**HTTP**

***Hypertext Transfer Protocol (HTTP)*** é um protocolo da camada de aplicação para transmissão de documentos hipermídia, como HTML. Foi projetado para comunicação entre navegadores e servidores web, mas também pode ser usado para outros propósitos, como comunicação máquina a máquina, acesso programático a APIs e muito mais. O HTTP segue um modelo clássico cliente-servidor, com um cliente abrindo uma conexão para fazer uma solicitação e aguardando até receber uma resposta do servidor. O HTTP é um protocolo stateless, o que significa que o servidor não mantém nenhum dado de sessão entre duas solicitações, embora a adição posterior de cookies adicione estado a algumas interações cliente-servidor.

***Referência***

A documentação de referência HTTP contém informações detalhadas sobre cabeçalhos, métodos de solicitação, respostas de status e lista especificações e documentos de padrões relevantes.

**HTTP headers**

Cabeçalhos de mensagem são usados ​​para enviar metadados sobre um recurso ou uma mensagem HTTP e para descrever o comportamento do cliente ou do servidor.

**HTTP request methods**

Os métodos de solicitação indicam a finalidade da solicitação e o que se espera em caso de sucesso. Os métodos mais comuns são GET e POST para recuperar e enviar dados aos servidores, respectivamente, mas existem outros métodos que atendem a propósitos diferentes.

**HTTP response status codes**

Os códigos de status de resposta indicam o resultado de uma solicitação HTTP específica. As respostas são agrupadas em cinco classes: informational, successful, redirections, client errors, and server errors.

**HTTP resources and specifications**

Esta página lista recursos relevantes sobre HTTP desde sua primeira especificação no início da década de 1990. As seguintes subseções também são dignas de nota:

**CSP directives**

O cabeçalho de resposta Content-Security-Policy (CSP) permite que os administradores de sites especifiquem quais recursos o agente do usuário tem permissão para carregar para uma determinada página. Esta seção lista as diretivas que podem ser usadas em um cabeçalho CSP, com páginas de documentação individuais que descrevem como as diretivas funcionam e como usá-las.

**Permissions-Policy directives**

O cabeçalho de resposta Permissions-Policy fornece um mecanismo para permitir ou negar o uso de recursos do navegador em um documento ou em qualquer elemento <iframe> do documento. Esta seção lista as diretivas que podem ser usadas em um cabeçalho Permissions-Policy, com páginas de documentação individuais que descrevem como as diretivas funcionam e como usá-las.

***Guias***

HTTP é um protocolo extensível que se baseia em conceitos como recursos e Uniform Resource Identifiers (URIs), uma estrutura básica de mensagens e um modelo de comunicação cliente-servidor. Além desses conceitos, inúmeras extensões foram desenvolvidas ao longo dos anos, adicionando funcionalidades e atualizando a semântica, incluindo métodos e cabeçalhos HTTP adicionais. Os guias HTTP estão listados em ordem, desde visões gerais até tópicos especializados, baseados em casos de uso. Iniciantes são incentivados a começar pelos guias básicos antes de explorar artigos mais específicos.

**Overview of HTTP**

Os recursos básicos do HTTP, o que ele pode fazer, seu uso pretendido na arquitetura da web e sua posição na pilha de protocolos.

**Evolution of HTTP**

O HTTP foi criado no início da década de 1990 e foi expandido diversas vezes. Este artigo analisa sua história e descreve HTTP/0.9, HTTP/1.0, HTTP/1.1, passando por HTTP/2 e HTTP/3, bem como as novidades introduzidas ao longo dos anos.

**A typical HTTP session**

Descreve o fluxo de uma sessão HTTP, desde o estabelecimento de uma conexão, envio de uma solicitação até o recebimento de uma resposta.

**HTTP messages**

As mensagens HTTP transmitidas como solicitações e respostas têm uma estrutura definida. Este artigo descreve essa estrutura geral, sua finalidade e os diferentes tipos de mensagens.

**MIME types**

Desde o HTTP/1.0, diferentes tipos de conteúdo podem ser transmitidos. Este artigo explica como isso é feito usando o cabeçalho Content-Type e o padrão MIME. Uma lista resumida de tipos comuns usados ​​por desenvolvedores web pode ser encontrada em Common MIME types

**Compression in HTTP**

Navegadores e servidores compactam suas mensagens antes de enviá-las pela rede para reduzir a quantidade de dados que precisam ser transmitidos, melhorando a velocidade de transferência e a utilização da largura de banda.

**HTTP caching**

O cache é um mecanismo extremamente importante para proporcionar experiências rápidas na web e para o uso eficiente de recursos. Este artigo descreve diferentes métodos de cache e como usar cabeçalhos HTTP para controlá-los.

**HTTP authentication**

A autenticação é uma forma de verificar a identidade de um cliente ao fazer solicitações a um servidor. Ela garante que apenas usuários ou sistemas autorizados possam acessar determinados recursos.

**Using HTTP cookies**

Embora o HTTP seja um protocolo sem estado, um servidor pode enviar um cabeçalho Set-Cookie com a resposta. O cliente então retorna o valor do cookie a cada solicitação subsequente ao servidor na forma de um Cookie request header. Isso adiciona a capacidade de armazenar e trocar uma pequena quantidade de dados, o que efetivamente adiciona estado a algumas interações cliente-servidor.

**Redirections in HTTP**

O redirecionamento de URL, também conhecido como URL forwarding, é uma técnica para atribuir mais de um endereço de URL a uma página, um formulário, um site inteiro ou um aplicativo web. O HTTP possui um tipo especial de resposta, chamado redirecionamento HTTP, para essa operação.

**HTTP conditional requests**

Em solicitações condicionais, o resultado de uma solicitação depende do valor de um validador na solicitação. Esse método é muito utilizado em cache e em casos de uso como retomar um download, evitar a perda de atualizações ao modificar um documento no servidor e muito mais.

**HTTP range requests**

Um range request solicita ao servidor que envie uma parte específica (ou partes) de um recurso de volta para um cliente, em vez do recurso completo. As solicitações de intervalo são úteis quando um cliente sabe que precisa apenas de parte de um arquivo grande ou quando um aplicativo permite que o usuário pause e retome um download.

**Content negotiation**

O HTTP define um conjunto de cabeçalhos de mensagem, começando com "Accept", como uma forma de um navegador anunciar o formato, idioma ou codificação de sua preferência. Este artigo explica como esse anúncio acontece, como o servidor deve reagir e como ele escolhe a resposta mais adequada a uma solicitação.

**Connection management in HTTP/1.x**

HTTP/1.1 foi a primeira versão do HTTP a oferecer suporte a persistent connections e pipelining. Este artigo explica ambos os conceitos, incluindo os prós e contras de cada um.

**Protocol upgrade mechanism**

O HTTP/1.1 fornece um mecanismo para atualizar uma conexão já estabelecida para um protocolo diferente usando o Upgrade header. Um cliente pode atualizar uma conexão de HTTP/1.1 para HTTP/2, ou uma conexão HTTP(S) para um WebSocket (ws / wss).

**Proxy servers and tunneling**

Um proxy pode estar no computador local do usuário ou em qualquer lugar entre o computador do usuário e um servidor de destino na internet. Esta página descreve alguns conceitos básicos sobre proxies e apresenta algumas opções de configuração.

**HTTP Client hints**

Client Hints são um conjunto de cabeçalhos de resposta que um servidor pode usar para solicitar proativamente informações de um cliente sobre o dispositivo, a rede, o usuário e as preferências do user-agent-specific. O servidor pode então determinar quais recursos enviar, com base nas informações que o cliente escolher fornecer.

**Network Error Logging (Experimental)**

O Network Error Logging é um mecanismo que pode ser configurado por meio do cabeçalho de resposta NEL HTTP. Este cabeçalho experimental permite que sites e aplicativos optem por receber relatórios sobre buscas de rede com falha (ou até mesmo bem-sucedidas) de navegadores compatíveis.

**Browser detection using the user agente**

Raramente é uma boa ideia usar user agent sniffing para detectar um navegador, mas há casos extremos que exigem isso. Este documento orientará você a fazer isso da forma mais correta possível quando necessário, com ênfase nas considerações a serem feitas antes de embarcar nessa rota.

***Security and Privacy***

**Permissions Policy**

A Permissions Policy fornece mecanismos para que desenvolvedores web declarem explicitamente quais funcionalidades podem ou não ser usadas em um site. Você define um conjunto de "políticas" que restringem quais APIs o código do site pode acessar ou modificam o comportamento padrão do navegador para determinados recursos.

**Cross-Origin Resource Sharing (CORS)**

Cross-site HTTP requests entre sites são requisições de recursos de um domínio diferente daquele do recurso que faz a requisição. Hoje em dia, páginas da web comumente carregam recursos entre sites. Por exemplo, uma página "Domínio A" (http://domaina.example/) solicita uma imagem no "Domínio B" (http://domainb.foo/image.jpg) por meio do elemento img. O CORS permite que desenvolvedores web controlem como seus sites reagem a requisições entre sites.

**Content Security Policy (CSP)**

O CSP permite que administradores de sites usem o cabeçalho de resposta Content-Security-Policy para controlar quais recursos o cliente tem permissão para carregar para uma determinada página. O guia do CSP descreve o mecanismo geral da Content Security Policy, que ajuda a detectar e mitigar certos tipos de ataques, incluindo Cross-Site Scripting (XSS) e data injection attacks.

**Cross-Origin Resource Policy (CORP)**

O CORP permite que sites e aplicativos optem pela proteção contra solicitações específicas de outras origens (como aquelas emitidas com elementos como <script> e <img>), para mitigar ataques especulativos de canal lateral.

**Mozilla web security guidelines**

Uma coleção de dicas para ajudar equipes operacionais a criar aplicativos web seguros.

***Related Resources***

**URIs**

Identificadores Uniformes de Recursos (URIs) são usados ​​para descrever e localizar recursos na web e são um componente essencial em solicitações HTTP.

**Configuring servers for Ogg media**

Este guia aborda algumas alterações na configuração do servidor que podem ser necessárias para que seu servidor web sirva corretamente arquivos de mídia Ogg. Essas informações também podem ser úteis caso você encontre outros tipos de mídia que seu servidor ainda não esteja configurado para reconhecer.