HTML5プロフェッショナル認定試験 レベル2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

本日の内容

- 試験概要
- ストレージ
  - Web Strage
  - Indexed Database API
  - File API
- テスト・デバッグ

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

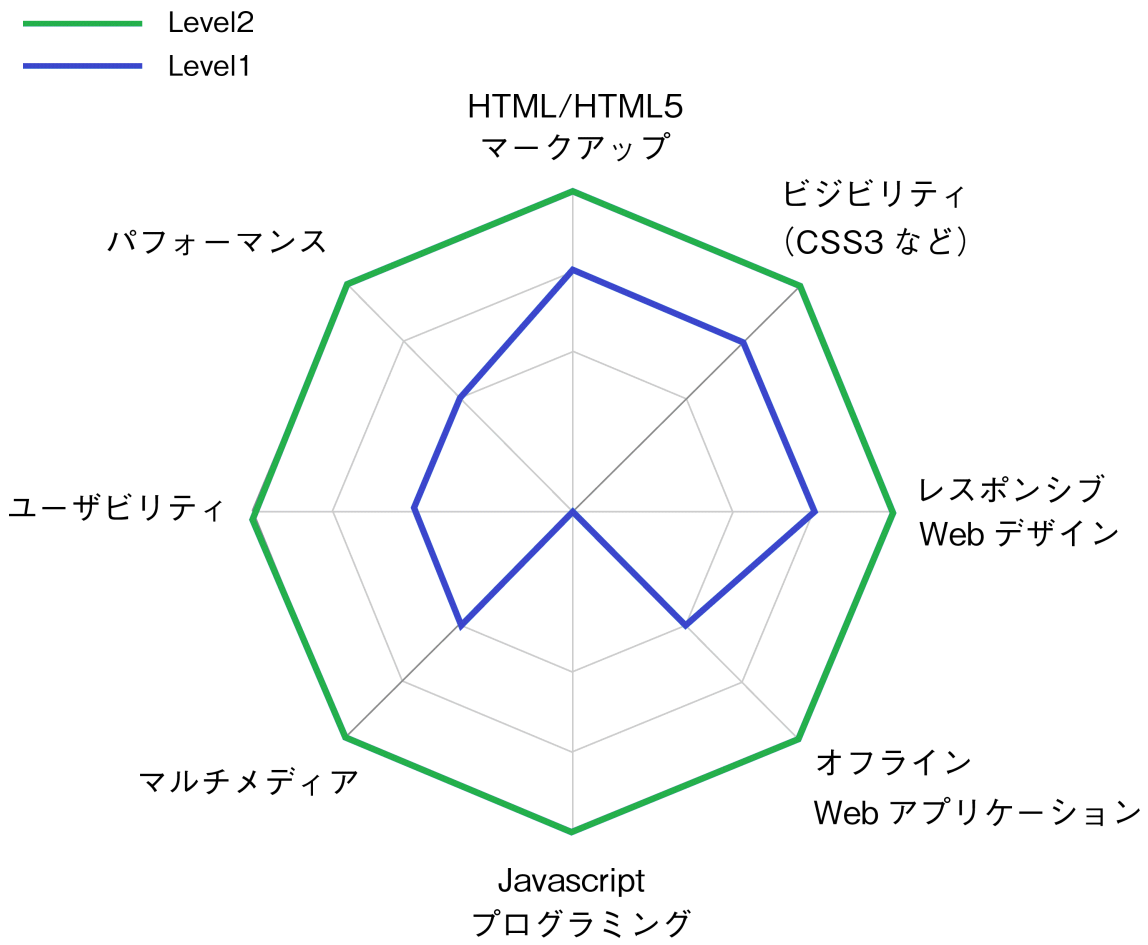
HTML5 Level.2

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：未定
受験料：未定
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。**

レベル2の出題構成(1)

主題	項目	重要度
JavaScript	JavaScript文法 Javascriptの概要 演算子 特殊数値 配列 制御文 関数 型・オブジェクト プロパティ スコープ	**10**
WebブラウザにおけるJavaScript API	イベント	**8**
	ドキュメントオブジェクト/DOM	**6**
	ウィンドウオブジェクト	**8**
	Selectors API	**4**
	テスト・デバッグ	**2**

レベル2の出題構成(2)

主題	項目	重要度
グラフィックス	Canvas(2D)	6
	SVG	1
マルチメディア	video要素,audio要素 (各要素の仕様や、マークアップの記述方法に関してはレベル1に含まれるため範囲外)	2
オフラインアプリケーションAPI	アプリケーションキャッシュの制御	2
Session History and Navigation	History API	3
表示制御	Page Visibility	4
	Timing control for script-based animations	2
ストレージ	**Web Storage**	**4**
	Indexed Database API	**2**
	File API	**2**

レベル2の出題構成(3)

主題	項目	重要度
通信	WebSocket	2
	XMLHttpRequest	4
Geolocation API	Geolocation APIの基本と位置情報の取得	2
Web Workers	並列処理の記述	4
パフォーマンス	Navigation Timing	4
	High Resolution Time	1

学習方法

- **参考書**
- **サンプル問題**
- **出題範囲を確認**
  - 説明できない用語が無いようにする。
- **自分でサンプルを作って確かめる。**
  - 処理の順番などを確認する。
  - Webブラウザ毎に動作が異なることがあるので注意。

2.8 ストレージ

ストレージの用途

範囲外

- セッション情報の保存
- オフライン時の送信データの一時保存
- 画像データ等のキャッシュ
- 作業履歴の保存
- 表示設定の保存
- データ集計のための一時保存

などなど

様々なストレージ

Webブラウザにデータを保存する方法

- Cookie(試験範囲外)
- **Web Strage**
- **Indexed DB**
- Flash(試験範囲外)

データ保存手法としてのCookie

範囲外

- 1データ最大4kb程度。
  (ブラウザによって違いあり)
- ドメイン毎に150〜180データ程度。

テキストデータやアイコン画像ぐらいなら保存できるサイズ。

- Webサーバとの通信時に全てのデータが送受信に含まれるため、トラフィックを無駄に増大させる。
- HTTPヘッダに含まれるため、通信を盗聴されると情報は全て見えてしまう。
- Cookieの最大容量等は
  http://browsercookielimits.squawky.net/
  で調べられる。

2.8.1 Web Strage

知識問題 コードリーティング問題

KVS(Key Value Store)

- 任意の**キー(Key)と値(Value)を関連付け**て管理するデータ保存形式。
- キーを指定すると値を取り出すことができる。
- キーが重複することはない。
- **Cookie**や**Web Storage**がキーによる検索に限定される単純なKVS、**Indexed Database**は複雑な検索が可能な多機能なKVSといえる。
- MySQLやPostgreSQLのようなリレーショナルデータベースに比べると機能は少ないが高速に動作する。

Web Storageの特徴や仕様(1)

- KVS形式で、Webブラウザにデータを保存する。
- **キーによる検索しかできない**ため、一般的なデータベースのような検索や集計の用途には向かない。
- 永続的にデータを保存する**ローカルストレージ**と、ブラウザ(タブ)を閉じるまで一時的にデータを保存する**セッションストレージ**の2種類がある。
- ドメインに紐付いているため、基本的に他のドメインのWeb Storageは操作できない。

Web Storageの特徴や仕様(2)

- 保存できる**データ形式は文字列に限られる**。文字列以外のデータを保存する場合は、Base64などでエンコード(文字列化)する。

- 画像データなどのサイズの大きいバイナリデータの保存には向かない。

- ローカルストレージは2MB〜5MB,セッションストレージは2MB〜無制限(ディスク容量まで)のデータを保存できる。

ローカルストレージとセッションストレージ

- **使用できるメソッドやイベントは同じ。**

- JavaScriptで操作するオブジェクトと、データが自動的に削除されるタイミング、保存可能な容量が異なる。

	オブジェクト	削除されるタイミング	最大容量
ローカルストレージ	localStorage	JavaScriptで明示的に操作	2MB〜 5MB
セッションストレージ	sessionStorage	タブが閉じられるか、 JavaScriptで明示的に操作	2MB〜 無制限

Web Storageの書き込み

Web Storageへデータを保存するには、
localStorageまたはsessionStorageオブジェクトの
setItemメソッドを使用する。

例) localStorage.**setItem('key','value')**;

setItemメソッドの引数には、キーと値を指定する。

文字列以外の値を指定すると...

配列(new Array('a','b','c'))や Dateオブジェクト	"a,b,c"や"Mon Jan 09 2017 00:03:01 GMT+0900 (JST) "という文字列になる
文字列表記のないオブジェクト	"[Object object]"という文字列になる

Web Storageの読み込み

Web Storageからデータを読み込むには、
localStorageまたはsessionStorageオブジェクトの
getItemメソッドを使用する。

例) var value = localStorage.**getItem('key')**;

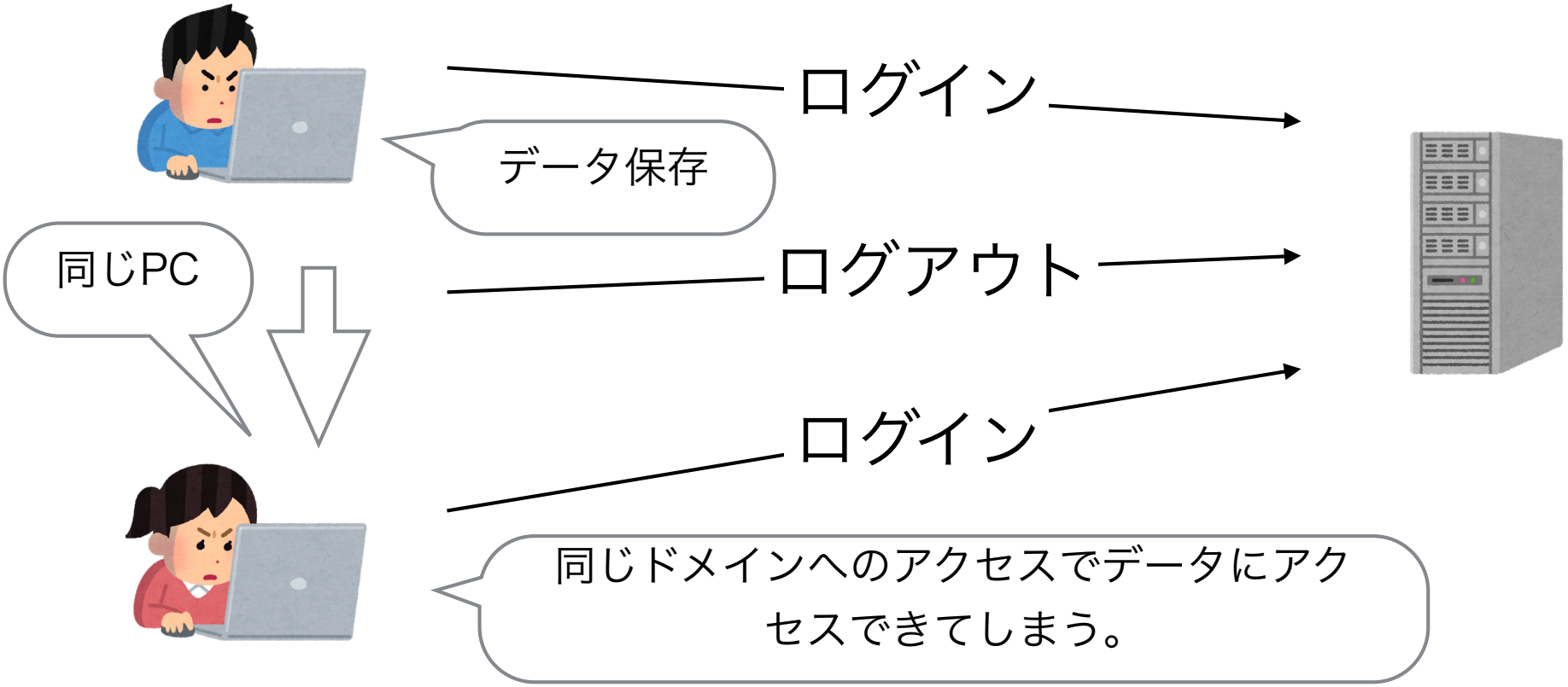
getItemメソッドの引数には、キーを指定する。

その他のメソッド/プロパティ

- **length**プロパティ
  - データ件数を返す。
- **key**メソッド
  - インデックスを指定して、キーを取得する。
  - var key = localStorage.key(0);
- **removeItem**メソッド
  - キーを指定して、エントリを削除する。
  - localStorage.removeItem('key');
- **clear**メソッド
  - 全てのエントリを削除する。

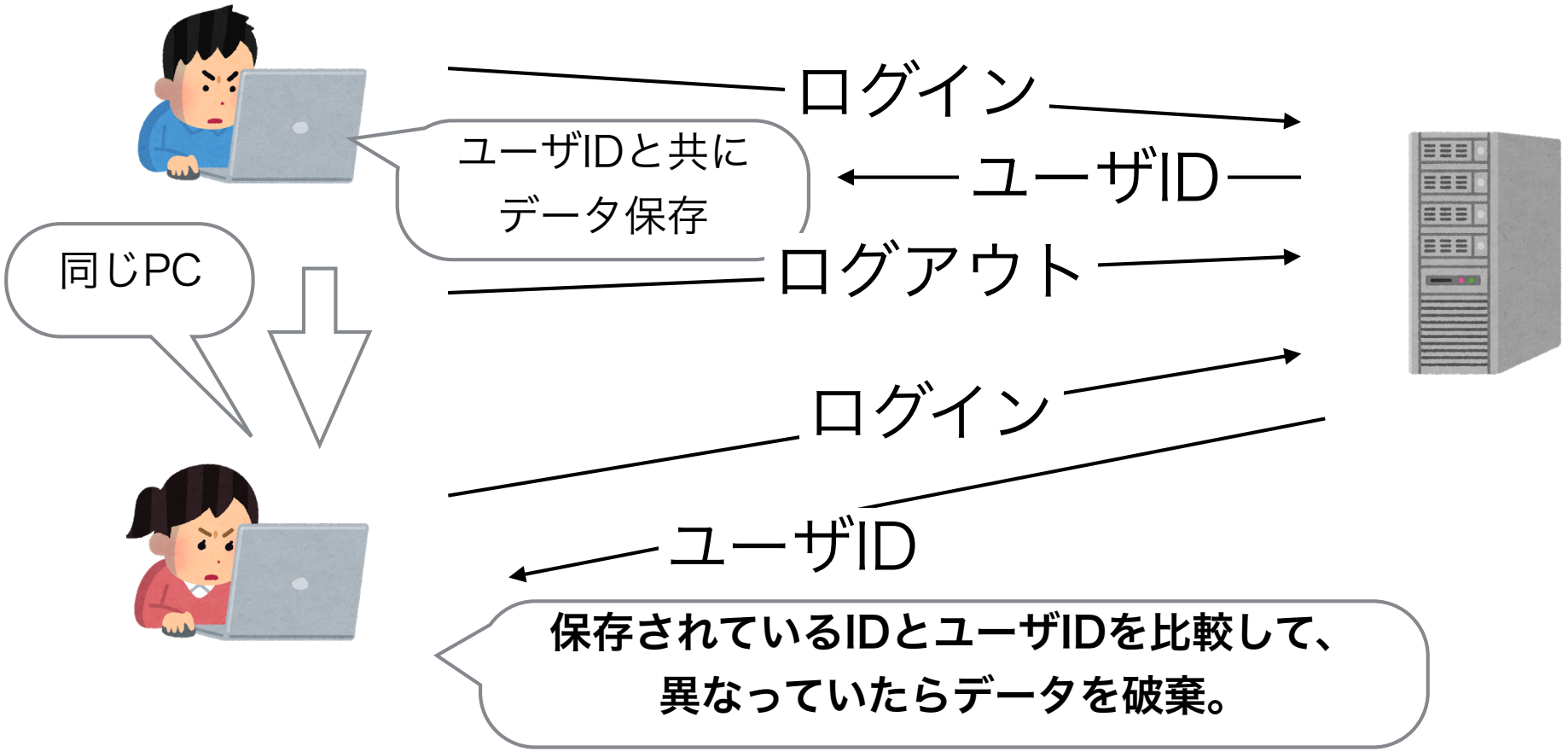
Web Storageのセキュリティ

Web Storageでは、Webサイトにログイン中にのみ参照可能(であるはず)なデータなどを保存すると、別ユーザがデータを参照できてしまうことがある。



セキュリティ対策

ユーザやセッションに紐付いた値をデータと共に保存しておいて、Web Storageを参照する際に異なっていたらデータを破棄するなどの工夫が必要。



Web Storageのイベント

- ローカルストレージまたはセッションストレージが変更されると、**storage**イベントが発生し、**window.onstorage**イベントハンドラで捕捉できる。

- Storage Eventオブジェクトのプロパティ
  - **key** … 変更されたエントリのキー
  - **oldValue** … 変更前の値
  - **newValue** … 変更後の値
  - **url** … 変更されたストレージの紐付けられたURL
  - **storageArea** … 変更されたストレージ (localStorage または sessionStorageオブジェクト)

問題1

以下のスクリプトのうち、Webブラウザを閉じても値がそのまま正しく保存されるものを選択しなさい。

A. localStorage.setItem('key' , 'value');

B. localStorage.setItem('key' , new Date( ));

C. sessionStorage.setItem('key' , new Date( ));

D. sessionStorage.setItem('key' , 'value');

2.8.2 Indexed Database API

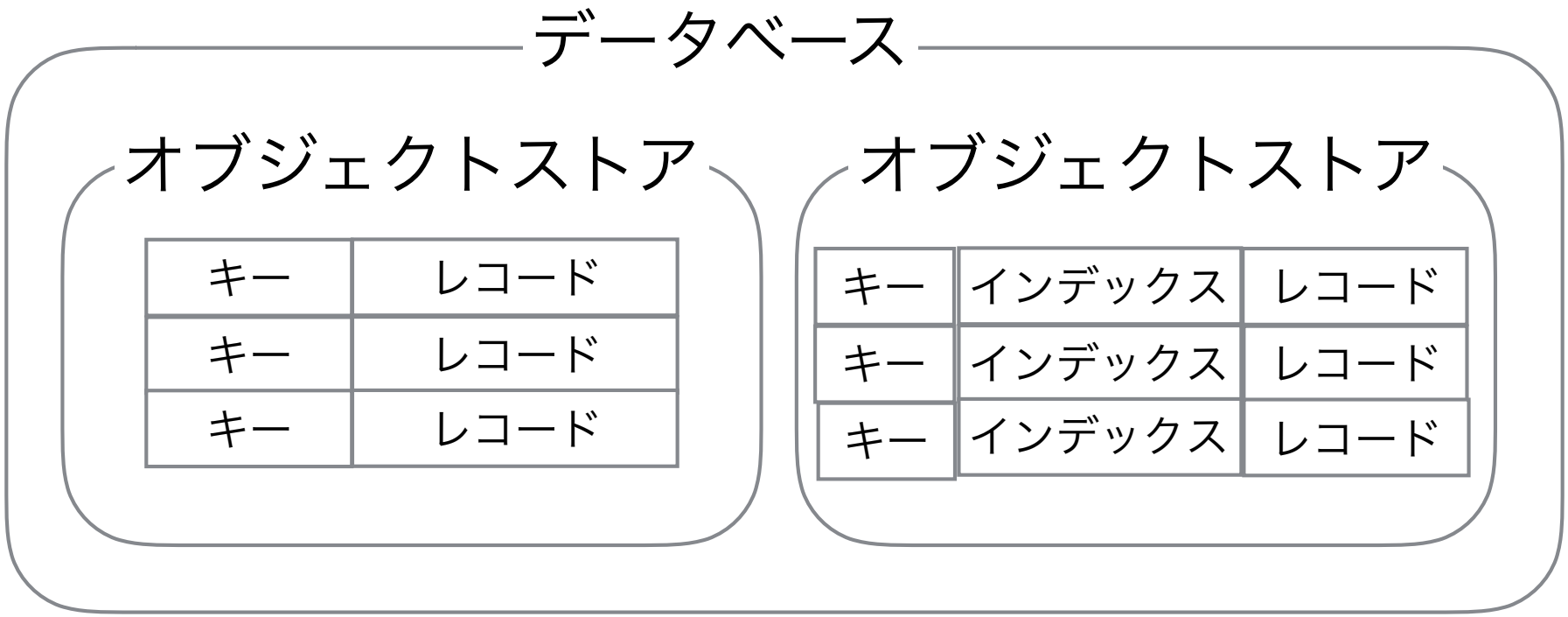
知識問題 コードリーティング問題

Indexed Database APIの特徴

- KVSである。
- JavaScriptの**オブジェクトをそのまま保存**できる。
- **インデックス**を作成すると**キー以外の項目でデータを検索**できる。
- **トランザクション**機能を持つ。
- **非同期処理**を行なう。

Indexed Databaseの構造

- ドメイン毎に複数のデータベースを持つことができる。
- オブジェクトストアは、MySQLなどのRDBMSでいうテーブルにあたる。
- オブジェクトストア内に、レコードとしてJavaScriptのオブジェクトを保存できる。



データベースへのアクセス

ブラウザ毎に異なるIDBFactoryオブジェクトを選択する。
Firefox 38(2015年5月12日リリース)でmozIndexedDBは削除

```
var indexedDB = window.indexedDB ||
window.mozIndexedDB || window.msIndexedDB;

var request = indexedDB.open('sample',1);
```

データベースを開く。データベースが無ければ作成する。

データベース名とバージョン

オブジェクトストアの作成

範囲外

```
var request = indexedDB.open('sample',1);

request.onupgradeneeded = function(e){

  db = request.result;

  db.createObjectStore("store1",{autoIncrement:true});

}
```

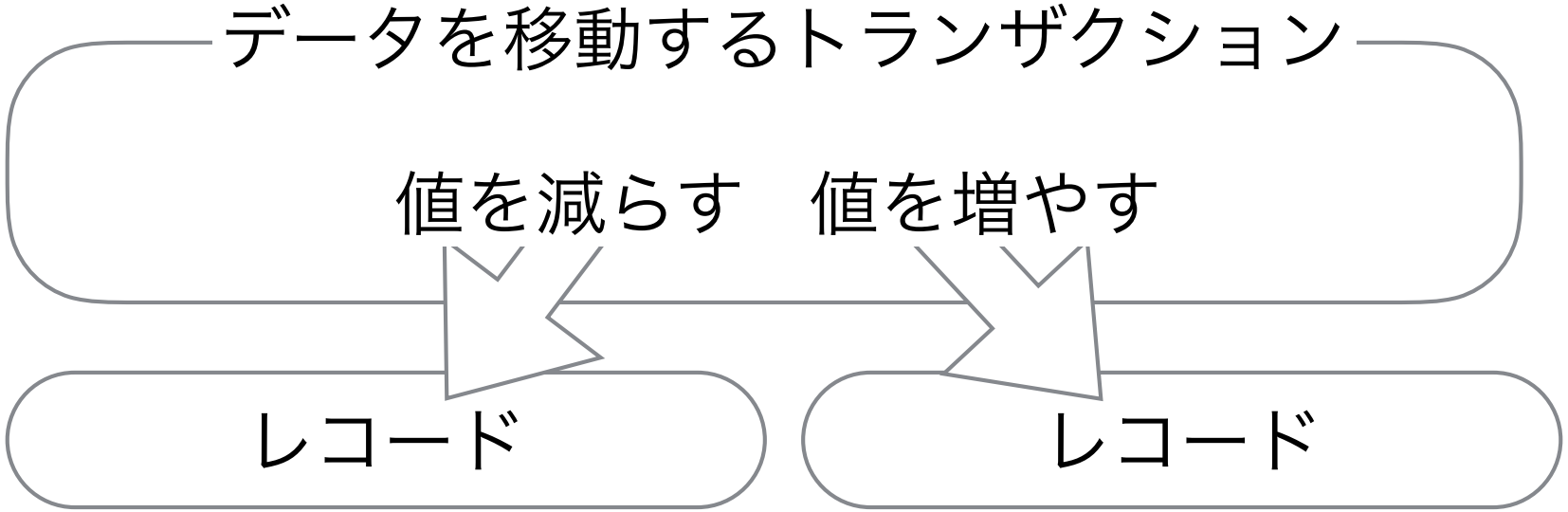
新規作成されたか、バージョンが上がったときだけupgradeneededイベントが発生する。

キーの自動採番を有効化

ObjectStore名を指定して作成する。

トランザクション

- データを操作する、分割できない一連の処理を**トランザクション**と呼ぶ。
- トランザクションに含まれる処理は、**全て完了するか、全て失敗するか**のどちらかである。
- Indexed Databaseでは、レコード操作は全てトランザクションを通じて行なう。



データの保存

```
/* トランザクションを作成する */

var tr = db.transaction(['store1'], 'readwrite');

/* トランザクションからオブジェクトストアを選択する */

var store1 = tr.objectStore('store1');

/* オブジェクトストアにレコードを追加する */

var req = store1.add( {id:1,text:'HTML5'} );
```

対象のオブジェクトストアを複数選択することもできる

オブジェクトストアがautoIncrementならキーは自動採番される。
第2引数にキーを指定することもできる。

データの読み込み

```
/* トランザクションを作成する */
var tr = db.transaction(['store1'], 'readonly');

/* トランザクションからオブジェクトストアを選択する */
var store1 = tr.objectStore('store1');

/* オブジェクトストアからキーを指定しレコードを取得する */
var req = store1.get(1);

/* データの取り出し */
var obj = req.result;
```

読み込み専用にもできる

キーを指定して取得

その他の機能

- データベースの削除

  indexedDB.deleteDatabase('データベース名');

- データベースを閉じる

  db.close();

- オブジェクトストアの削除

  db.deleteObjectStore('オブジェクトストア名');

問題2

Indexed Databaseの特徴として誤っているものを選択しなさい。

A. KVSである。

B. インデックスによる検索に対応している。

C. トランザクションには対応していない。

D. オブジェクトを保存できる。

ストレージ機能の違い

Web StorageとIndexed Databaseの違いは、データの保存形式と検索方法,最大容量,トランザクションといえる。

	保存形式	検索方法	最大容量	トランザクション
Web Storage	文字列のみ	キーのみ	2MB～5MB	なし
Indexed Database	JavaScriptのオブジェクト	キー 項目値	無制限	あり

最大容量は、"デバイスの空き容量の10%まで"や、"ユーザの許可が得られれば無制限"など、Webブラウザによって大きく異なる。

問題3

様々な画像データを、検索可能なタグを付けて、Webブラウザに永続的に保存しておきたい。適切な選択肢を選びなさい。

A. Cookie

B. Local Storage

C. Session Storage

D. Indexed Database

2.8.3 File API

知識問題 コードリーティング問題

ローカルファイルアクセス機能の概要

- **<input type="file">で選択された、またはWebブラウザにドラッグ&ドロップされたローカルファイルを、JavaScriptで読み込むことができる。**

- サーバサイドでしかできなかったことが、クライアントサイドでできるようになった。

- クライアントサイドで画像ファイルを読み込んで加工、その後サーバにアップロードなども可能に。

セキュリティ観点での制限事項

- セキュリティ的な観点から、アクセスできるファイルはユーザが意識的に選択またはドラッグ&ドロップしたものに限られる。
- プログラム側から任意のファイルを選択することはできない。
- 任意のファイルへの新規保存や上書き保存はできないが、File APIで作成、編集したファイルをユーザにダウンロードさせることはできる。

ファイルオブジェクトリストの取得

```
<input id="fileItem" type="file" multiple>
<script>
/* Input要素を取得 */
var fileItem = document.getElementById('fileItem');
/* changeイベントが発生したら */
fileItem.addEventListener('change',function(){
    /* Input要素からFileListオブジェクトを取得  */
    var fileList = fileItem.files;
}
</script>
```

複数選択できるようにするなら、
multiple属性を付ける

FileListオブジェクト

FileAPIで、input要素または、ドラッグ&ドロップで選択されたファイルのリストを表わすオブジェクト。

プロパティ	概要
length	**リストに含まれるFileオブジェクトの数。**

メソッド	概要
item	**引数にインデックスを指定してFileオブジェクトを取得する。**

Fileオブジェクト

File APIで**ひとつのファイルを管理**するオブジェクト

プロパティ	概要
name	**ファイル名**
lastModifiedDate	**最後に変更された日時のDateオブジェクト**

Blobオブジェクトのプロパティやメソッドを継承

プロパティ	概要
size	データのバイト数
type	データの種類を表すMIMEタイプ

メソッド	概要
slice	指定した範囲の部分的なデータを返す

ローカルファイルの読み込み

```
var fileItem = document.getElementById('fileItem');
var reader = new FileReader();
reader.onload = function(){
  alert(reader.result);
}
fileItem.addEventListener('change',function(){
  var item = fileItem.files.item(0);
  reader.readAsText(item);
});
```

FileReaderオブジェクトの作成

ファイル読み込み完了時の処理を登録
resultに結果が格納されている

ファイルリストから先頭のFileオブジェクトを取得

ファイルを文字列として取得するよう指定

FileReaderオブジェクト

Fileオブジェクトから、内容を読み出すためのオブジェクト

プロパティ	概要
readyState	状態を表わす数値 0:EMPTY 1:LOADING 2:DONE
result	読み込んだファイルの内容
error	ファイルの読み込み中に生じたエラー

メソッド	概要
readAsArrayBuffer	ArrayBufferオブジェクトとして読み込む
readAsText	データを文字列として読み込む
readAsDataURL	data:URLとして読み込む
abort	読み込み処理を中断する

問題4

ローカルファイルの内容をバイナリデータとして
取得するメソッドを選択しなさい。

A. FileReader.readAsArrayBuffer

B. File.readAsBinary

C. FireReader.readAsDataURI

D. File.readAsArrayBuffer

2.2.5 テスト・デバッグ

JavaScriptのテスト・デバッグ手法

- 各Webブラウザの開発者ツールを使う。
- **JavaScript内で変数の内容などを出力する。**
  - HTML5ではdocument.writeによる出力は非推奨になっている。

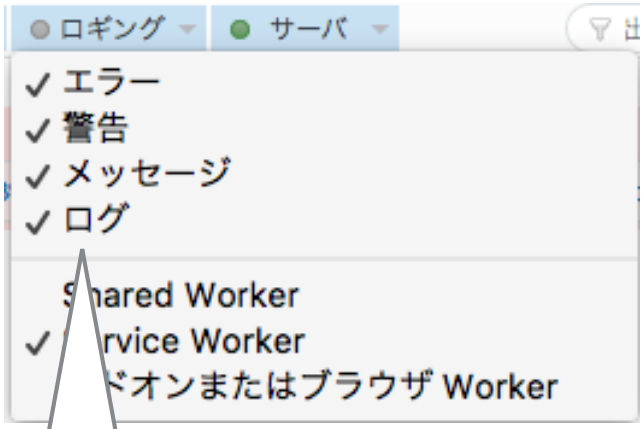
    https://www.w3.org/TR/2011/WD-html5-20110525/apis-in-html-documents.html#document.write
  - 一般的な表示：innerHTMLプロパティの使用。
  - **テスト・デバッグ表示：Consoleオブジェクトを使用。**

Consoleオブジェクト

- Webブラウザの**開発者ツール(コンソール)への情報出力**や、**処理の実行時間の計測**など、開発に便利なメソッドが用意されている。
- Webブラウザによって、実装状況や表示結果が異なるので要注意。

consoleへの出力

メソッド名	概要、記述例
debug	console.logと同等。Firefoxでは非推奨。
error	エラーメッセージを表示。
info	メッセージタイプのログ情報を出力。
log	一般タイプのログ情報を出力。
warn	警告メッセージを出力



コンソールの表示設定で種類別に表示非表示を切り替えて表示を整理できる。

基本的に、
アイコンの表示などが違うだけ。
Webブラウザによっては、
error,warnメソッドの結果に、
呼び出し箇所が表示される。

特殊な出力

- **assert**
  console.assert(条件,'msg');
  条件がfalseのとき、msgを出力する。
  関数の返り値をチェックして異状を検知する場合など。

- **dirxml**
  console.dirxml(HTML要素);
  HTML要素を表示。
  console.logでは構造がわかりづらい場合など。

- **trace**
  console.trace();
  スタックトレース(関数の呼び出し情報)を表示。
  目的の関数が、どこから呼び出されたかを把握する。

プロファイリング

- JavaScriptの実行でどの処理に時間が掛っているかを探す。
- **console.profile**メソッドの呼び出しから、**console.profileEnd**メソッドの呼び出しまでの間に実行された関数の実行時間(プロファイル)を計測する。
- 実行時間は開発者ツールのパフォーマンス(Firefox)またはProfiles(Chrome)に表示される。
- 各プロファイルに名前を付けることで複数のブロックにわけてプロファイルを取得できる。
- 開発者ツールの機能だけでも計測できるが、全体か関数単位での計測結果になる。

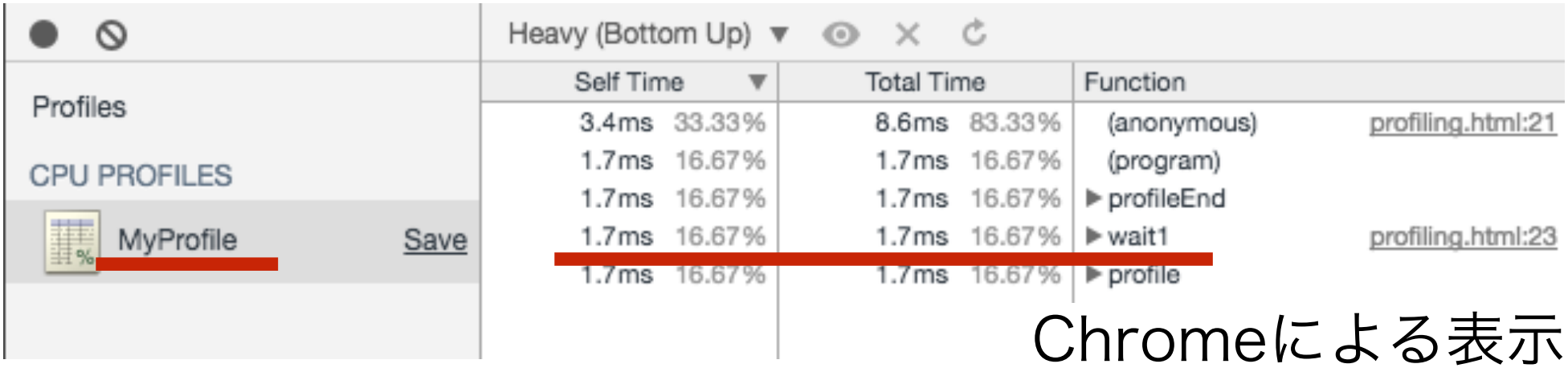
プロファイリングの実例

```
/* プロファイリング開始 */

console.profile('MyProfile');

wait1(1000);

/* プロファイリング終了 */

console.profileEnd('MyProfile');

wait2(1000);
```

wait1とwait2は、forループを使って時間を浪費する関数

profileとprofileEndの間のwait1は計測に含まれる

profileEndの後のwait2は計測に含まれていない



Chromeによる表示

問題5

HTML要素について構造を考慮した出力を行なうメソッドを選択しなさい。

A. console.log( )

B. console.dirhtml( )

C. console.dirxml( )

D. console.info( )

本日の内容

- 試験概要
- ストレージ
  - Web Strage
  - Indexed Database API
  - File API
- テスト・デバッグ

問題の解答

- 問題1: A　/ Webブラウザを閉じてもデータが保持されるのは、localStorageであり、正しく保存できるのは文字列だけである。

- 問題2: C / Indexed Databaseはトランザクション機能を持つ。

- 問題3: D / 複数の画像データを保存する場合のデータ量と、検索可能なタグをデータとして保存するには、Indexed Databaseが適切。

- 問題4: A / ファイルからのバイナリデータ読み込みはFileReader.readAsArrayBufferメソッドを使用。

- 問題5: B / dirhtmlではなく、dirxmlなので注意。