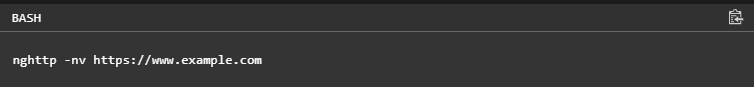
:scheme - a parte do esquema do URI de destino, que geralmente é HTTP(S).

:authority - a parte da autoridade do URI de destino.

:path - as partes do caminho e da consulta do URI de destino.

Nas respostas, há apenas um pseudocabeçalho, que é o :status, que fornece o código da resposta.

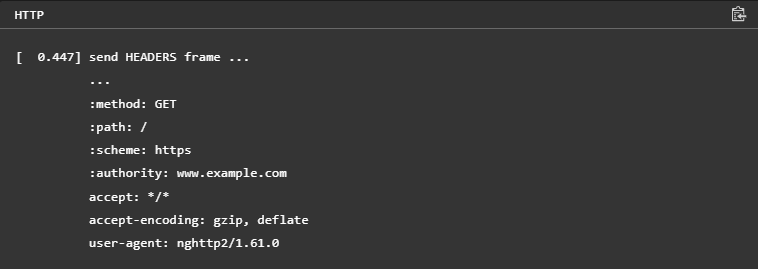
Podemos fazer uma requisição HTTP/2 usando nghttp para buscar example.com, o que imprimirá a requisição em um formato mais legível. Você pode fazer a requisição usando este comando, onde a opção -n descarta os dados baixados e -v é para uma saída "detalhada", mostrando a recepção e a transmissão de quadros:



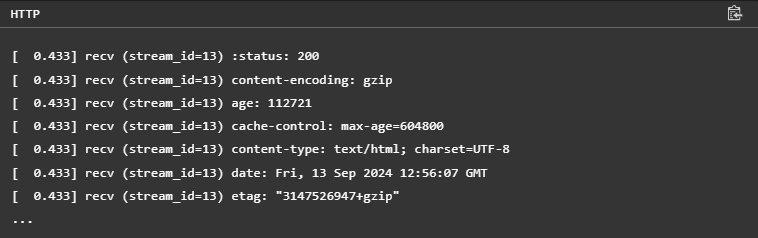
Se você olhar a saída, verá o tempo de cada quadro transmitido e recebido:



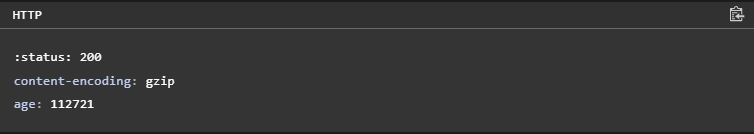
Não precisamos entrar em muitos detalhes sobre esta saída, mas procure pelo quadro HEADERS no formato [ 0.123] send HEADERS frame .... Nas linhas após a transmissão do cabeçalho, você verá as seguintes linhas:



Isso deve parecer familiar se você já estiver familiarizado com HTTP/1.x e os conceitos abordados na seção anterior deste guia ainda se aplicarem. Este é o quadro binário com a solicitação GET para example.com, convertida em um formato legível pelo nghttp. Se você observar mais adiante na saída do comando, verá o pseudocabeçalho :status em um dos fluxos recebidos do servidor:



E se você remover o tempo e o ID do fluxo desta mensagem, ela deverá ficar ainda mais familiar:



Explorar mais profundamente os quadros de mensagens, IDs de fluxo e como a conexão é gerenciada está além do escopo deste guia, mas para entender e depurar mensagens HTTP/2, você deve estar bem equipado usando o conhecimento e as ferramentas deste artigo.

**Conclusão**

Este guia fornece uma visão geral da anatomia das mensagens HTTP, utilizando o formato HTTP/1.1 como ilustração. Também exploramos o enquadramento de mensagens HTTP/2, que introduz uma camada entre a sintaxe HTTP/1.x e o protocolo de transporte subjacente, sem modificar fundamentalmente a semântica do HTTP. O HTTP/2 foi introduzido para solucionar os problemas de bloqueio de cabeçalho de linha presentes no HTTP/1.x, permitindo a multiplexação de solicitações.

Um problema que persistiu no HTTP/2 é que, embora o bloqueio de cabeça de linha tenha sido corrigido no nível do protocolo, ainda há um gargalo de desempenho devido ao bloqueio de cabeça de linha dentro do TCP (no nível do transporte). O HTTP/3 soluciona essa limitação usando o QUIC, um protocolo baseado em UDP, em vez do TCP. Essa mudança melhora o desempenho, reduz o tempo de configuração da conexão e aprimora a estabilidade em redes degradadas ou não confiáveis. O HTTP/3 mantém a mesma semântica básica do HTTP, de modo que recursos como métodos de solicitação, códigos de status e cabeçalhos permanecem consistentes em todas as três principais versões do HTTP.

Se você entende a semântica do HTTP/1.1, já tem uma base sólida para entender o HTTP/2 e o HTTP/3. A principal diferença está em como essas semânticas são implementadas no nível de transporte. Seguindo os exemplos e conceitos deste guia, você agora se sentirá preparado para trabalhar com HTTP e entender o significado das mensagens e como os aplicativos usam HTTP para enviar e receber dados.