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
- history API
- ストレージ
- Web Strage
- Indexed Database API
- File API
- バイナリーデータ

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
- グラフィックデザイナー
- フロントエンドプログラマー
- HTMLコーダー
- Webディレクター
- 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

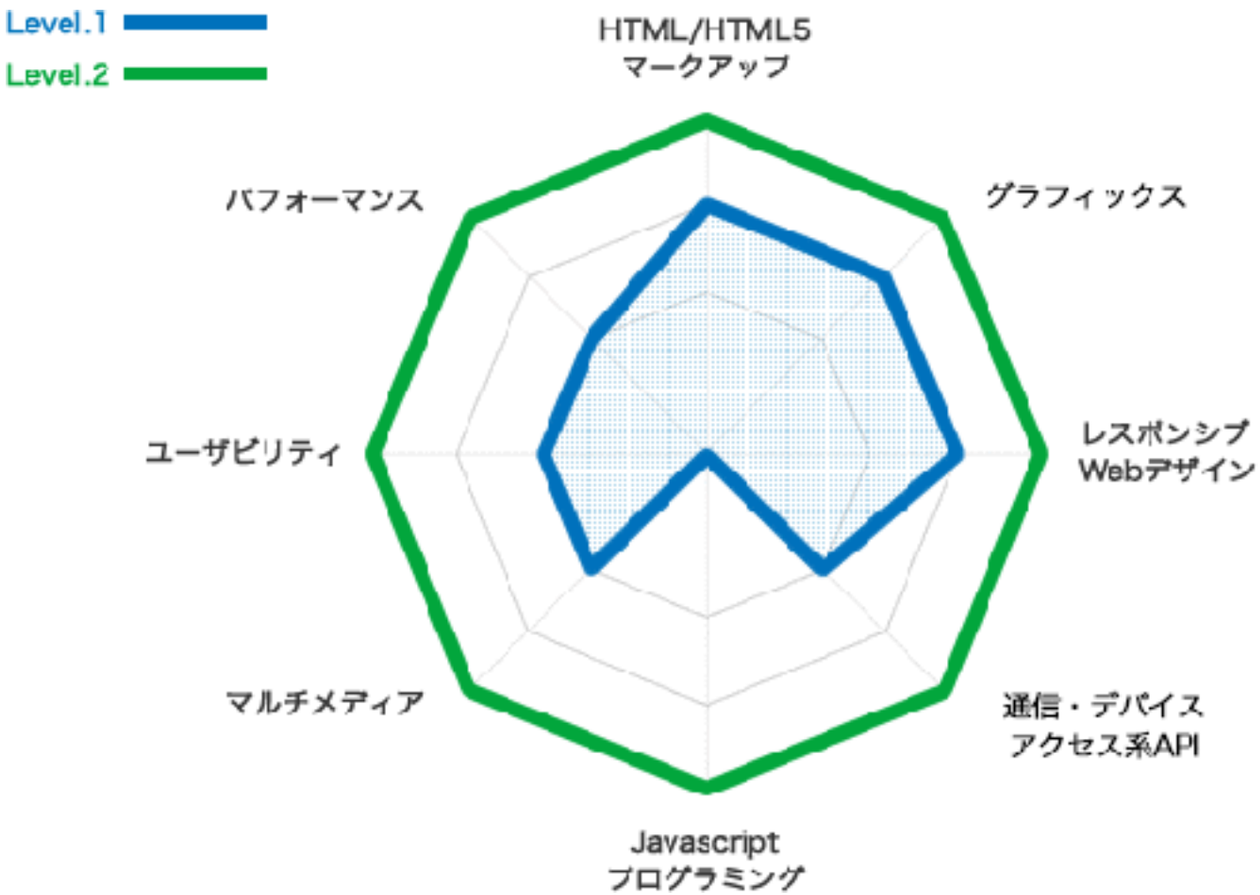
HTML5 Level.2

最新のAPIを駆使したWebアプリケーションを設計・開発できる。

対象

- Webデザイナー
- フロントエンドプログラマー
- HTMLコーダー
- Webディレクター
- 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

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

グラフィックス

JavascriptやCSSなどを用いて、動的にグラフィックスを生成したりアニメーションを実現したりする技術

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

通信・デバイスアクセス系API

JavaScriptからクラウドと通信をして情報の送受信を行ったり、センサーなどのデバイスにアクセスしたりする技術

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

JavaScriptやCSSなどを用いて、デザイン仕様に沿った見やすい表示や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

ストレージ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たり、オフラインでも動作する仕組みを作るための技術

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 （税別）
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Webコンテンツを作成することができる。**
- **ユーザー体験を考慮したWEBコンテンツを設計・作成することができる。**
- **スマートフォンや組み込み機器など、ブラウザが利用可能な様々なデバイスに対応したコンテンツを制作できる。**
- **HTML5で何ができるか、どういった技術を使うべきかの広範囲の基礎知識を有する。**

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(税別)
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る。**
- **スマートフォンなどでネイティブアプリに近い機能を組み込んだ先端のWebアプリケーションに近い機能を組み込んだ先端のWebアプリケーションを開発することができる。**
- **APIのセキュリティモデルを理解したうえで開発することができる。**

レベル2の出題構成(1)

主題	項目	重要度
JavaScript	JavaScript文法	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	7
	History API	**7**
	テスト・デバッグ	6
グラフィックス・アニメーション	Canvas(2D)	8
	SVG	2
	Timing control for script-based animation	2
マルチメディア	メディア要素のAPI	5
ストレージ	**Web Storage**	**7**
	Indexed Database API	**5**
	File API	**5**
	バイナリーデータ	**4**

レベル2の出題構成(2)

主題	項目	重要度
通信	Web Socket	5
	XMLHttpRequest	5
	Server-Sent Event	1
デバイスアクセス	Geolocation API	5
	Device Orientation	1
パフォーマンスとオフライン	Web Workers	5
	High Resolution Time	2
	オフラインアプリケーションAPI	3
	Page Visibility	2
	Navigation Timing	1
セキュリティモデル	クロスオリジン制約とCORS	4
	セキュリティモデルとSSLの関係	4

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。(初期化->処理->後片付け)
  - Webブラウザ毎に動作が異なることがあるので注意。
  - Webサーバの有無で動作が異なることがあるので注意。

2.2.5 history API

知識問題 コードリーティング問題

History APIの概要

- **window.history**オブジェクトを使うと、Webブラウザの履歴に任意のURLを追加したり、履歴を基にページを戻ったり進めたりできる。

- ただし、履歴内の現在位置や、履歴内の任意のURLを知る方法はない。

- ajaxなどでコンテンツを書き変えると、URLとコンテンツの関連性が崩れてしまうがHistory APIでは現在のURLを任意に変更できるため、URLとコンテンツを適切に連携できる。

ページ履歴に関する情報取得

- 現在の履歴の個数を知るには、window.history.lengthプロパティを使用する。
- JavaScriptの通常の配列(Array)と異なり、lengthプロパティに値を代入することはできない(read only)。

履歴の操作とページ移動

- 進む

  例) window.**history.forward**();

- 戻る

  例) window.**history.back**();

- 履歴内の任意のページに移動する

  例) window.**history.go**(1); // ひとつ進む

  window.**history.go**(-1); // ひとつ戻る

URL操作

- HTML5から追加された**pushState()**,**replaceState()**メソッドを使うと、ロケーションバー(アドレスバー)上のURLを書き換えることができる。

- 書き換えてもリロードはされない。

- pushState()とreplaceState()の違いは、履歴にあたらしいURLを追加するか、現地点の履歴を置き換えるか。

  例)
  window.**history.pushState**(null , null , url);
  window.**history.replaceState**(null , null , url);

現在のページの情報取得

Locationオブジェクトのプロパティから現在のページの情報を取得できる。

プロパティ	概要	例
href	URL全体	http://html5exam.jp:80/index.html?q=abc#def
origin	スキーマ:ホスト名:ポート	http://html5exam.jp:80
protocol	プロトコル	http:
host	ホスト名:ポート	html5exam.jp:80
hostname	ホスト名	html5exam.jp
port	ポート番号(明示したときのみ)	80
pathname	パス	/index.html
search	URLの?以降(クエリ文字列)	?q=abc
hash	URLの#以降(ハッシュ文字列)	#def

ページのロードとリロード

ページのロード(遷移)には、以下の3通りの方法が使用できる。

- window.locationオブジェクトへURLを代入
  例) window.**location = 'http://html5exam.jp';**

- window.location.hrefプロパティへURLを代入
  例) window.**location.href = 'http://html5exam.jp';**

- window.location.assign(url)メソッドを実行
  例) window.**location.assign(**'http://html5exam.jp'**);**

ページのリロードは、window.location.reload()メソッドを実行する。
例) window.**location.reload();**

ページ履歴の置換

window.location.assign()メソッドなどの替わりに、window.location.replace()メソッドを使用してページを移動すると、現在のページがHistoryに保存されない(History上の現在のページのURLがReplaceされる)。

例)

```
// "戻る"で現在のページに戻れる
window.location.assign('http://html5exam.jp');

// "戻る"で現在のページに戻れない
window.location.replace('http://html5exam.jp');
```

問題1

以下のスクリプトを実行した結果、ロケーションバーに表示されるURLはどれか。

```
window.history.pushState(null,null,'/a.html');
window.history.pushState(null,null,'/b.html');
window.history.pushState(null,null,'/c.html');
window.history.replaceState(null,null,'/d.html');
window.history.back();
window.history.go(-1);
```

A. a.html

B. b.html

C. c.html

D. d.html

2.5 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)

2.5.1 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。

ログイン

データ保存

同じPC

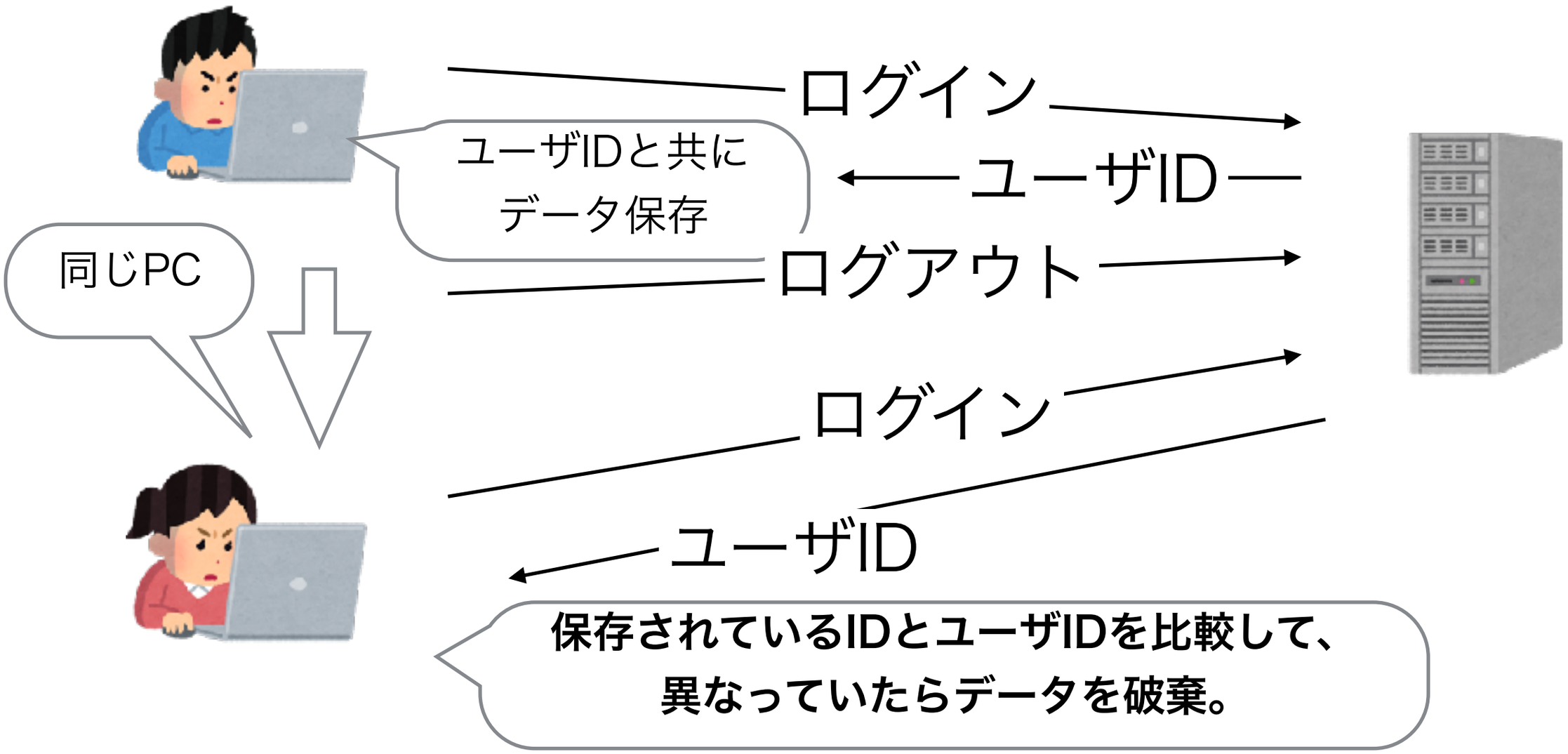
ログアウト

ログイン

同じドメインへのアクセスでデータにアクセスできてしまう。

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2

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');

2.5.2 Indexed Database API

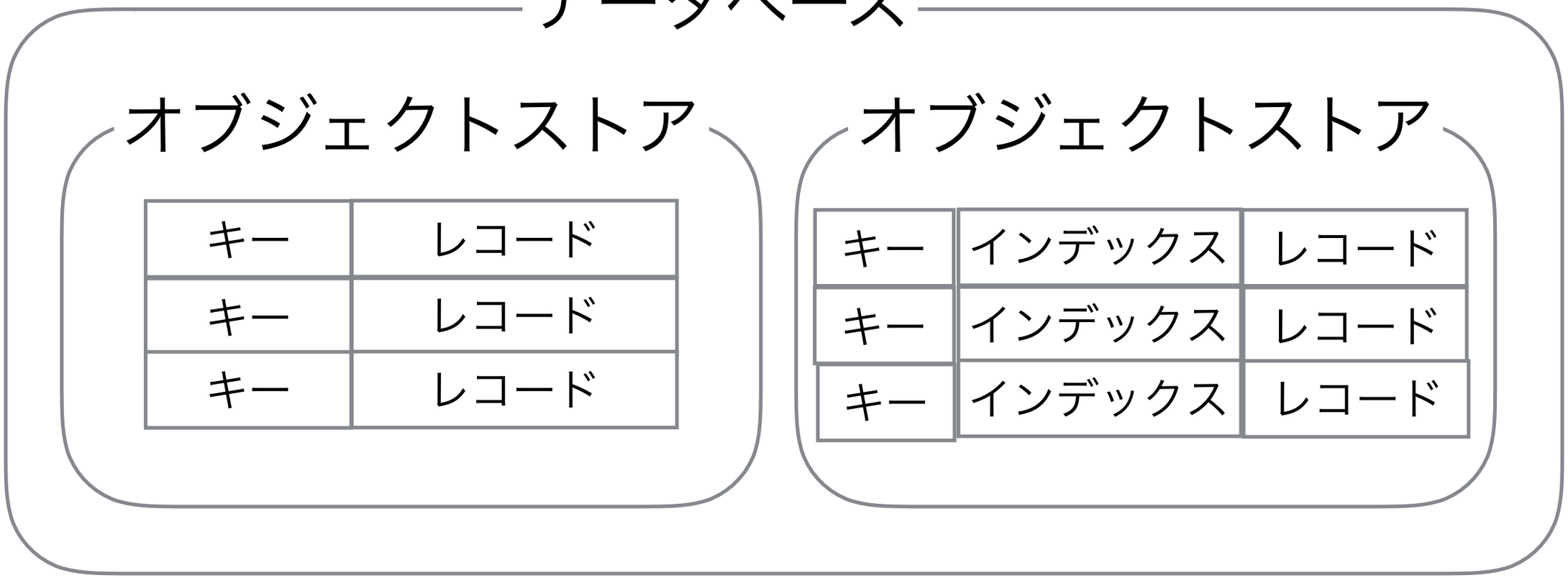
知識問題 コードリーティング問題

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

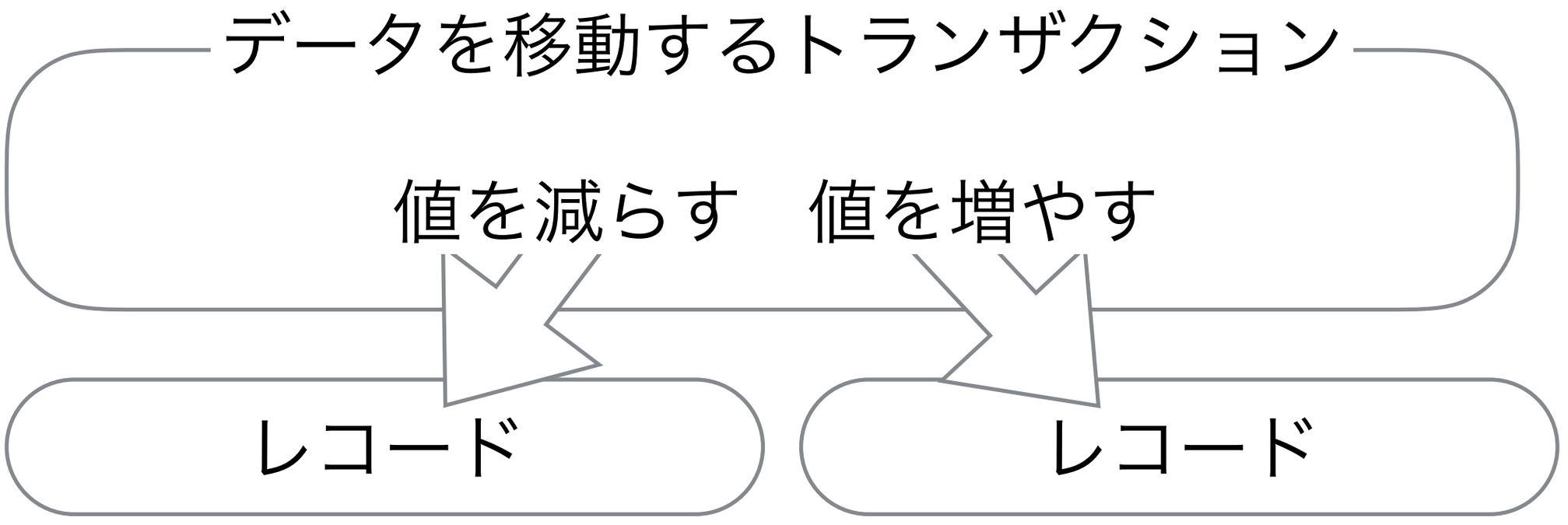
ObjectStore名を指定して作成する。

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3

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4

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

2.5.3 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.getElementByld('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.getElementByld('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5

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

2.5.4 バイナリーデータ

知識問題 コードリーティング問題

バイナリーデータの概要

- Webアプリケーションの利用範囲が広がるにつれ、Webブラウザが対応していない独自形式のファイルを操作する必要性がでてきた。(動画、画像、VR用データなど)

- ローカルファイルや、サーバから取得したデータを直接操作するためにArrayBuffer、型付き配列などの仕組みが用意されている。

- 任意の形式のバイナリーデータを、FileAPIなどを使ってWebブラウザからダウンロードすることもできる。

Blob,Blob URL

- **Blob**は変更不可能(immutable)な生データ(バイナリデータ)を扱うためのオブジェクト。File APIで使用するFileオブジェクトの基になっている。

- Blobからデータを取り出すには、FileReaderを使用する。(File APIを参照)

- 作成、編集したBlob/FileをWebブラウザからダウンロードするには、**createObjectURL**メソッドでBlob URLを作成する。

Blob URLの例

```
var objectURL = window.URL.createObjectURL(blob);
console.log(objectURL);
// => blob:null/e749dd31-24d1-974e-84f3-a369581d361c
```

createObjectURLを呼ぶごとに異なるURLが生成される

ArrayBufferとビュー

ArrayBufferはバイナリデータを表すためのデータ型

ArrayBufferには直接アクセスできないため、ビュー(型付き配列、またはDataView)を通じて操作する

メモリ上の ArrayBuffer

...
0A
1B
2C
3D
4E
5F
...

Byte単位で値を保存

型付き配列ビュー (Int16Array)

...	
0A1B	Int16
2C3D	Int16
4E5F	Int16
...	

指定した型の配列としてアクセスできる

DataView

...	
0A	Int8
1B2C3D4E	Int32
5F	Int8
...	

任意の構造のデータとしてアクセスできる

型付き配列(Typed Array)

- ArrayBufferを操作するためのビューの一種。
- Byte単位ではなく、指定した型の値の配列としてアクセスできる。

型	ビット数	オブジェクト
浮動小数点型(**Float**)	64	Float64Array
	32	Float32Array
符号付き整数(**Int**)	32	Int32Array
	16	Int16Array
	8	Int8Array
符号なし整数(**Uint**)	32	Uint32Array
	16	Uint16Array
	8	Uint8Array
	8	Uint8ClampedArray

※Uint8ClampedArrayは0以下の数値は0に、
255以上の数値は255に制限されるため画像処理に向いている。

使用例

```
var buf = new ArrayBuffer(4);
var i8 = new Int8Array(buf);
i8[0] = 0x01;
i8[1] = 0x02;
i8[2] = 0x03;
i8[3] = 0x04;
var i16 = new Int16Array(buf);
console.log(i16[0].toString(16));
// => 0x0201
console.log(i16[1].toString(16));
// => 0x0403
```

バイトオーダー

- 複数バイトの値が、メモリに保存される順番のことを**バイトオーダー**という。

- 最上位バイトが先頭のものを**ビッグエンディアン**、最下位バイトが先頭になるものを**リトルエンディアン**という。

- **型付き配列ビュー**は**ネイティブのバイトオーダー**になる。
(Intel x86系CPUはリトルエンディアン、Motorola 68000系CPUやJava VMはビッグエンディアン)

- エンディアンを指定してアクセスする場合は、**DataView**を使用する

数値(16進数)	ビッグエンディアン	リトルエンディアン
ABCDEF12	AB CD EF 12	12 EF CD AB

※リトルエンディアンにも繰り上りの
計算がしやすいなどのメリットがある。

ファイルからバイナリデータを読み込む

```
var fileItem = document.getElementById('fileItem');
var reader = new FileReader();
fileItem.addEventListener('change',function(){
  var item = fileItem.files.item(0);
  var buf = reader.readAsArrayBuffer(item);
  var uint8 = Uint8Array(buf);
  console.log(uint8[0]);
});
```

問題6

リトルエンディアンの環境で、変数bufに[0x01,0x02,0x03,0x04]というArrayBufferオブジェクトが入っているとき以下のプログラムを実行したi16[1]の値はどれか。

```
var i16 = new Int16Array(buf);
```

A. 0x43

B. 0x0403

C. 0x0304

D. 0x4321

本日の内容

- 試験概要
- history API
- ストレージ
- Web Strage
- Indexed Database API
- File API
- バイナリーデータ

問題の解答

- 問題1: A　/ 履歴にはa,b,cと追加され、cがdにreplaceされる。back()でb,go(-1)でaと戻ることになる。

- 問題2: 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3: C / Indexed Databaseはトランザクション機能を持つ。

- 問題4: 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5: 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6: B / リトルエンディアンでは、前後が逆になる。