HTML5 Professional Certification

HTML5プロフェッショナル認定 レベル1 技術解説無料セミナー

2023/09/30 開催

主題	1.1.1 HTTP, HTTPSプロトコル
副題	HTTPの基礎知識 （リクエストのメソッド、レスポンスのステータスコード）

本日の講師

FUJITSU
株式会社富士通ラーニングメディア
吉田 隼人

LPI-JAPAN

自己紹介

- 吉田隼人
  - 所属企業：　株式会社富士通ラーニングメディア
  - 担当業務：　研修講座の運営、登壇
  - 趣味：　旅行、ゴルフ、英会話


Batu Caves, Malaysia

- これまでに登壇した研修講座の分野
  - C言語、C++、Java、C#、VB
  - 組み込みソフトウェア開発（6502マイコン互換機の機械語・アセンブリ言語）
  - **HTML、CSS、JavaScript**
  - ソフトウェアテスト技術
  - オブジェクト指向デザインパターン
  - データベース論理設計、オブジェクト指向データモデリング
  - サーバー仮想化ソフトウェア、クラウドコンピューティング
- 登壇以外の業務
  - クラウド人材の育成企画立案および推進
  - クラウドサービスの運用支援、マニュアル開発（DITA、XSLTスタイルシート）
  - 業務マニュアル開発のプロジェクトマネジメント
  - 無線LAN環境の構築（RADIUSサーバーの構築）
  - **社内教育ポータルサイト**の機能開発（PHP、jQuery、PostgreSQL）

所属企業のご紹介

株式会社富士通ラーニングメディア

- **人材育成**に関するコンサルティング、人材力診断／適性診断等の提供
- **研修講座**（コース、カリキュラム）の企画、開発、実施、運営および運営支援
- コース教材／**マニュアル**等の開発、制作、**翻訳**、**出版**および販売
- 人材／研修講座の運営／マニュアル制作の管理に関連する**ソフトウェア**の開発および販売

関連講座

HTMLとCSSによるホームページ作成

HTMLとCSSによるホームページ作成方法について、講義およびテキストエディタを使用した実習によって学習します。**基本的なHTMLの記述**として、文章の表現、画像の表示、リンクの設定、フォームの作成方法などを修得します。またCSSによってWebページのデザインを設定する方法を修得します。

集合開催　Web開催

JavaScriptプログラミング基礎

Webアプリケーションを実装する際に使用する**JavaScriptの基本文法**を学習します。制御文、関数、イベント処理といったJavaScriptの文法に加え、オブジェクトを使用して、文字列操作、ウィンドウ操作、フォームの入力チェックなどを実装する方法について、説明と実習によって学習します。

集合開催　Web開催

JavaScript基礎ステップアップAPI編 ～ストレージ・通信・デバイスの利用

Webページのオフライン動作やサーバとの通信処理、ハードウェアアクセスなどを実現するためのWeb StorageやWebSocket、Geolocation APIなどのJavaScript**API**を講義と実習によって学習します。

集合開催　Web開催

■HTML5プロフェッショナル認定とは

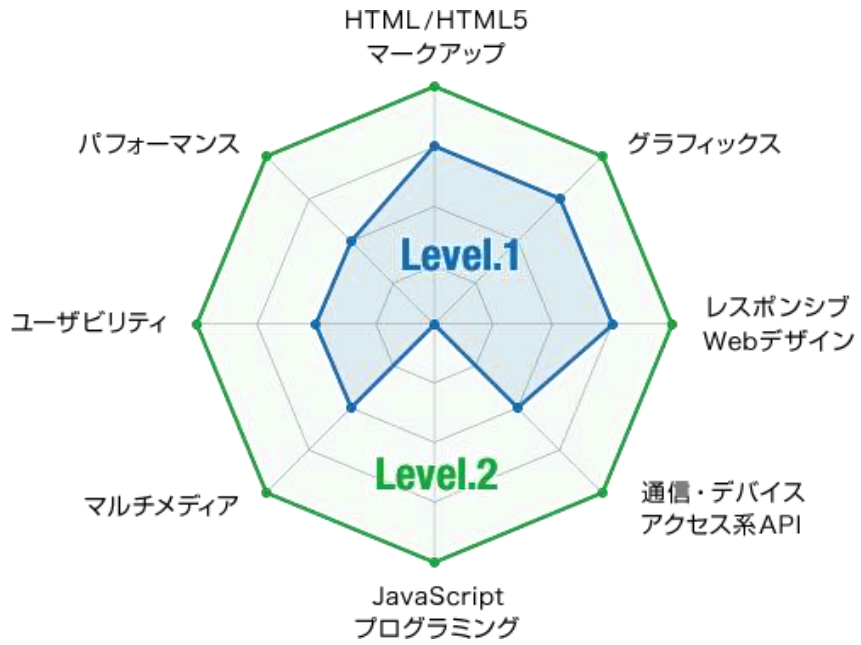
WEBサイトやWEBアプリケーションを開発する上で必須である HTML5／CSS JavaScriptなどについての技術力を証明する認定です。基礎から網羅的に学ぶことは、効率的に開発を行う上できっと役立つことでしょう。

✓レベル１はHTMLとCSS

HTMLの基本的な部分からレスポンシブデザインが中心で、サイト制作のためのスキルの証明

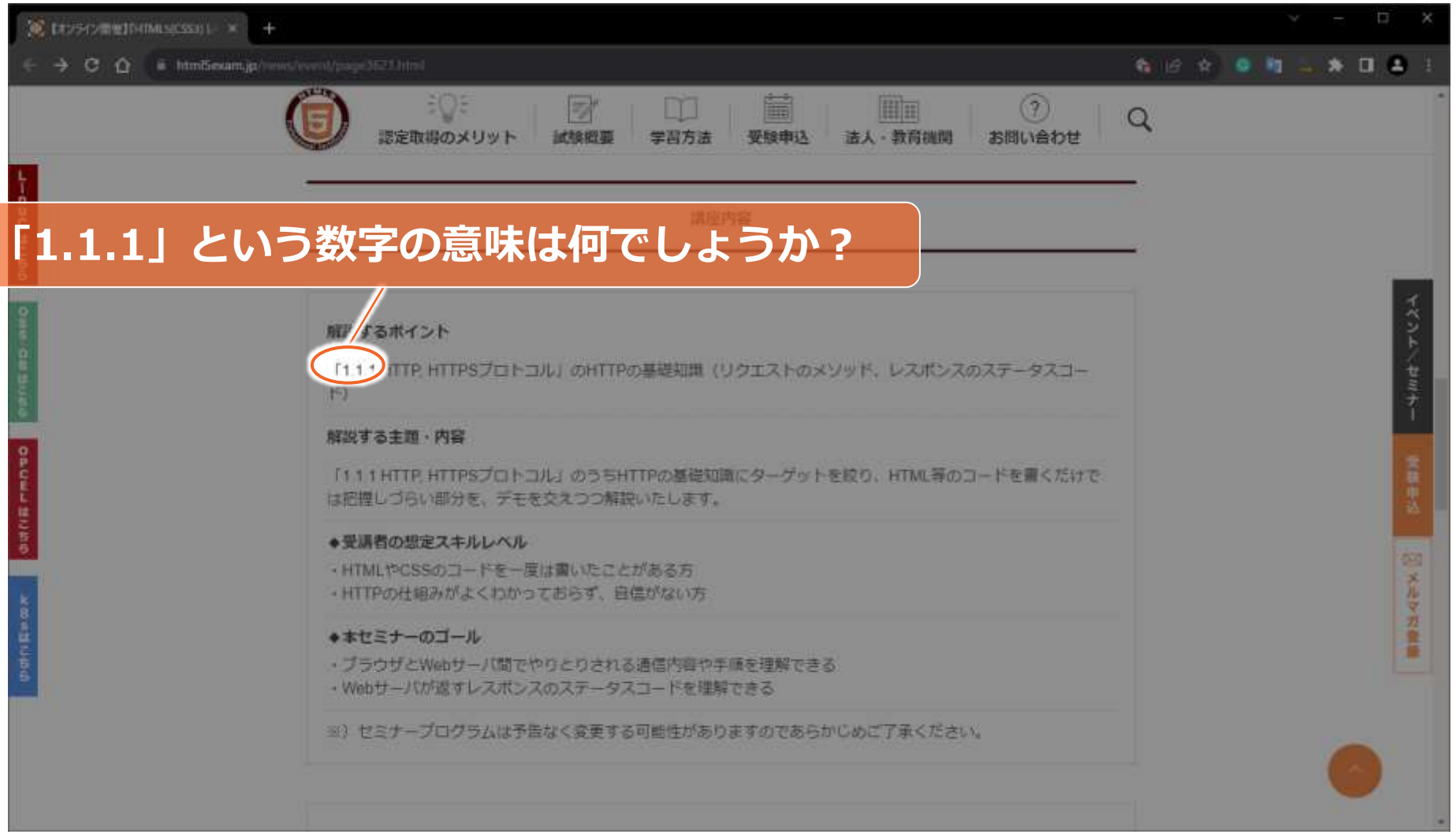
✓レベル２ではJavaScript

JavaScriptを使ってWEBアプリケーションを構築できるだけのスキルの証明



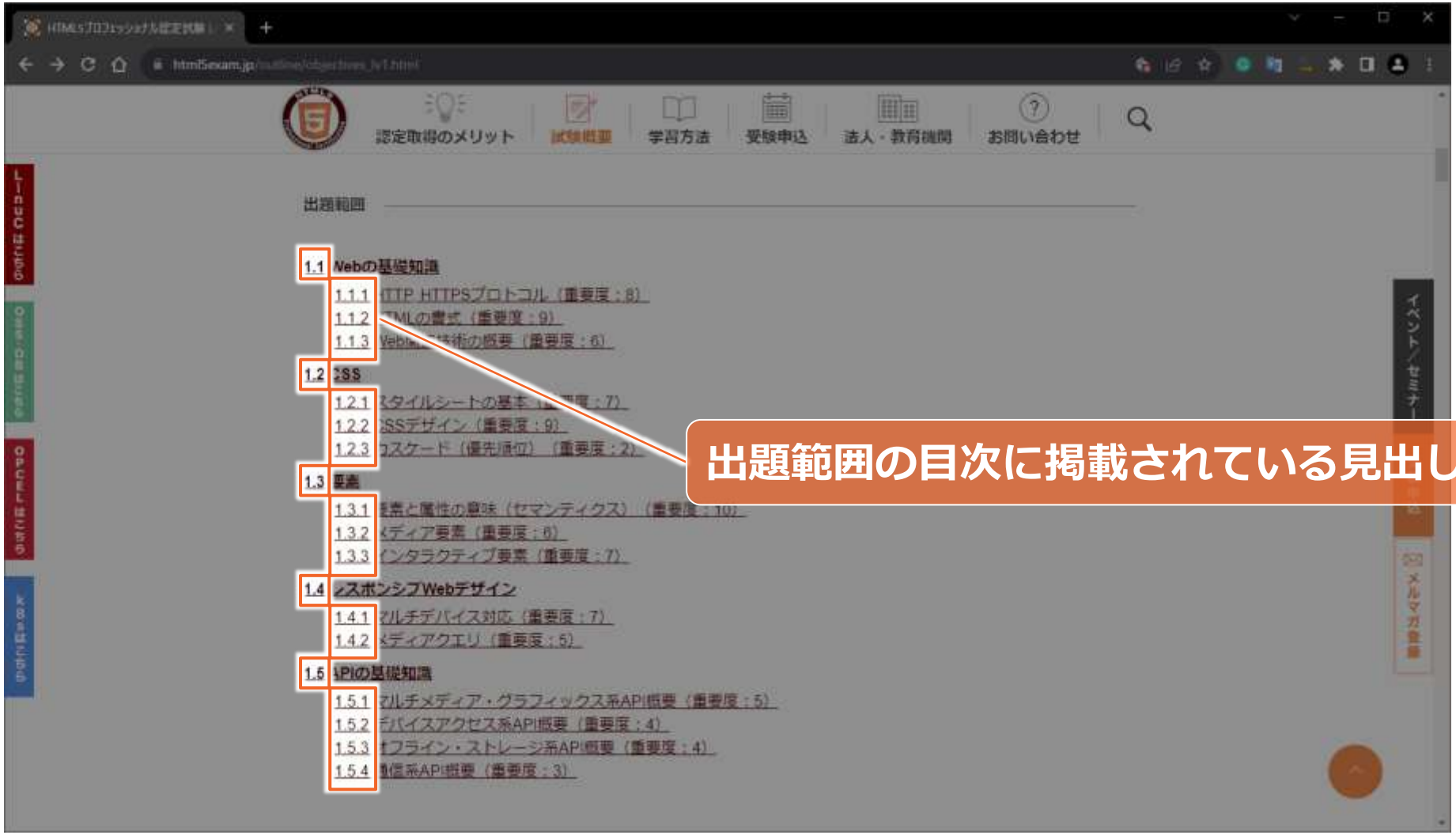
今回のテーマ

■ 今回のセミナーのお申し込み画面



今回のテーマ

HTML5プロフェッショナル認定試験 レベル1 出題範囲ページ



今回のテーマ

- 「1.1.1 HTTP, HTTPSプロトコル」の詳細

1.1.1 HTTP, HTTPSプロトコル

重要度

★★★★★★★★ 8

出題種別

- 知識問題
- 記述問題

説明（望まれるスキル）

HTTPのコンテンツ作成や、サイト全体の設計を行うために必要なHTTPおよびHTTPSプロトコルに関する知識を有している。
また、ブラウザでアクセスした時に返ってくるエラーコードの意味を理解できて、問題を解決するヒントとする事ができる。

主要な知識範囲

- ブラウザとWebサーバ間でやりとりされる通信内容や手順
- HTTPリクエストにおけるメソッド種類と違い
- HTTPプロトコルバージョンによる違い
- リクエストURIの仕様について書式や利用可能文字
- Webサーバが返すレスポンスのヘッダ項目
- Webサーバが返すレスポンスのステータスコード
- HTTPプロトコルに規定されている認証方式

重要な技術要素

- HTTP, HTTPS, SSL/TLS
- リクエストメソッド（GET, POST, HEAD, PUT, DELETEなど）
- HTTP/1.1, HTTP/2
- URI, URL
- ステータスコード, リダイレクト
- HTTP Header Fields（Accept, Authorization, Cache-Control, Content-Language, Expiresなど）
- Basic認証、Digest認証
- HTTP cookie

この辺りが今回のテーマです。

本日の目次

1. HTMLとHTTP
2. HTTPの通信内容を覗いてみよう
3. HTTPリクエストのメソッドの代表例
4. HTTPレスポンスのステータスコードの代表例

Hyper Text Markup Language / Hyper Text Transfer Protocol

1. HTMLとHTTP

Webページにアクセスするときに何が起きているか？

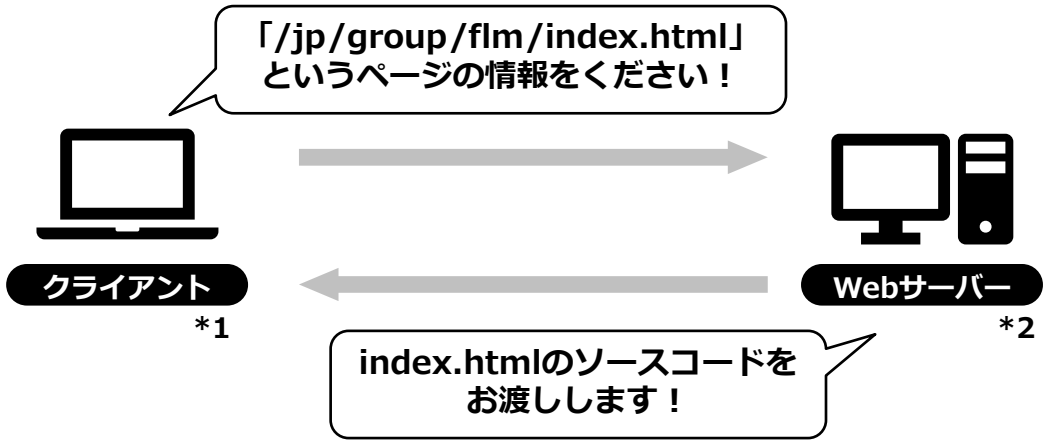
- 「https://www.fujitsu.com/jp/group/flm/index.html」にアクセスしたとき…

人間の目から見える世界

Webブラウザの画面に、下図のような内容が表示される。

コンピューターの中の世界

Webブラウザに表示される内容を下図のように授受している。



*1: Webブラウザの持ち主
*2: 「www.fujitsu.com」という名前の持ち主

次ページへ続く

Webページにアクセスするときに何が起きているか？

- 「https://www.fujitsu.com/jp/group/flm/index.html」のソース（一部省略、抜粋）

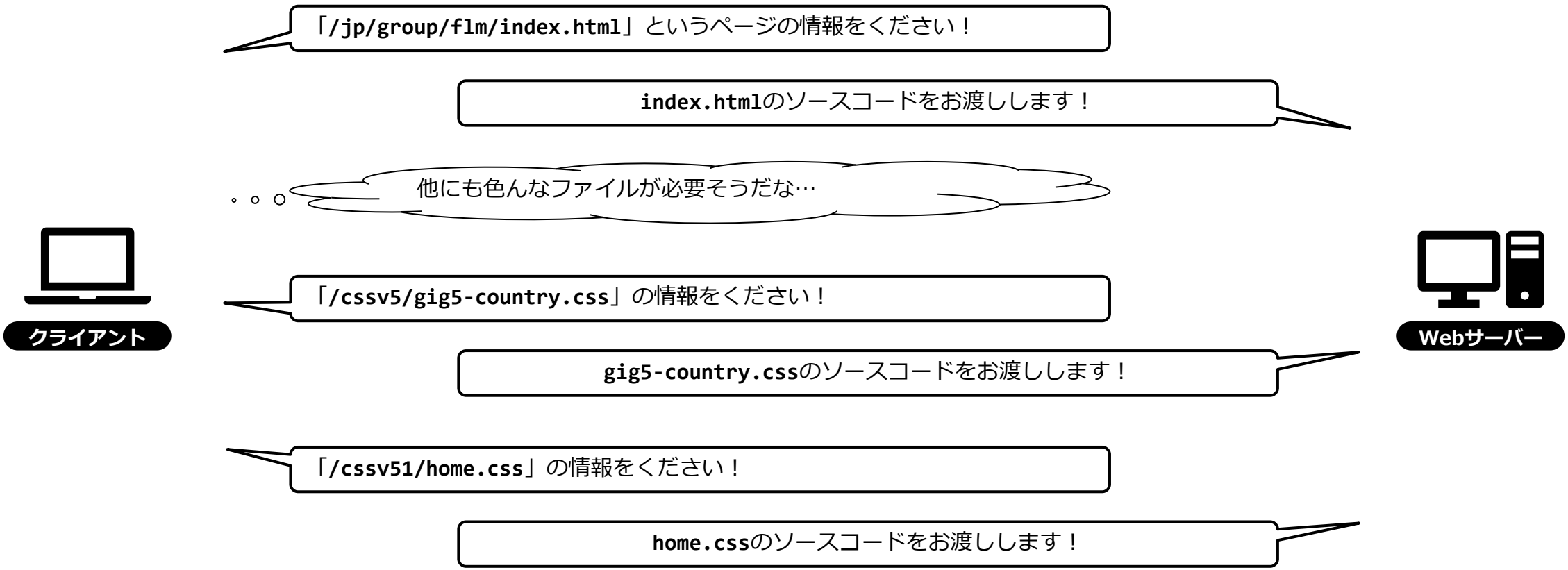
```
<!DOCTYPE html>
<html lang="ja">
    <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width">
    <meta name="description" content="株式会社富士通ラーニングメディアは、「知」のサービスを通じ、常に新しい価値創造に努める～～～">
    <meta name="keywords" content="富士通ラーニングメディア，富士通ラーニングメディア、FLM,教育,研修,ドキュメント,富士通オープンカレッジ">
    <meta name="language" content="ja">
    <meta name="robots" content="">
    <meta property="og:title" content="富士通ラーニングメディア">
    <meta property="og:type" content="website">
    <meta property="og:description" content="株式会社富士通ラーニングメディアは、「知」のサービスを通じ、常に新しい価値創造に努める～～～">
    <meta property="og:url" content="https://www.fujitsu.com/jp/group/flm/">
    <meta property="og:site_name" content="富士通ラーニングメディア">
    <meta property="og:image" content="https://www.fujitsu.com/jp/group/flm/imagesgig5/OGP画像用_1200×630px_tcm211-7278384_tcm211-2750236-32.png">
    <meta name="twitter:card" content="summary_large_image">
    <meta name="twitter:title" content="富士通ラーニングメディア">
    <meta name="twitter:description" content="株式会社富士通ラーニングメディアは、「知」のサービスを通じ、常に新しい価値創造に努める～～～">
    <meta name="twitter:image" content="https://www.fujitsu.com/jp/group/flm/imagesgig5/OGP画像用_1200×630px_tcm211-7278384_tcm211-2750236-32.png">

    <title>富士通ラーニングメディア ： 富士通ラーニングメディア</title>

    <link rel="canonical" href="https://www.fujitsu.com/jp/group/flm/">
    <link rel="stylesheet" href="/cssv5/gig5-country.css">
    <link rel="stylesheet" href="/cssv51/home.css">
    <link rel="stylesheet" href="/cssv51/lf-component.css"/>
    <link rel="stylesheet" href="/cssv5/gig5-multibyte.css">

    <script src="/incv5/init.js"></script>
```

index.htmlのソース内容だけでは完結しておらず、cssやjsなどの他のファイルも必要としている。

次ページへ続く

Webページにアクセスするときに何が起きているか？

- クライアントとWebサーバーの間では複数回のやり取りが発生している。

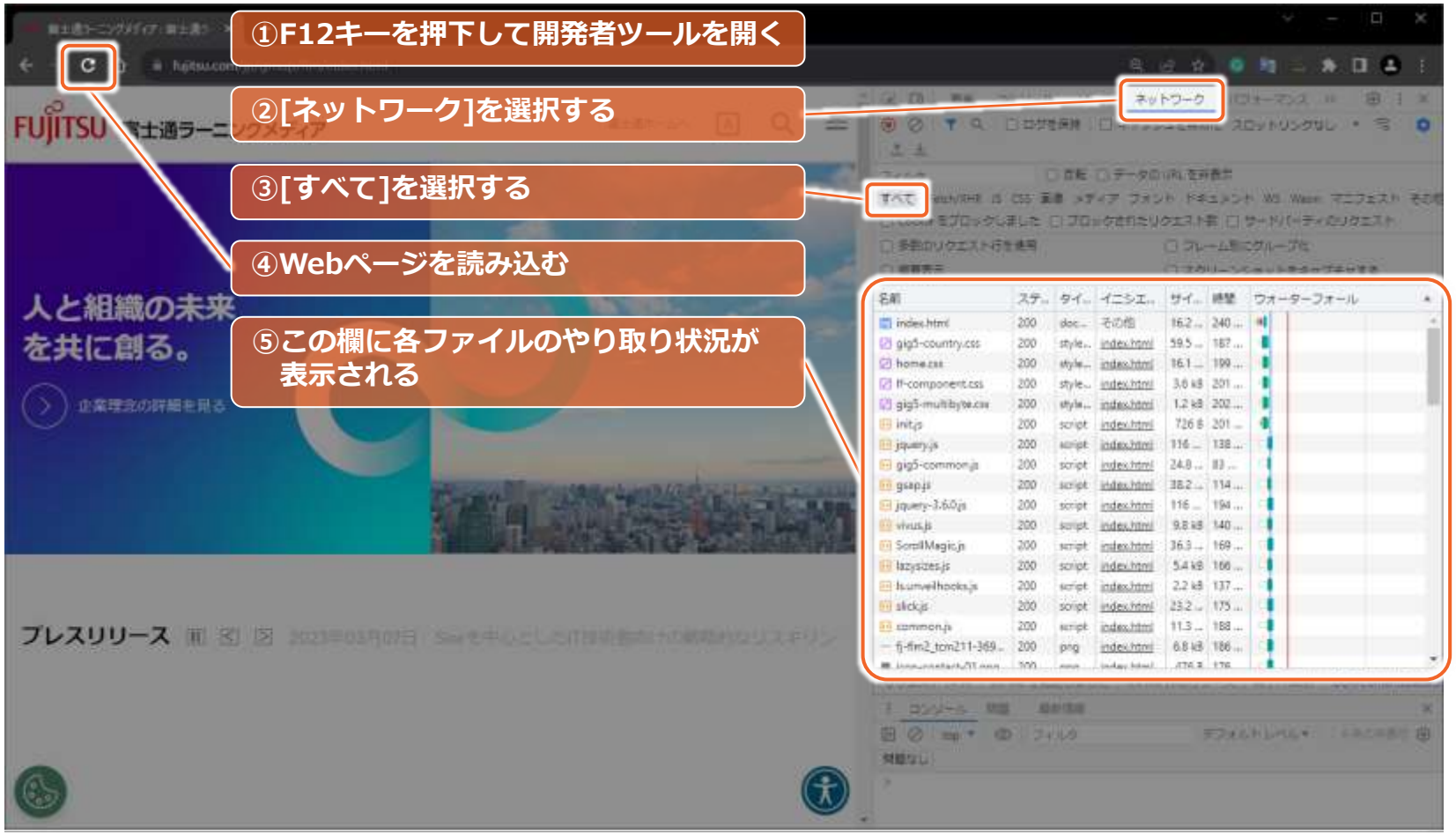


（以下省略）

次ページへ続く

Webページにアクセスするときに何が起きているか？

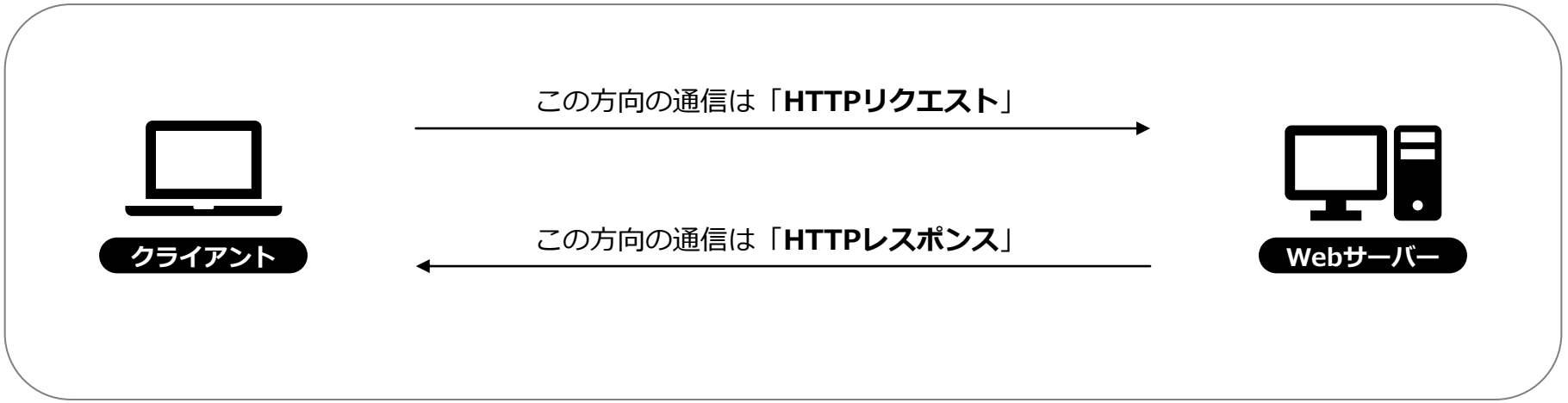
- Webブラウザの開発者ツールを見ると、複数回のやり取りが発生していることを確認できます。



次ページへ続く

Webページにアクセスするときに何が起きているか？

- なお、クライアントとWebサーバーとのやり取りには下記のように名前がついています。
  - ✓ クライアントからWebサーバーに対して依頼を行うための通信を「**HTTPリクエスト**」といいます。
  - ✓ Webサーバーからクライアントに対して応答を返すための通信を「**HTTPレスポンス**」といいます。



- Webページにアクセスするときに何が起きているか？

ここまでのまとめ

クライアントとWebサーバーとの間で、HTTPリクエストとHTTPレスポンスのやり取りが複数回行われている。

1. HTMLとHTTP

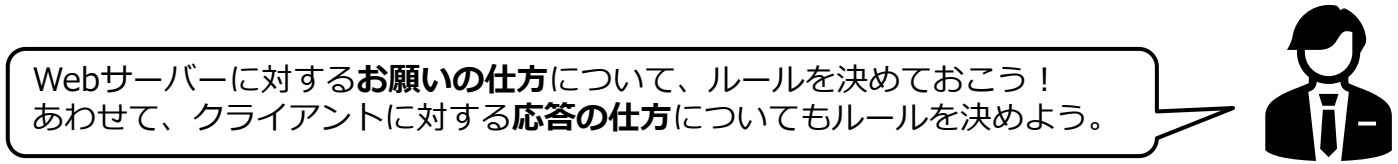
HTMLとは？

- Hyper Text Markup Languageの略称。
- 文書の構造や構成要素の意味、他のファイルとの繋がり等を表現するための言語。

HTTPとは？

- Hyper Text Transfer Protocolの略。
- HTMLファイル等をやり取りする際に適用される通信規約。

このようなことがあると困るので…



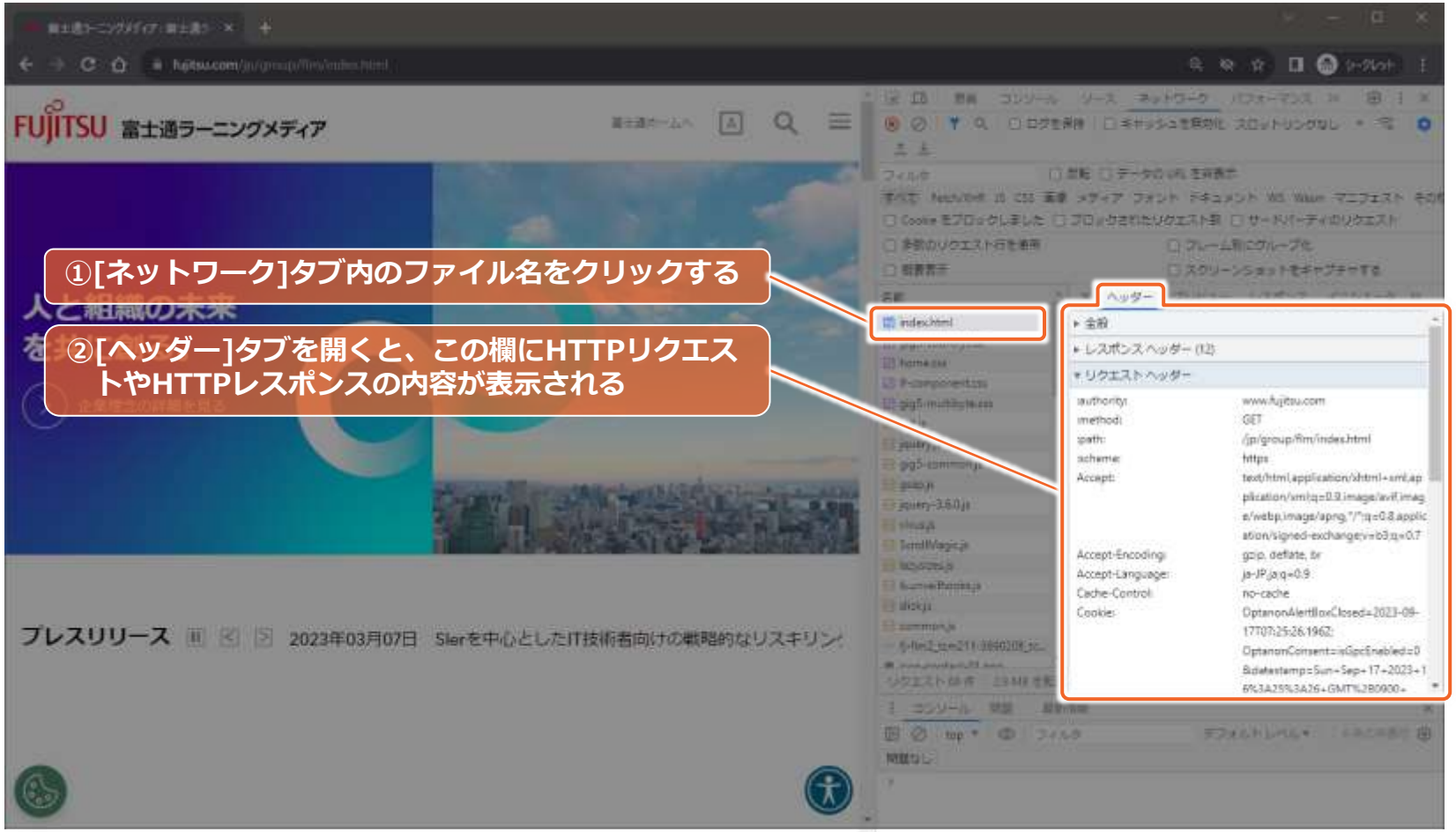
1. HTMLとHTTP

- 余談
  - HTTPは「**Hyper Text** Transfer Protocol」という名前であるため「HTMLファイルを転送するときに使う技術なのかな」と誤解されるかもしれませんが、CSSファイルや画像ファイルなど、HTMLファイルを含む様々なリソースを転送するときに使われます。

2. HTTPの通信内容を覗いてみよう

2. HTTPの通信内容を覗いてみよう

- 先ほども使ったWebブラウザの開発者ツールで、HTTPの通信内容を確認できます。



2. HTTPの通信内容を覗いてみよう

前ページのHTTPリクエストの内容

- Chrome上では見やすく整形して表示されますが、整形前の形は次のとおりです。

```
GET /jp/group/flm/index.html HTTP/1.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Accept-Language: ja-JP,ja;q=0.9
Cache-Control: no-cache
Cookie: OptanonAlertBoxClosed=2023-09-17T07:25:26.196Z; OptanonConsent=isGpcEnabled=0&datestamp=Sun+Sep+17+2023+16%3A25%3A26+GMT%2B0900+(%E6%97%A5%E6%9C%AC%E6%A8%99%E6%BA%96%E6%99%82)&version=202307.1.0&browserGpcFlag=0&isIABGlobal=false&hosts=&consentId=a8b541ca-3f03-40f0-b2a7-2ef00ffa4189&interactionCount=1&landingPath=NotLandingPage&groups=C0001%3A1%2CC0003%3A0%2CC0004%3A0%2CC0002%3A0
Dnt: 1
Pragma: no-cache
Sec-Ch-Ua: "Chromium";v="116", "Not)A;Brand";v="24", "Google Chrome";v="116"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.0.0 Safari/537.36
Host: www.fujitsu.com
```

次ページへ続く

2. HTTPの通信内容を覗いてみよう

■ 前ページのHTTPリクエストが示す意味（意訳）

1. 今回のHTTPリクエストの**メソッドは「GET」**です。
2. 「**/jp/group/flm/index.html」の情報を提供して欲しい**です。
3. 今回使用する**HTTPのバージョンは1.1**です。
4. Webサーバーから応答として返ってきたときに理解できるデータの形式は、HTMLやXHTMLなど*1です。
5. 応答データがgzipなど*1の形式で圧縮されていても大丈夫です。
6. クライアント側が理解できる言語は日本語です。
7. プロキシサーバーやCDNなどのキャッシュからではなく、Webサーバーから最新の情報を返して欲しいです。
8. 以前、そちらのWebサーバーから覚えておいて欲しいと言われたCookieデータは「OptanonAlertBoxClosed=2023-09-17T07:25:26.196Z」など*1です。
9. このWebサイトでのトラッキングを拒否します。
10. HTTPリクエストの送信元であるブラウザは、Google Chromeのバージョン116です。
11. HTTPリクエストの送信元はモバイルデバイスではありません。
12. HTTPリクエストの送信元のOSはWindowsです。
13. HTTPリクエストの宛先はHTMLもしくはXMLです。
14. 今回のHTTPリクエストからこのWebサイトの閲覧を開始します。
15. 今回のHTTPリクエストは、アドレスバーへURLを入力したりブックマークを開くなどのユーザーの操作によって発生したものです。
16. 今回のHTTPリクエストは、ユーザーによって発信されたものです。
17. より安全なWebサイトにアクセスするように応答が返された場合は、それに従います。
18. HTTPリクエストの送信元であるユーザーエージェント（ブラウザやOSの種類などを識別するための情報）は「Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.0.0 Safari/537.36」です。
19. 今回のHTTPリクエストは「www.fujitsu.com」に対して送っています。

*1: 全てを列挙すると長くなってしまうような箇所は一部省略しています。

後ほど解説します

次ページへ続く

2. HTTPの通信内容を覗いてみよう

■ 前ページのHTTPリクエストヘッダーのrawデータ（再掲）

```
GET /jp/group/flm/index.html HTTP/1.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;
v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Accept-Language: ja-JP,ja;q=0.9
Cache-Control: no-cache
Cookie: OptanonAlertBoxClosed=2023-09
00+(%E6%97%A5%E6%9C%AC%E6%A8%99%E6%BA
03-40f0-b2a7-2ef00ffa4189&interaction
Dnt: 1
Pragma: no-cache
Sec-Ch-Ua: "Chromium";v="116", "Not)A
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 1
Host: www.fujitsu.com
```

HTTPでは、ルールとして次のようなことが規定されています。

✓ HTTPリクエストの**開始行**で、空白文字で区切って下記3点を指定すること。
1. HTTPメソッド（※後述）
2. Webサーバーが持つリソース
3. HTTPのバージョン

✓ 開始行に続いて、HTTP**リクエストヘッダー**を必要なだけ列挙すること。
- ヘッダー名とコロン（:）に続いて、設定値を記述します。
- 指定可能なヘッダー名やその意味も、HTTPにおいて規定されています。
- ヘッダー項目ごとに改行します。

✓ HTTPリクエストヘッダーに続いて、必要であれば**本文**を記述すること。
- HTTPメソッドとしてPOSTを利用する場合には、本文があることが多いです。
- HTTPメソッドとしてGETを利用する場合には、通常は本文は不要です。

ここまでのまとめ

クライアントがWebサーバーに送るHTTPリクエストには、
様々な情報が含まれている。

3. HTTPリクエストのメソッドの代表例

3. HTTPリクエストのメソッドの代表例

HTML5プロフェッショナル認定試験の例題

https://html5exam.jp/measures/lv1_1.html#q1_23

例題1.23「1.1.1 HTTP, HTTPSプロトコル」

レベル1の出題範囲「1.1.1 HTTP, HTTPSプロトコル」からの出題です。
HTTP/1.1プロトコルのリクエストメソッドの説明として正しくないのは次のうちどれか。

A. HTMLのフォームで指定できるのはGETとPOSTのみである。
B. GETでは、リクエストパラメータはURLに含まれるが、POSTではボディに含まれる。
C. GETではデータサイズの制限があるが、POSTにはない。
D. GETリクエストに対しては、PUTリクエスト付きのレスポンスメッセージが戻る。

※この例題は実際のHTML5プロフェッショナル認定試験とは異なります。
例題公開日：2018年7月3日

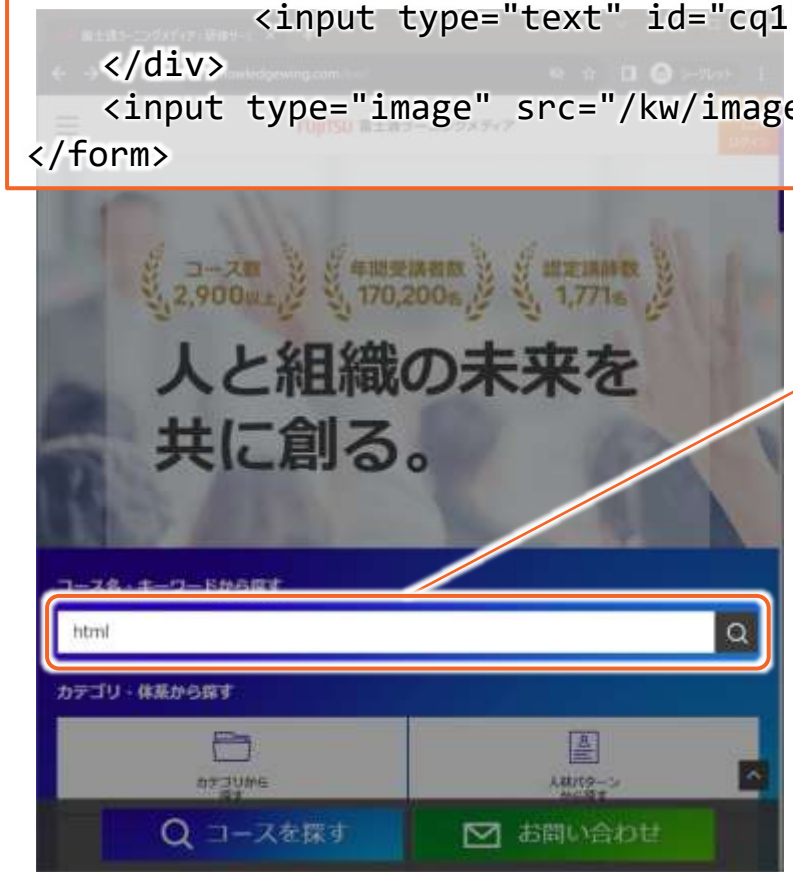
Aは、その通りなので、不正解です。
Bも、その通りなので、不正解です。ブラウザのURL欄に?kkk=vvv&KKK=VVV のようにKey=Valueが＆で区切られてリクエストパラメータが付いているのはGETリクエストのURLです。
Cも、その通りで、GETは255文字の制限がありますが、POSTは制限がないので、不正解です。
Dは、レスポンスメッセージにリクエストメソッドが含まれるとしていますが、リクエストメソッドは含まれずステータスコードが含まれるので、間違いであり、正解となります。

正解はDですが、今回はAの話に着目します。

3. HTTPリクエストのメソッドの代表例

- HTMLのフォームの例

```
<form action="/kw/sch/course.html" method="get">
    <div class="form-input">
            <input type="text" id="cq1" name="cq" class="search_btn" placeholder="コース名を入力してください">
    </div>
    <input type="image" src="/kw/images/icon/search-white.svg" alt="検索">
</form>
```



3. HTTPリクエストのメソッドの代表例

HTTPメソッドの種類

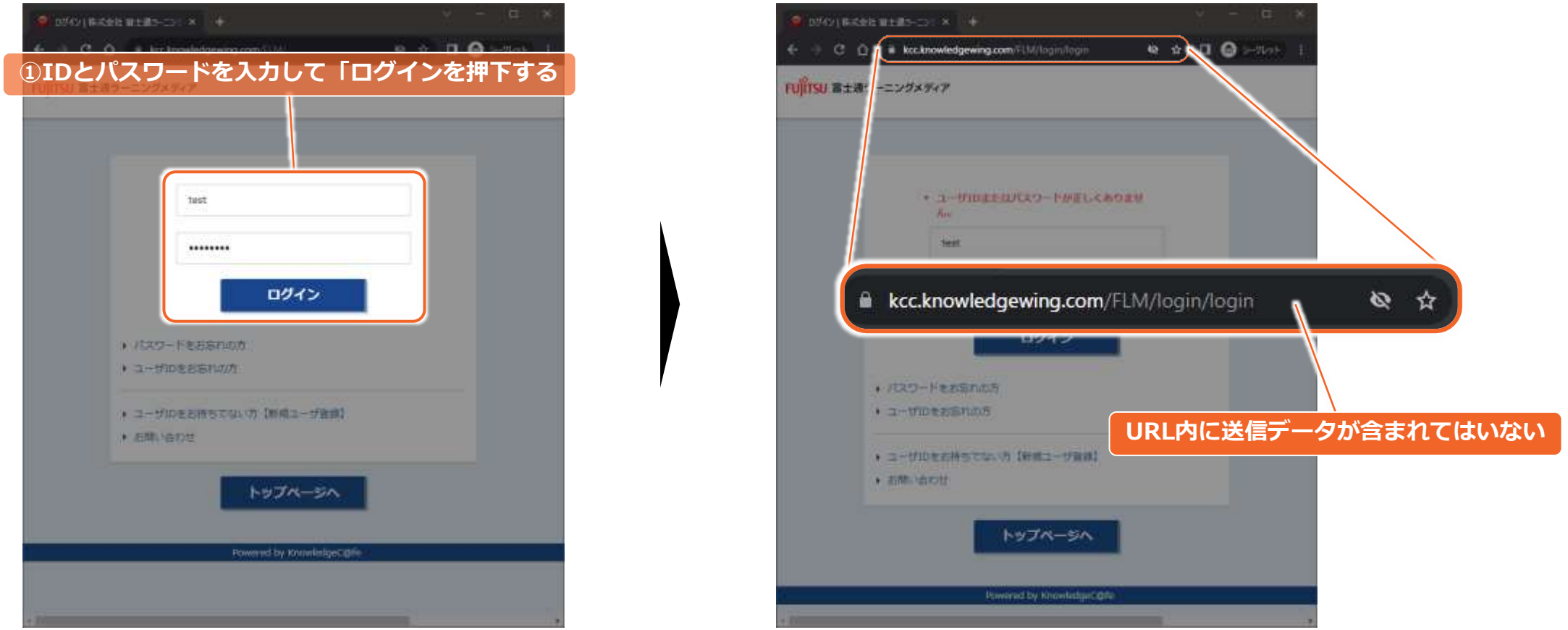
- HTTPメソッドとは、Webサーバーに対する依頼の種類を識別するための情報です。
- 以下の9種類があります。
  - **GET**　　リソースの取得（返信）を要求します。
  - HEAD　　GETメソッド利用時に返信されるHTTPヘッダーを返信するように要求します。
  - **POST**　　Webサーバーにデータを送信します。
  - PUT　　リソースを更新（置換）します。
  - DELETE　　リソースを削除します。
  - CONNECT　　プロキシ環境でのHTTPS通信の開始を要求します。
  - OPTIONS　　利用可能なメソッドの一覧を要求します。
  - TRACE　　疎通確認のために、リクエスト内容をそのまま返信するように要求します。
  - PATCH　　リソースを部分的に変更します。

3. HTTPリクエストのメソッドの代表例

- **GETメソッド**の使用例　https://www.knowledgewing.com/kw/
  - 下図のように、URL内に送信データが含まれます。

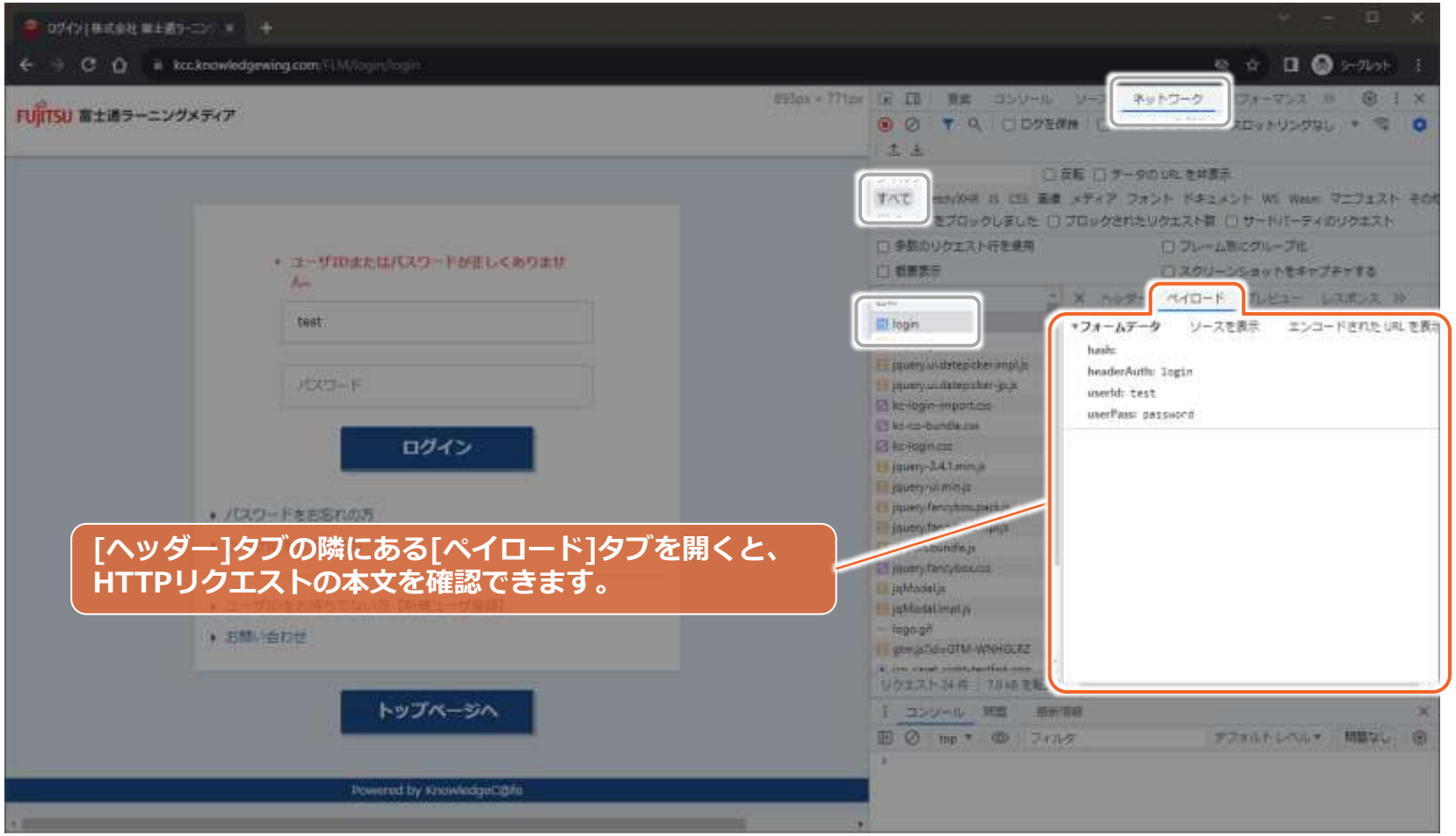
3. HTTPリクエストのメソッドの代表例

- **POSTメソッド**の使用例　https://www.kcc.knowledgewing.com/FLM
  - URL内には送信データが含まれません。



3. HTTPリクエストのメソッドの代表例

- POSTメソッドでは、HTTPリクエストヘッダではなく**本文**に送信データが含まれます。
  - 下図のように、開発者ツールでHTTPリクエストの本文を確認できます。



3. HTTPリクエストのメソッドの代表例

GET vs POST

- アドレスバーを見ることで送信データを確認できるか否かという違いだけでなく、下図のようにHTTPリクエスト内のどの部分に送信データを含めるのかが異なっています。

	GETメソッドのHTTPリクエストの例	POSTメソッドのHTTPリクエストの例
開始行	`GET /kw/sch/course.html?cq=html HTTP/1.1`	`POST /FLM/ HTTP/1.1`

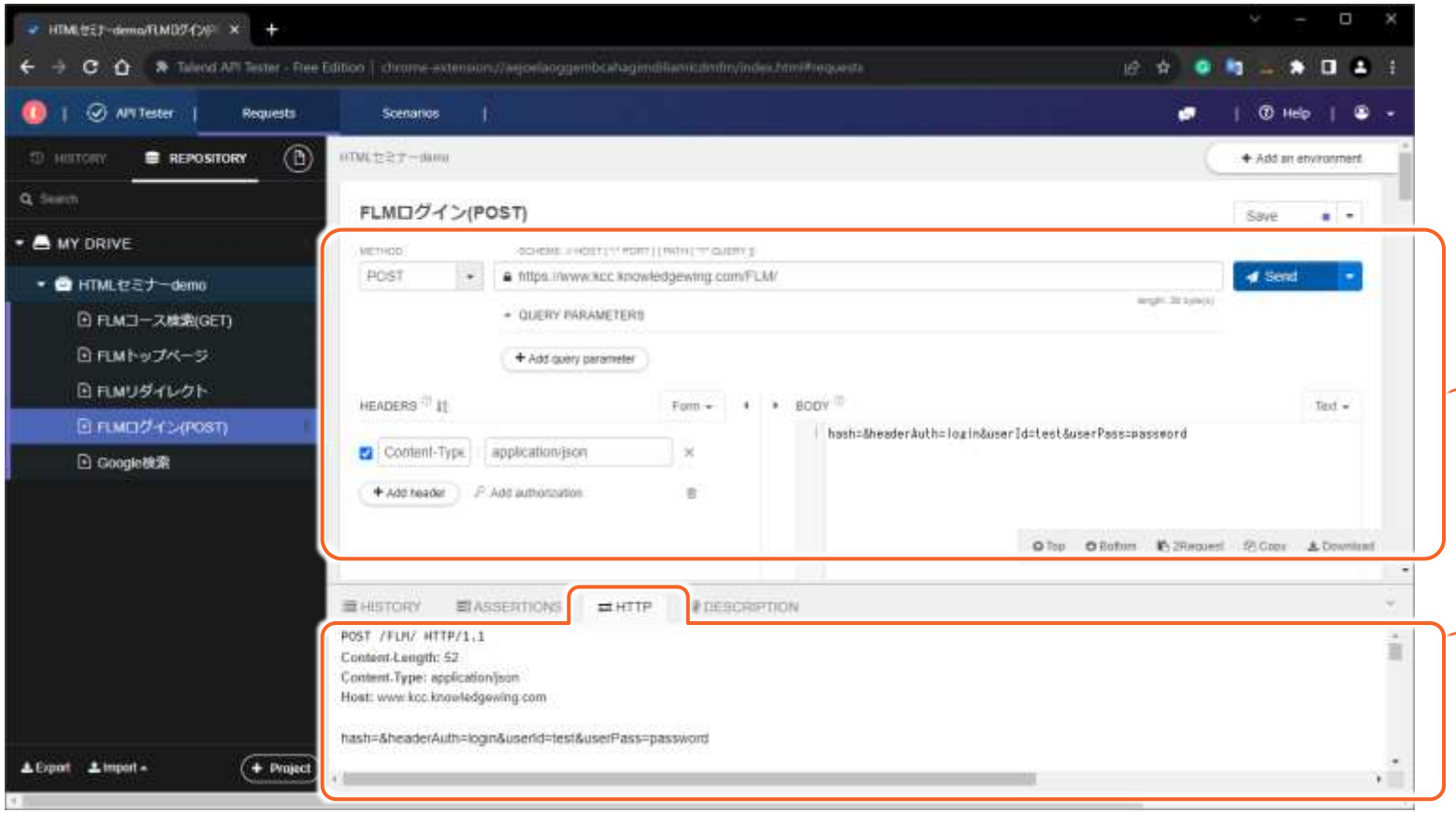
HTTPヘッダー	`Host: www.knowledgewing.com`	`Content-Length: 52`
		`Content-Type: application/json`
		`Host: www.kcc.knowledgewing.com`

本文		`hash=&headerAuth=login&userId=test&userPass=password`

4. HTTPレスポンスのステータスコードの代表例

4. HTTPレスポンスのステータスコードの代表例

- Google Chromeの拡張機能である「Telend API Tester」を利用すると、HTTPリクエストを手動で組み立てて送信できます。また、HTTP通信の中身を確認することもできます。
  - HTTPリクエストだけでなく、HTTPレスポンスも確認できます。
  - 開発者ツールの[ネットワーク]タブの表示とは違い、こちらでは整形前の状態を確認できます。



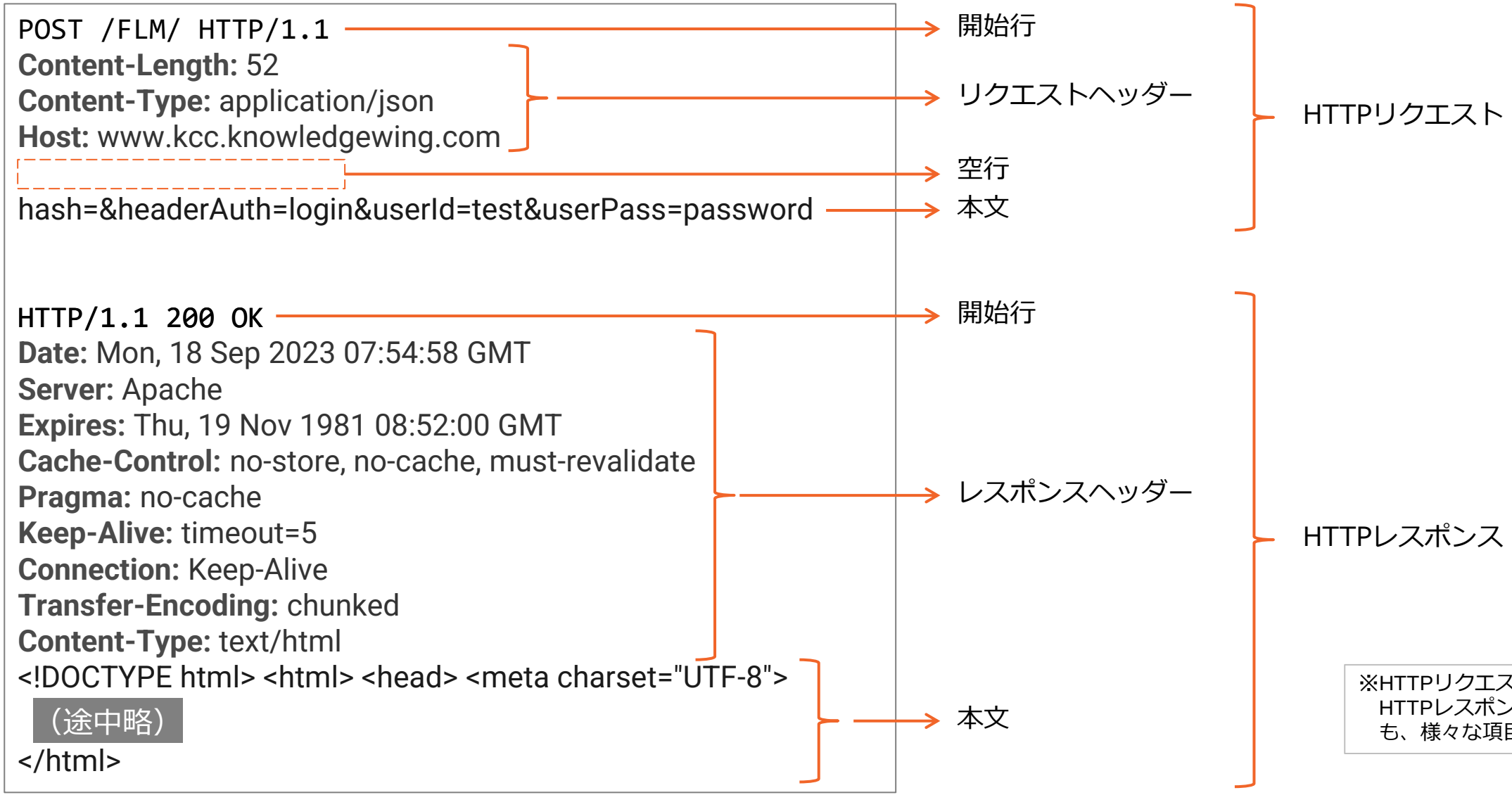
この欄でHTTPリクエストの開始行、ヘッダー、本文をそれぞれ組み立てて、送信できます。

[HTTP]タブにHTTPリクエストとHTTPレスポンスの内容が表示されます。

※ここのスクリーンショットでは、HTTPレスポンスはスクロール下部に見切れていて表示されていません。

4. HTTPレスポンスのステータスコードの代表例

■ 前ページの[HTTP]タブの内容

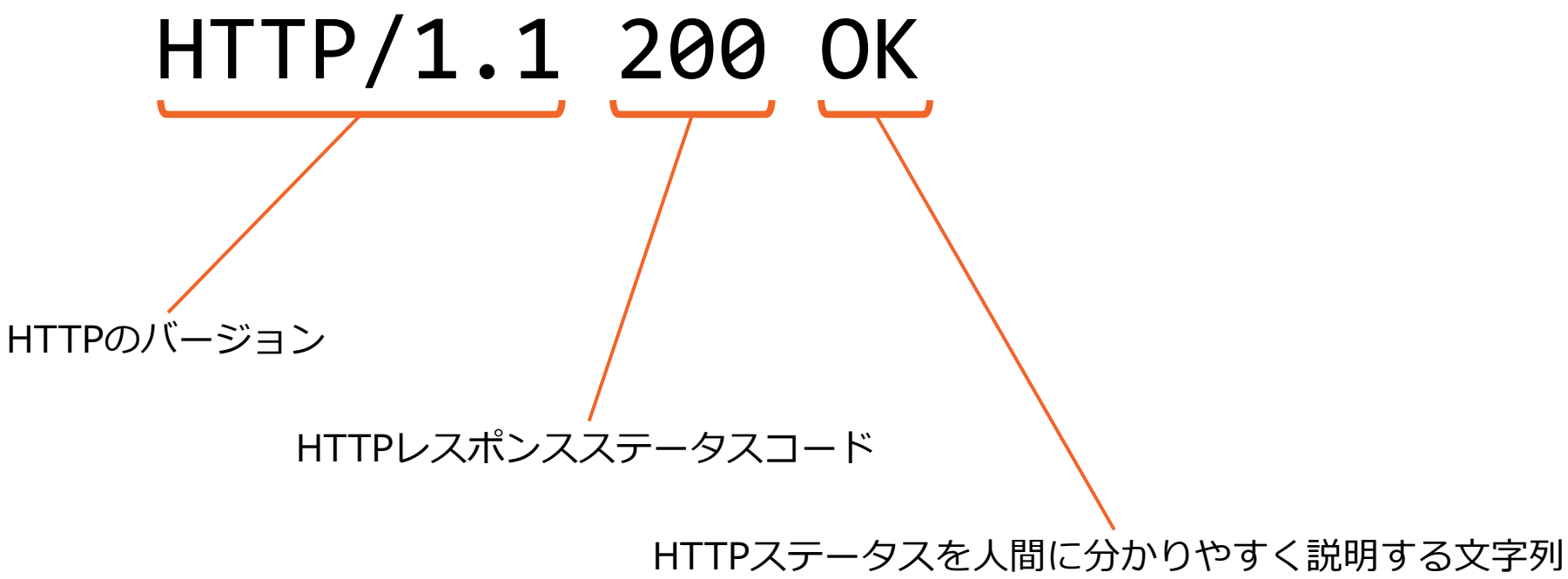


※HTTPリクエストと同様に
HTTPレスポンスのヘッダーに
も、様々な項目があります。

4. HTTPレスポンスのステータスコードの代表例

HTTPレスポンスの開始行

- 下の例のように、下記3点が空白文字で区切られて列挙されている。



4. HTTPレスポンスのステータスコードの代表例

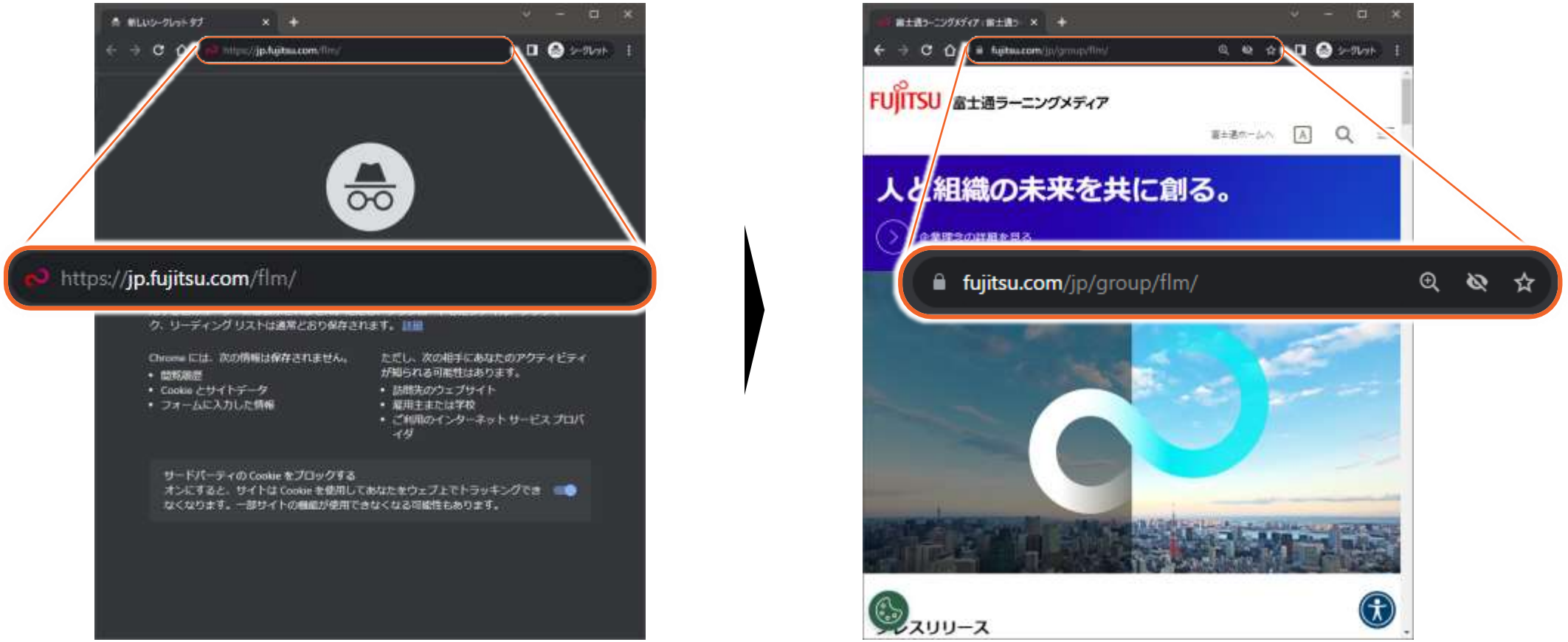
代表的なHTTPレスポンスステータスコードとその意味

- 1xx系 (Informational)： リクエストが受信され、処理継続中である
  - 100 (Continue): サーバー側で問題なく処理を継続中
- 2xx系 (Successful)： リクエストが正常に受信され、受け入れられた
  - 200 (OK)： リクエストを受け取り、処理が成功した。
  - 202 (Accepted)： クエストを受け取ったが、処理が完了していない。
- 3xx系 (Redirection)： リクエストを完了するために更にアクションが必要である
  - 301 (Moved Permanently): リクエストされたリソースが別の場所に完全に移動した。
  - 302 (Found): リクエストされたリソースが別の場所に一時的に移動した。
- 4xx系 (Client Error)： リクエスト内容に不備がある
  - 403 (Forbidden)： 権限がなくリソースへのアクセスを許可できない。
  - 404 (Not Found)： リクエストされたリソースを見つけられない。
- 5xx系 (Server Error): サーバー側でリクエストを処理できなかった
  - 503 (Service Unavailable)： サーバーがリクエストを処理できる状態にない。
  - 504 (Gateway Timeout)： 規定時間内にサーバーからの応答が得られなかった。

4. HTTPレスポンスのステータスコードの代表例

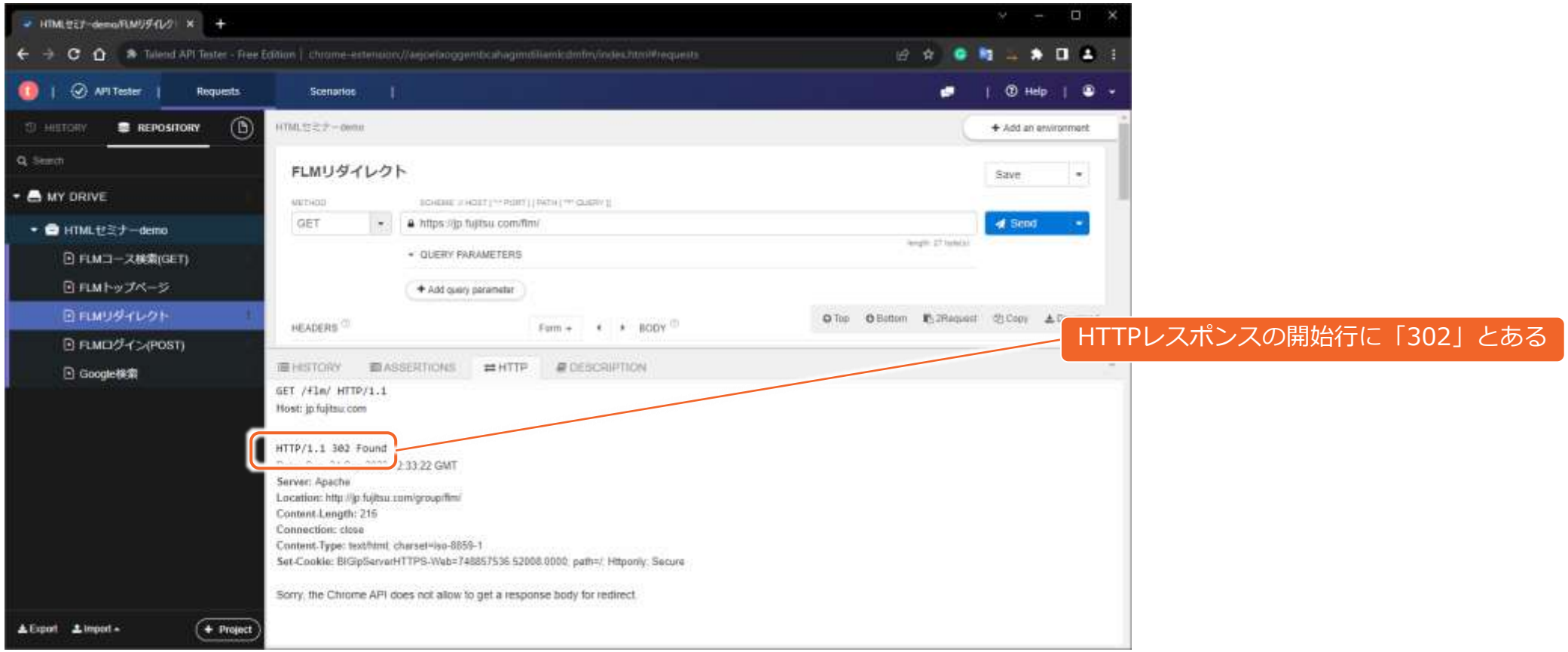
HTTPレスポンスステータスコードが使われている場面

- WebサイトのURLが変更された場合には3xx系のHTTPステータスコードがレスポンスされ、それを受け取ったWebブラウザは新しいURLにリダイレクト（再アクセス）しています。



4. HTTPレスポンスのステータスコードの代表例

- 「Telend API Tester」を利用すると、ステータスコードとして302がレスポンスされていることがわかります。



5. おわりに

参考サイト

- **HTTPの仕様は、下記のサイトに掲載されています。**
  - https://developer.mozilla.org/ja/docs/Web/HTTP
    - 日本語なので分かりやすいです。
  - https://datatracker.ietf.org/doc/html/rfc7231
    - HTTPの仕様が掲載されている本家です。

まとめ

本日のポイントは下記のとおりです。

- クライアント（Webブラウザ）とWebブラウザの間では、HTTPという通信規約に則って要求や応答が連絡されている。
- HTTPリクエストのメソッドごとに、サーバーに要求する行動が異なる。
- HTTPリクエストのメソッドのうち、GETとPOSTでは送信データが含まれる場所が異なっている。
  - GETではリクエストの開始行
  - POSTではリクエストの本文
- HTTPレスポンスのステータスコードは、1桁目の数字でおおよその意味がグループ化されている。
  - 1xx系 (Informational)： リクエストが受信され、処理継続中である
  - 2xx系 (Successful)： リクエストが正常に受信され、受け入れられた
  - 3xx系 (Redirection)： リクエストを完了するために更にアクションが必要である
  - 4xx系 (Client Error)： リクエスト内容に不備がある
  - 5xx系 (Server Error): サーバー側でリクエストを処理できなかった
- リダイレクトは、Webブラウザが3xx系のステータスコードを受け取ることによって起きる。

ご清聴ありがとうございました。

以降、質問タイムです。