HTML5プロフェッショナル認定試験
レベル1ポイント解説無料セミナー

株式会社クリーク・アンド・リバー社 認定講師

高井 歩

本日の概要

- HTML5 レベル1試験について
- HTTP,HTTPSプロトコル
- オフラインWebアプリケーション

HTML5プロフェッショナル認定資格とは

- 次世代のWebプロフェッショナルのスキルの向上に貢献するために、
  HTML5を活用したWebページやWebアプリケーションなどのデザイン、
  設計、構築に関する体系だった知識とスキルを備えたHTML5のプロフェッ
  ショナルを中立的な立場で公平かつ厳正に認定する資格制度です。

- Webデザイナー、Webプログラマー、スマートフォンアプリ開発者、
  サーバーサイドエンジニアなどの、Web開発プロジェクトやWebサービス
  に関わるあらゆるプロフェッショナルが対象です。

- 多くの企業が推進する次世代Web言語の認定資格として、
  HTML5のプロフェッショナルのスキルの向上に役立ちます。
  また、企業内や研修機関での『技術力を担保する客観的基準』としても
  活用できます。

二つのレベル

HTML5 Level.1

マルチデバイスに対応した静的なWebコンテンツをHTML5を使ってデザイン・作成できる。

対象

- Webデザイナー/HTMLコーダー
- Webディレクター/グラフィックデザイナー
- フロントエンドプログラマー
- Webシステム開発者
- スマートフォンアプリ開発者
- サーバーサイドエンジニア

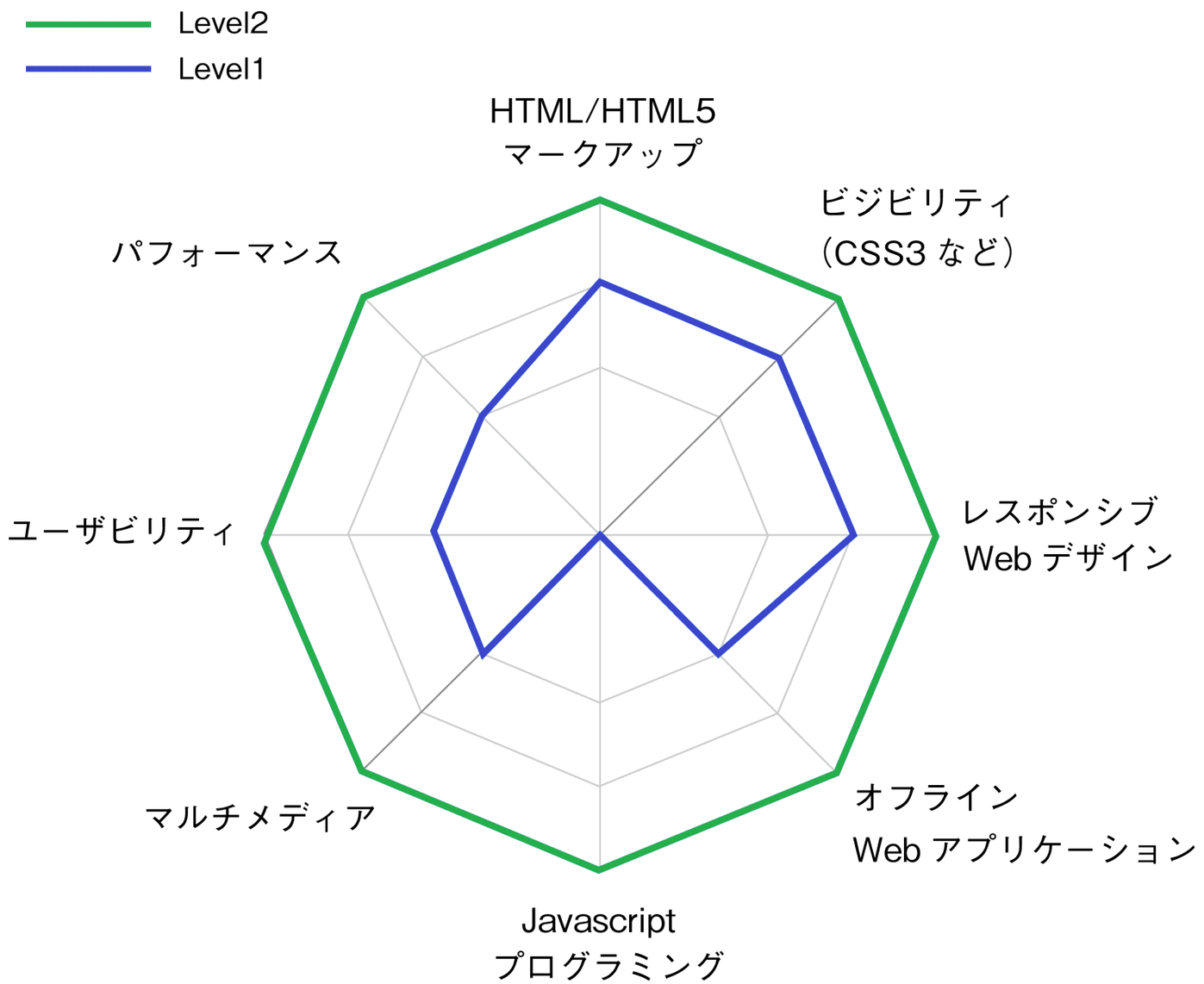
HTML5 Level.2

システム間連携や最新のマルチメディア技術に対応したWebアプリケーションや動的Webコンテンツの開発・設計ができる。

対象

- Webシステム開発者
- スマートフォンアプリ開発者
- フロントエンドプログラマー
- Webディレクター
- サーバーサイドエンジニア
- Webデザイナー/HTMLコーダー

レベル1とレベル2の資格体系



HTML/HTML5マークアップ
HTML5に関するタグの用途、構造の組み立て方に関する技術

ビジビリティ
JavascriptやCSS3などを用いて、デザイン仕様に沿った見やすい表示を行うための技術

レスポンシブWebデザイン
一つのソースで、スマートフォンなどの様々なデバイスの画面サイズに対応させるための技術

オフラインWebアプリケーション
通信が常時接続状態ではない環境でも、効率的にWebコンテンツを動作させるための技術

Javascriptプログラミング
Javascriptを使って、動的なWebコンテンツを作成する技術

マルチメディア
３D・動画・音声ファイルなどのマルチメディアコンテンツの表示・再生に関する技術

ユーザビリティ
ナビゲーション、地図表示など操作しやすいコンテンツを作成するための技術

パフォーマンス
データベースや、並列処理を使ってコンテンツを効率良く高速に動作させるための技術

レベル1とレベル2の資格体系

ベーシックレベル
HTML5プロフェッショナル向け

所要時間：90分
試験問題数：約60問
受験料：\15,000 （税抜）
認定条件： HTML5 レベル1試験に合格すること
認定の有意性の期限：5年間

認定名： HTML5 Level1 (Markup Professional)
試験名： HTML5 Level1 Exam

この資格の認定者は、下記のスキルと知識を持つWebプロフェッショナルであることを証明できます。

- **HTML5を使って静的なWebコンテンツを作成することができる。**
- **ユーザビリティ・ビジビリティの高いWEBコンテンツを設計・作成することができる。**
- **スマートフォンや車載システムなど、様々なデバイスに対応したコンテンツ作成ができる。**

アドバンストレベル
HTML5プロフェッショナル向け

所要時間：90分
試験問題数：40〜45問
受験料：15,000
認定条件：HTML5 レベル2試験に合格し、かつ有意なHTML5レベル1認定を保有していること。
認定の有意性の期限：5年間

認定名： HTML5 Level2 (Application Development Professional)
試験名： HTML5 Level2 Exam

この資格の認定者は、下記のスキルと知識を持つWebプロフェッショナルであることを証明できます。

- **動的に動作させて高いユーザビリティを実現するリッチユーザインターフェイスアプリケーションを作成することができる。**
- **マルチデバイスに対応し高パフォーマンスで動作する動的コンテンツを作成することができる。**
- **システム間連携を行いリアルタイムな情報を提供するアプリケーションを作成することができる。**

出題範囲

主題	項目
Webの基礎知識	・HTTP,HTTPSプロトコル(☆7) ・HTMLの書式(☆8) ・ネットワーク・サーバ関連技術の概要(☆6) ・Web関連技術の概要(☆6)
CSS3	・スタイルシートの基本(☆6) ・CSSデザイン(☆9) ・カスケード（優先順位）(☆2)
要素	・HTML4.01以前の要素および属性(☆7) ・HTML5で新しく加わった要素および属性(☆10) ・HTML5で廃止されたタグ(☆5)
レスポンシブWebデザイン	・マルチデバイス対応ページの作成(☆4) ・メディアクエリ(☆4) ・スマートフォンサイト最適化(☆3)
オフラインWebアプリケーション	・オフラインWebアプリケーション(☆2)

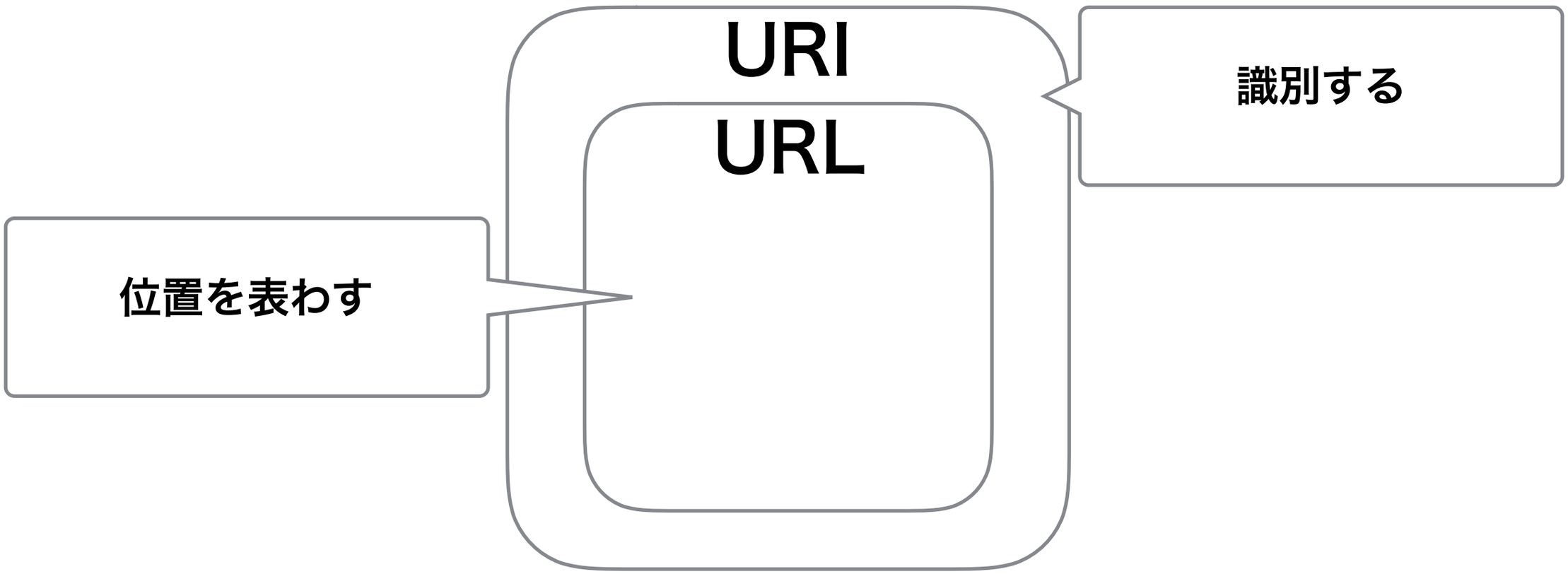
HTTP,HTTPSプロトコル

知識問題 記述問題

URIとURL

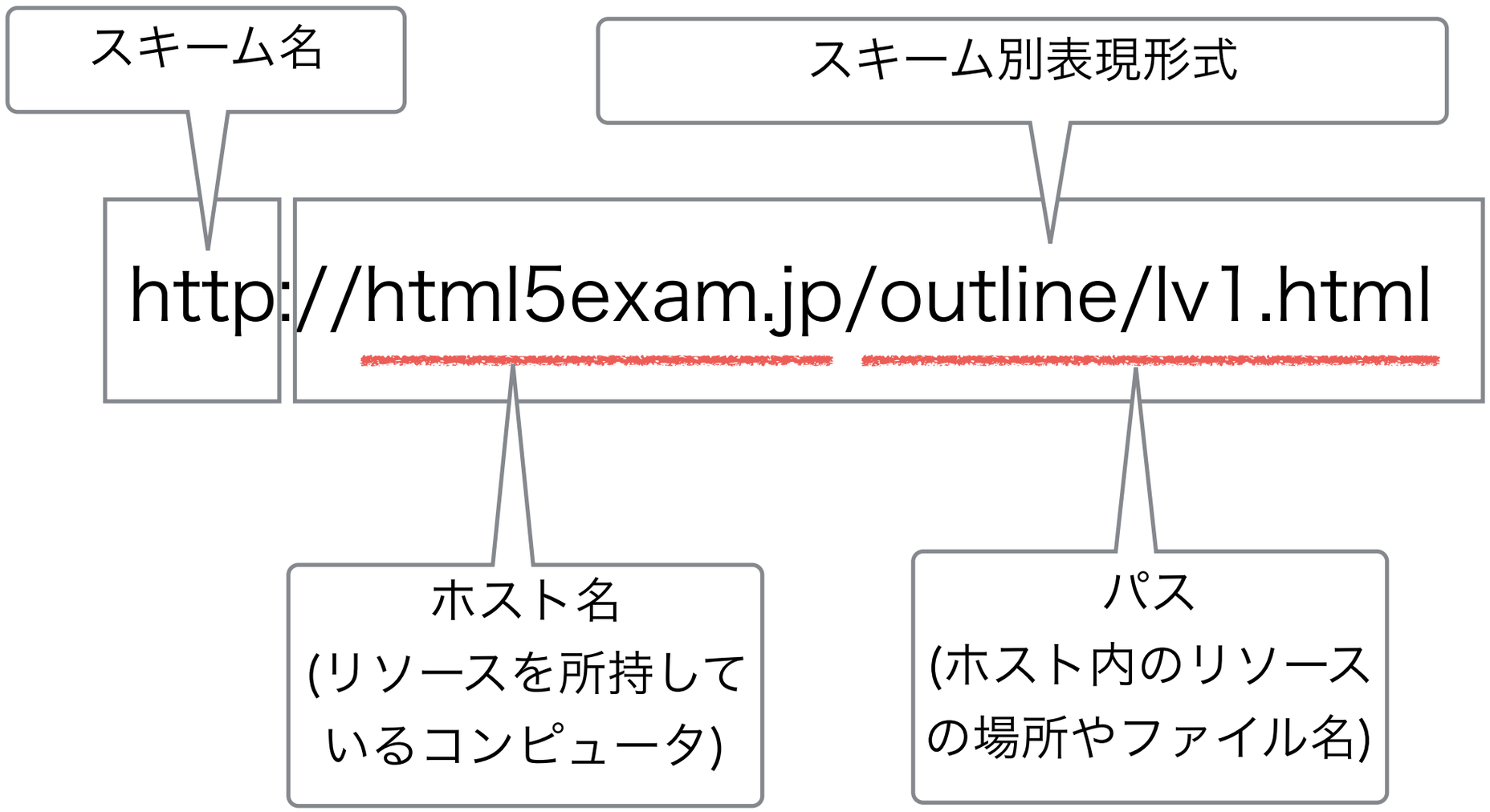
- Webページ(や画像などのリソース)のネットワーク上での位置を表すのがURL(Uniform Resource Locator)です。

- URLはURI(Uniform Resource Identifier)というリソースを識別するための記法の一種です。



URL

URLは以下のような形式になっています。



URI,URLで使用できる文字

非予約文字	予約文字	
A-Z	!	+
a-z	#	,
0-9	$	/
- (ハイフン)	&	:
_ (アンダーバー)	'	;
. (ピリオド)	(	=
~(チルダ)	)	?
	*	@
	+	[
	,	]

非予約文字：ホスト名、ファイル名などに自由に使用できる

予約文字：区切り文字など特定目的のみ使用できる

URLエンコーディング

- 半角スペースや日本語の文字などは、URLにそのまま含めることはできません。

- そのため、%に続けて16進数二桁で文字コードを指定する、**URLエンコード**(パーセントエンコード)を使って記述します。

- 例えば、半角スペースは%20,ひらがなの'あ'は%82%A0になります。

HTTP

HTTP

http://example.com

URLの先頭にあるコレ

Hyper Text Transport Protocol

簡単に言うと、Webページ(ハイパーテキスト)を
WebブラウザとWebサーバの間で送受信するための通信規約
規約に則っていれば、どのWebブラウザとWebサーバの
組み合わせでも通信できます

HTTPの基本

- リクエスト/レスポンス
- HTTPメッセージ
- TCP 80番ポート
- ステートレス
- 通信経路の暗号化はされていない

リクエスト/レスポンス

http://www.example.com/index.html
を下さい

リクエスト

レスポンス

Webクライアント
(Webブラウザ)

index.html

Webサーバ

1対1の関係

HTTPメッセージ

- リクエスト/レスポンスは両方共、**HTTPメッセージ**という形式でデータをやりとりします。
- HTTPメッセージは、**リクエストライン**または**ステータスライン**と、**メッセージヘッダ**及び**メッセージボディー**から構成されます。

レスポンスメッセージの例

リクエストライン

- リクエストメッセージの先頭には、**リクエストライン(リクエスト行)**が含まれます。

- リクエストラインは以下の書式で記述されます。

  メソッド パス名 HTTP/バージョン

  例) GET / HTTP/1.1

- リクエストラインによって、リクエストの目的(メソッド)、対象のWebページ(またはリソース)を指定します。

リクエストメソッド

メソッドはリクエストの目的(種類)を表わします。

メソッド	概要
GET	**Webページの素材(HTMLファイルなど)を要求します。**
HEAD	データ本体ではなくヘッダのみを要求します。
POST	**データをWebサーバに送信します。**
PUT	ファイルをWebサーバに転送します。
DELETE	Webサーバ上の指定したリソースの削除を要求します。
CONNECT	プロキシ接続への変更を要求します。
OPTIONS	指定したURIで使用できるオプションの一覧を要求します。
TRACE	接続試験用で、Webサーバは送られた内容を返送します。

ステータスライン

- リクエストライン同様にレスポンスメッセージの先頭に存在するのが**ステータスライン(ステータス行)**です。

- レスポンスラインは以下の書式で記述されます。

  HTTP/バージョン ステータスコード 補足メッセージ

  例) HTTP/1.1 200 OK

- レスポンスラインによって、リクエストの結果を取得することができます。
  (リソースそのものはメッセージボディーに格納されています)

ステータスコード

レスポンスに含まれる3桁のコード

まず大分類を覚える

1XX	情報
2XX	**成功**
3XX	リダイレクション
4XX	**クライアントエラー**
5XX	**サーバエラー**

次によく使われるコードを覚える

200	**OK**
301	恒久的に移動した
302	一時的な移動
303	他を参照
401	認証が必要
403	アクセス権限なし
404	**リソースが存在しない**
500	**サーバ内部のエラー**
503	メンテナンスや過負荷

各Webブラウザのデベロッパーツール内のネットワーク(Firefox,Chrome),タイムライン(Safari)などで確認できます。

練習問題1

ステータスコードと内容の組み合わせで正しいものを2つ選んでください。

A. 200 OK.

B. 100 NG.

C. 301 Forbidden

D. 404 Not Found.

E. 500 External Server Error.

HTTPヘッダ

リクエストメッセージ・ヘッダの例

Host: html5exam.jp
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:46.0)(省略)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ja,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Cookie: _ga=GA1.2.1885427586.1438121009; _gat=1
Connection: keep-alive
Cache-Control: max-ge=0
(改行)
~~~ メッセージ・ボディー ~~~

ヘッダ項目は、項目名:(スペース)値 の形で列挙します。
~~~

HTTPメッセージを確認する方法

Webブラウザの開発者ツールから「ネットワーク」タブを選択し、確認したいリソースを選択する

Firefoxの例(生ヘッダ表示)

HTTP Header Fields

HTTPヘッダに含まれる項目は大きく4つに分類できます。

- **General Header Fields**
  (リクエスト/レスポンス双方に含まれる項目)

- Request Header Fields
  (リクエストにのみ含まれる項目)

- Response Header Fields
  (レスポンスにのみ含まれる項目)

- Entity Header Fields
  (リクエスト/レスポンスのコンテンツに関する項目)

General Header Fields

リクエスト/レスポンス共に使用される項目

項目	概要
Cache-Control	**プロキシやクライアントへのキャッシュに関する指示 no-cacheでキャッシュさせない**
Connection	HTTP/1.1の持続接続機能(KeepAlive)の通知
Date	時刻
Pragma	様々な用途で使用
Trailer	ヘッダ項目をコンテンツの後ろに付ける場合に指定
Transfer-Encoding	転送に使用するデータエンコード
Upgrade	別のプロトコルを推奨する場合に使用
Via	メッセージの転送経路
Warning	ステータス行に付加される警告とメッセージ

Request Header Fields

リクエストヘッダに使用される項目(一部)

項目	概要
Accept	**HTTPクライアントが処理できるメディアの種類**
Authorization	**Web認証のための情報**
Host	リクエストラインのパスのホスト
Proxy-Authorization	プロクシのための認証情報
Referer	リクエスト中URIのリンク元
User-Agent	HTTPクライアント情報

Response Header Fields

レスポンスヘッダのみに使用される項目

項目	概要
Accept-Ranges	Rangeヘッダ項目で使用できる単位(現状bytesのみ)
Age	コンテンツが生成されてからの予測経過時間
ETag	コンテンツのバージョンを表す識別子
Location	**リダイレクト先(絶対URLで指定)**
Proxy-Authenticate	プロキシサーバとクライアント間で認証が必要
Retry-After	指定秒数後に再度要求を求む
Server	サーバ情報
Vary	Accept,Accept-Charset,Accpep-Languageなどの情報
WWW-Authenticate	認証の必要性を指示

Entity Header Fields

リクエスト/レスポンスのコンテンツに使用される項目(一部)

項目	概要
Allow	使用可能なメソッドの一覧(GET,PUTなど)
Content-Encoding	コンテンツのエンコード方式
Content-Language	コンテンツの言語
Content-Length	**コンテンツの長さ(バイト)**
Content-Location	コンテンツの別のURL
Content-MD5	コンテンツ破損/改変チェック用データ
Content-Range	コンテンツの範囲(コンテンツ一部を送信する場合)
Content-Type	**コンテンツの種別(MIMEタイプ)**
Expires	**コンテンツの有効期間(期限切れはキャッシュから削除)**
Last-Modified	**コンテンツが最後に更新された時刻**

リダイレクト

- リクエストされたWebページから、別のWebページへ自動的に転送させることを**リダイレクト**と呼びます。

- HTTPヘッダ、HTMLのmetaタグ、JavaScriptによってリダイレクトさせることが出来ます。

- HTTPヘッダによるリダイレクトでは、ステータスコード3XXを返し、**Location: ヘッダフィールド**で転送先を指定します。

練習問題2

リダイレクトのために設定するHTTPメッセージの項目を選択してください。

A. レスポンスラインのRedirect:

B. リクエストヘッダのLocation:

C. レスポンスヘッダのLocation:

D. レスポンスボディーのRedirect:

E. レスポンスヘッダのRedirect:

TCP 80番ポート

HTTPはトランスポート層の**TCPによって通信を行ないます。**

0

53442

65535

送信側は未使用ポート(49152以上)から**ランダムで選択**されます。

0

80

65535

HTTPサーバは原則80番で待受けます。**0〜1023**迄は用途の決っている**ウェルノウンポート**とされています

認証

- HTTPでは、リソースへのアクセスに**認証**を要求することができます。**認証に失敗するとリソースを取得できません。**

- 出題範囲には**Basic認証**と**Digest認証**が含まれています。

- **Basic認証では、ID/パスワードは暗号化されずに送信されます。Digest認証では、パスワードがそのまま送信されることはありません。**

- 開発中のWebサイトを関係者にのみアクセスを許可する場合などに良く使用されています。

Basic認証

リクエスト

レスポンス
ステータスコード401
(Authorization Required)

リクエスト

Authorization: Basic [Base64エンコードしたID/Pass]

レスポンス

認証成功ならステータスコード200
認証失敗ならステータスコード401

暗号化されていないことに注意

Digest認証

チャレンジコードとパスワードをもとに、レスポンスコードを算出(**非可逆**)

リクエスト

レスポンス

ステータスコード401

チャレンジコード(nonce)

サーバ側で算出したレスポンスコードと一致すれば認証成功

リクエスト

Authorization: Digest

username="ABC",[チャレンジコード],[**レスポンスコード**]

レスポンスコードからパスワードを推測することは難しい

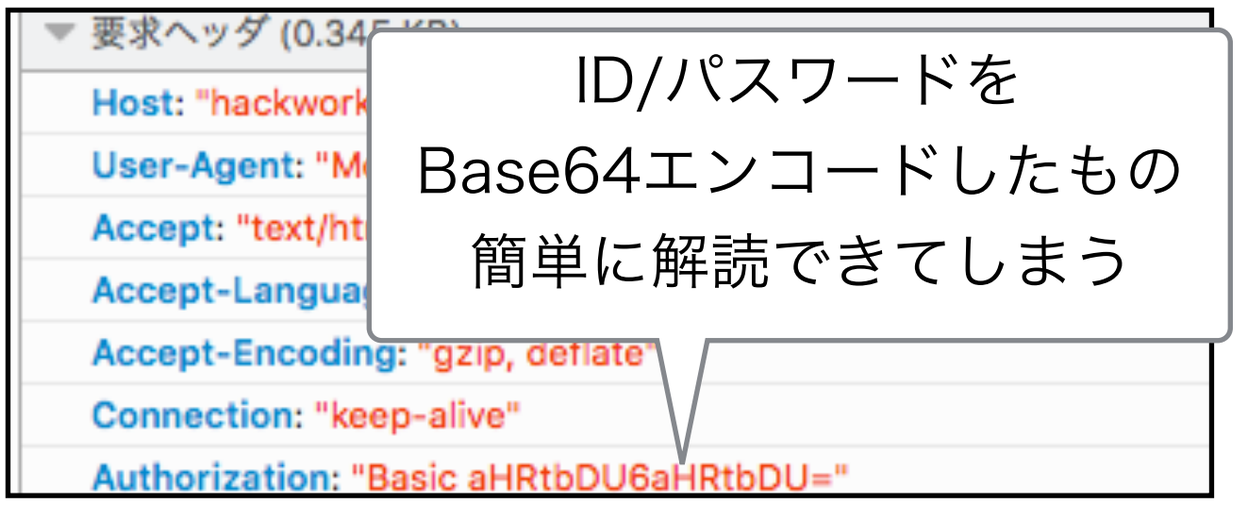
レスポンス

認証成功ならステータスコード200

認証失敗ならステータスコード401

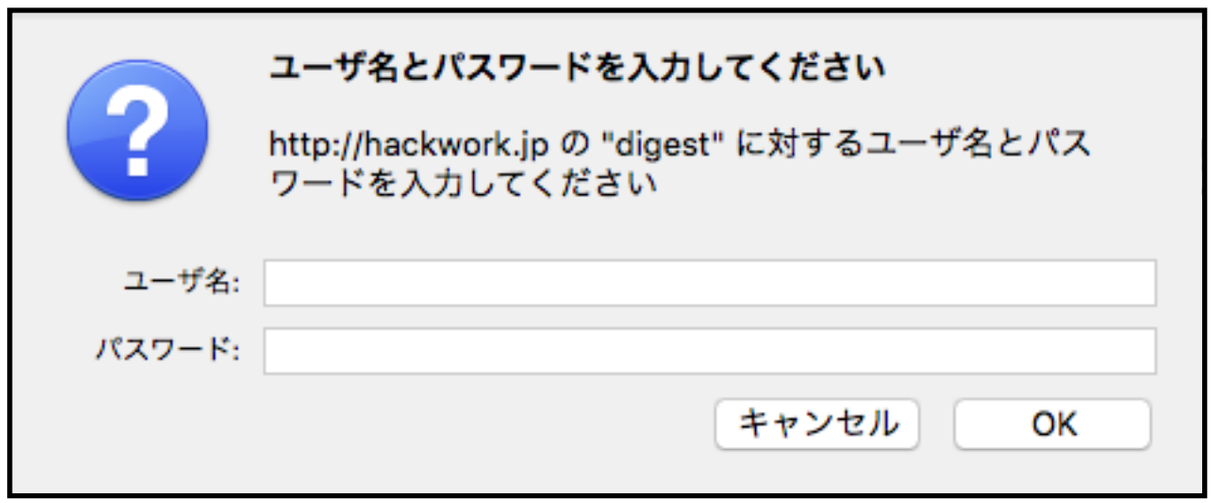
認証実演

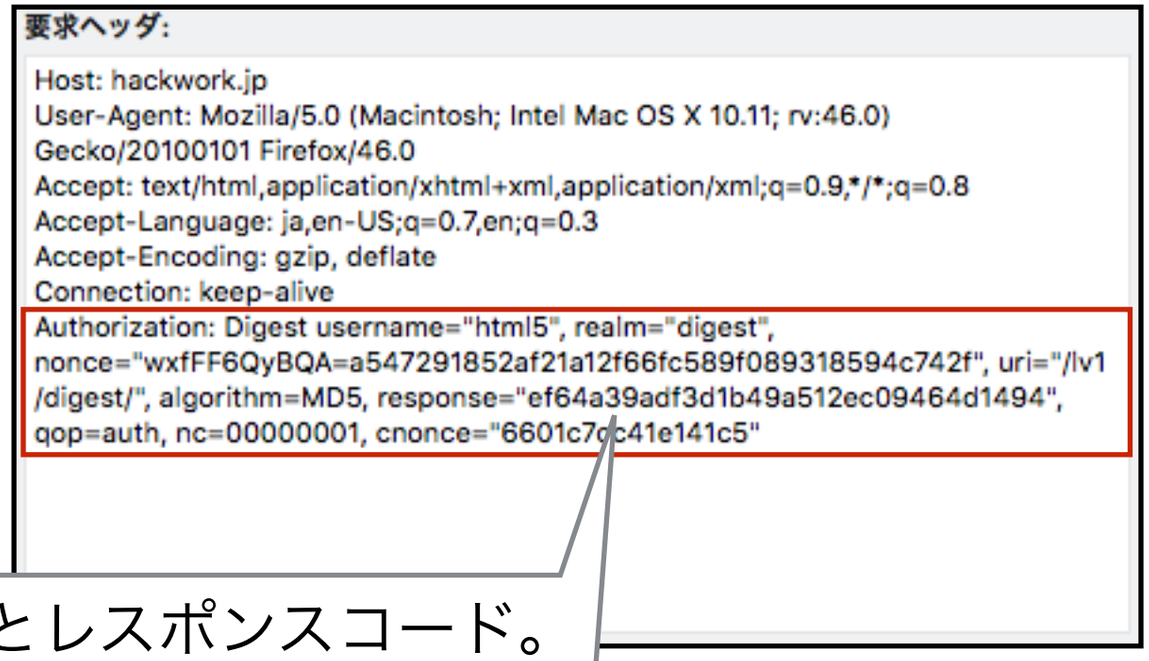
Basic認証



ID/パスワードを
Base64エンコードしたもの
簡単に解読できてしまう

Digest認証





チャレンジコードとレスポンスコード。
解読はそれなりに難しい。

練習問題3

ID/パスワードが平文のまま送信される認証を選択してください。

A. SSLクライアント認証

B. Basic認証

C. Digest認証

D. Kerberos認証

E. Fortran認証

ステートレス

- HTTPは一回のリクエスト/レスポンスが完了すると、接続を切断します。そのため、接続情報(ステート)を維持できないため**ステートレス**な接続と呼ばれています。

- 電話のように、会話をしている間相手が変化せず、接続情報を維持できる通信を**ステートフル**な接続と言います。

- HTTPでも、Cookieやサーバ側プログラムによるセッション管理を使うことによりステートフルな通信を行なうことができます。

HTTP Cookie

WebサイトやJavaScirptが、HTTPクライアント(Webブラウザ)にデータを一時的に保存する仕組み

リクエスト

レスポンス

Set-Cookie: name=value;
expires=日付; path=パス;
domain=クッキーが有効なドメイン；

リクエストヘッダに
Cookie情報を記述する

リクエスト

Cookie: name=value;

レスポンス

レスポンスヘッダに
セットするCookie情報を記述する

練習問題4

Cookieについて正しい説明を選択してください

A. value=nameの組み合わせで登録する

B. 設定されたCookieはどのサーバでも読みとれる

C. Webサーバが設定することはない

D. Cookieに期限はない

E. リクエストメッセージに含まれる

HTTPS

HTTPの危険性

- ネットワークを流れる情報を盗聴するツールをSnifferと呼びます。

- Snifferを使うと、簡単にネットワーク上の通信内容を取得できます。

- 特にHTTPやBasic認証では、情報が暗号化されずに送受信されるため、通信内容が丸わかりです。

- また、通信相手(Webサーバ)が本当に正しいのか(なりすましを検知できるか)、確認する手段がありません。

HTTPS

- 基本的な仕組み(メッセージやステートレス、認証)はHTTPと同じ

- **TCP 443番ポート**

- **SSL/TLS**を使用して**通信経路の暗号化、通信先Webサーバの証明**

- Googleも検索ランキングの指標として、SSL/TLSの導入の有無を使用することを検討

- 業界のトレンドとして、全てのWebサイトでHTTPSへと切り替える流れ

SSL/TLS

- **SSL(Secure Socket Layer)**
- **TLS(Transport Layer Security)**
- TCPのかわりに使用できる暗号化通信の仕組み
- **電子証明書**を使って相手先を確認できます
- SSLの後継がTLSですが、総称してSSLと呼ばれることが多いです。(もしくはSSL/TLSと表記)
- SSLおよびTLSの古いバージョンは脆弱性が発見されているため使用が非推奨です

SSL/TLSの証明書の種類

発行には企業について厳格な審査が必要。
アドレスバーの表示などで他の認証と区別できる。

- EV(Extended Validation)

日本ジオトラスト株式会社 (JP)

発行には登記簿の審査などが必要。
企業の公式サイトであることを証明できる。

- 企業実在認証

ドメインおよびホストの正当性を証明
ただし悪意のあるサイトも証明書は発行できてしまう。

- ドメイン認証

- オレオレ証明書(自己署名証明書)

ユーザがブラウザに証明書を登録すると表示

自分のサーバが認証局の役割をするため、
無料で証明書を発行できるが、信頼性はまったくない。
~~SSL/TLSを導入するサイトの実験に使うことも。~~

※アイコン表示はFirefoxの場合

時事ネタ: Let's Encrypt

- Let's Encryptとは,電子フロンティア財団(EFF)が提供する、電子証明書の無料配布サービス

- MozillaやCisco,Akamaiなどのインターネット関連企業が協力

- ドメイン証明書のみを提供

- 証明書の発行や更新の自動化ツールも提供するため、証明書発行のための専門知識が不要になる(LinuxやApacheなどの基本的な操作がわかればOK)

Let's Encryptの導入実演

証明書の導入手順

1. CSR(署名リクエストの作成)
2. 認証局へ申請
3. 証明書の発行
4. Webサーバへ証明書をインストール

Let's Encrypt では、
専用ツールにより自動化

※一般的な証明書の導入について、
詳しくはLPIC Level.3 303試験などを参照してください

Let's Encryptの問題点

- Webサーバに対して設定を行なうための権限が無いと導入できない
  (レンタルサーバによっては専用のSSL導入サービスがあるのでそちらを検討したほうが良い場合も)

- 専用ツールをインストールするので、脆弱性がある可能性も0ではない

オフラインWeb

アプリケーション

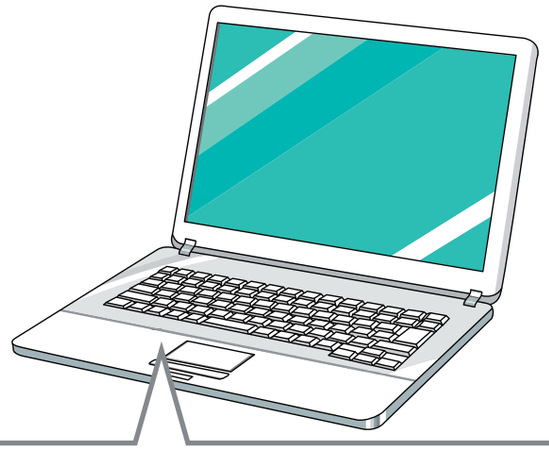
知識問題 コードリーティング問題 記述問題

オフラインWebアプリケーション

- Webサーバに接続できない状況でも、Webページを表示できるよう、**ローカルコンピュータに各種ファイルをキャッシュする**仕組み
- キャッシュするファイルのリストを**マニフェストファイル**と呼ばれるファイルに記述し、Webサーバに設置する

オフラインWebアプリケーションの動作

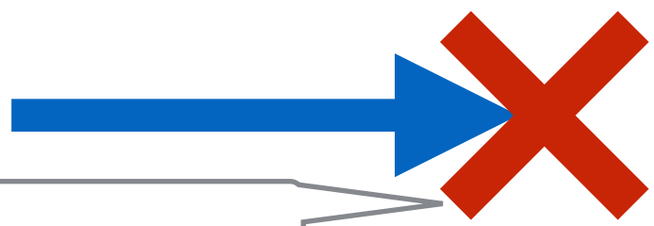
初回アクセス



コンテンツと
マニフェストファイル

マニフェストファイルで指定されたファイルは永続的にキャッシュする

2回目以降のアクセス



Webサーバに接続できなければキャッシュされたファイルを使用する

マニフェストファイルが更新されていれば、マニフェストファイルとリソースをダウンロードする。
更新されていなければキャッシュしたファイルを使用する。

メリット

- 更新が無ければ毎回リソースをダウンロードする必要が無いため、表示の高速化やネットワーク帯域の節約につながる。
- クライアントがネットワーク接続していなくても、アプリケーションを使用できる。

デメリット

- 更新を即座に反映できない

マニフェストファイル

マニフェストファイルの例

```
CACHE MANIFEST

CACHE:
index.html
map.html

NETWORK:
list.php

FALLBACK:
img1.png alt1.png
img2.png alt2.png .
```

- マニフェストファイルは、先頭に"CACHE MANIFEST"の一行を記述する必要があります。
- CACHE,NETWORK,FALLBACKの三つのセクションがあり、それぞれにファイル名を記述します。
- CACHE(キャッシュするファイル)
- NETWORK(キャッシュしない)
- FALLBACK(オフライン時の代替)

manifest属性

- HTMLの**htmlタグのmanifest属性**にマニフェストファイルを指定(絶対URLまたは相対URL)することで、オフラインWebアプリケーションが有効になります。

- 記述例

```
<!doctype html>
<html manifest="sample.appcache">
<head>
...省略...
```

マニフェストファイルの設置

- オフラインWebアプリケーションを有効化するには、まずWebサーバ上にマニフェストファイルを設置します。

- 次にWebサーバの設定を変更し、マニフェストファイルの拡張子に**MIMEタイプ(text/cache-manifest)**を関連付けます。
(Webサーバが**Apache**の場合、**.htaccessファイル**などに記述します)

- 最後にhtmlファイルのhtmlタグのmanifest属性を設定します。

キャッシュデータの更新

- Webサーバ上のキャッシュ対象のリソースファイルだけではなく、**マニフェストファイルも更新する**必要があります。

- マニフェストファイル内では、**#以降はコメント**になるため、コメントに更新日時などを記述することで、簡単にマニフェストファイルを更新できます。

- Webブラウザによっては、単にリロードするだけではなく、JavaScriptによる明示的な更新が必要になるものもあるようです。

キャッシュデータの確認と削除

- Firefoxの場合
  about:preferences#advanced にアクセス
  ネットワークータブのオフラインWebページとユーザデータ

- Chromeの場合
  chrome://appcache-internals/ にアクセス

- Safariの場合
  「開発メニュー」ー「キャッシュを空にする」

- Internet Explorer
  「インターネットオプション」ー「全般」ー「閲覧の履歴」ー「設定」ボタンー「キャッシュおよびデータベース」

練習問題5

マニフェストファイルのセクションとして有効な組み合わせを選択してください

A. CACHE, NETWORK, FALLBACK

B. CACHE, NETWORK, FALLDOWN

C. CATCH, INTERNET, FALLBACK

D. CASH, NOCASH, ALIAS

E. CACHE,NOCACHE, FALLBACK

本日の概要

- HTML5 レベル1 試験について
- HTTP,HTTPSプロトコル
- オフラインWebアプリケーション

練習問題正解

- 練習問題1: A,D
- 練習問題2: C
- 練習問題3: B
- 練習問題4: E
- 練習問題5: A