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

2019年8月17日

株式会社富士通ラーニングメディア

結城陽平 高橋映美

富士通ラーニングメディアのご紹介

1. 会社概要

設立	1977年6月30日
資本金	3億円（全額 富士通株式会社）
売上高	103億円（2017年度連結）
従業員	432名（2018年3月末現在）
事業内容	人材育成・研修サービス(公開コース***1,450***コース、年間***92,000***名受講） 個人のお客様向けパソコン教室（富士通オープンカレッジ）
関係会社	株式会社富士通ラーニングメディア沖縄（研修サービス・研修サービスサポート） 株式会社富士通ラーニングメディア・スタッフ（人材派遣）
出資会社	株式会社アクト・ブレーン・ベトナム（ソフトウェア開発など）
事業所 関連施設	東京・名古屋・大阪・沖縄に5拠点（約***40***教室、***900***名以上の定員数）


品川本社


品川LC


名古屋事業所/LC


関西事業所/LC

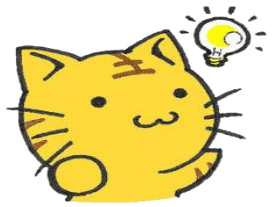

沖縄事業所/LC

2. 富士通ラーニングメディアが目指すこと

ICT人材を支える人材育成のパートナーでありつづける

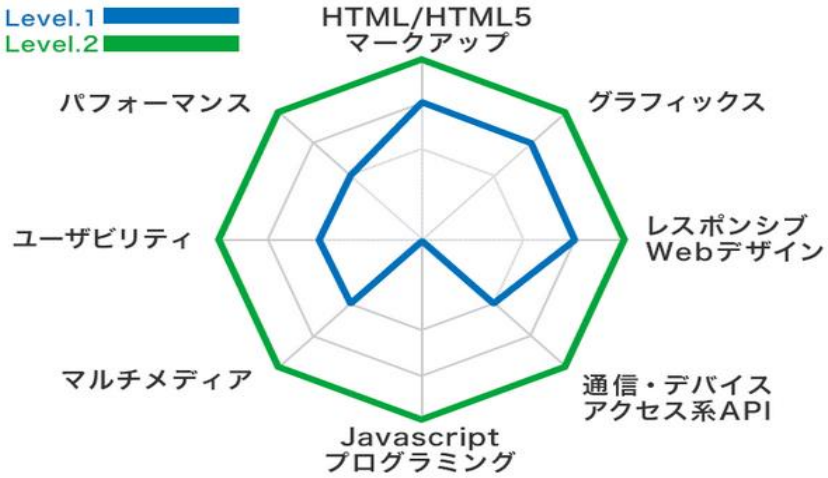
- 最高水準の『知』のサービスを提供することにより、お客様の真のパートナーを目指します。
- お客様の「成長のスパイラル」をサポートします。

HTML教科書 HTML5プロフェッショナル認定試験 スピードマスター問題集 Ver2.0対応



絶賛販売中です！！

HTML5プロフェッショナル認定資格 レベル2



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よ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

目次

- 2.1.1 JavaScript文法
- 2.5.2 Indexed Database API
- 2.9.1 クロスオリジン制約とCORS
- 2.9.2 セキュリティモデルとSSLの関係

サンプルコード

サンプルコードは以下のリポジトリに格納しています。

クロスブラウザ対策などを行っていないので、環境によっては動作しないので予めご了承ください。

https://github.com/FujitsuLearningMedia/HTML5-Lv2-Seminar-20190817

JavaScript文法

型

代入した値によって、動的に型が決まる

分類	型	値例
プリミティブ	string	'LPI', "FLM"
	number	10, 20
	boolean	true, false
オブジェクト	object	new String("")

```
1.  var str1 = "FLM";                  // プリミティブ型
2.  var str2 = new String("LPI");      // オブジェクト型
```

プリミティブとオブジェクト

- プリミティブ型からオブジェクト型に暗黙的に変換される。
  →**プリミティブ型の変数でもメソッドを呼び出せる**

```
1.  var str1 = "FLM";                  // プリミティブ型
2.  var str2 = new String("LPI");      // オブジェクト型
3.
4.  console.log(str1);
5.  console.log(str2);
6.  console.log(str1.substring(0, 1));
7.  console.log(str2.substring(0, 1));
```

演算子

関係演算子

演算子	説明
<	左辺は右辺より小さい
<=	左辺は右辺以下である
>	左辺は右辺より大きい
>=	左辺は右辺以上である
==	左辺と右辺の値は等しい
!=	左辺と右辺の値は等しくない
===	左辺と右辺はデータ型と値が等しい
!==	左辺と右辺はデータ型または値が等しくない

```
1.  var num1 = 10;
2.  var num2 = "10";
3.
4.  console.log(num1 == num2);      // true
5.  console.log(num1 === num2);     // false
```

サンプル問題 1

■ 以下の変数、str1とstr2が宣言されています。選択肢のソースコードの実行結果として、trueと表示されるものを**すべて**選びなさい。

```
1.  var str1 = "FLM";
2.  var str2 = new String("FLM");
```

A) console.log(str1 == str2);

B) console.log(str1 === str2);

C) console.log(typeof str1 == typeof str2);

D) console.log(str1 instanceof String);

E) console.log(str2 instanceof String);

解答 1

■ 以下の変数、str1とstr2が宣言されています。選択肢のソースコードの実行結果として、trueと表示されるものを**すべて**選びなさい。

```
1.  var str1 = "FLM";
2.  var str2 = new String("FLM");
```

A) console.log(str1 == str2);
B) console.log(str1 === str2);
C) console.log(typeof str1 == typeof str2);
D) console.log(str1 instanceof String);
E) console.log(str2 instanceof String);

typeof演算子は変数の型を返すよ。str1はstring、str2はobjectだよ。
instanceof演算子はオブジェクトの型をbooleanで判定するよ。
str1はネイティブなのでfalseが返されるんだ。

スコープ

変数や関数を呼び出せる範囲

- 宣言の場所によって以下の３種類に分かれる
  - グローバルスコープ
  - ローカルスコープ
  - ブロックスコープ

```
1. var global = “グローバル変数”;
2.
3. function func(){
4.     var local = “ローカル変数”;
5.     console.log(global);
6. }
7.
8. func();                    //globalと出力
9. console.log(local); //エラーが発生
```

グローバル変数は
どこからでも
参照できる

ローカル変数は
宣言した関数内でのみ
参照できる

巻き上げ

- 変数や関数は、スコープの先頭で宣言されたものとして動作する
  - 変数

```
1.value = “Hello!”;
2.var value;
3.
4.console.log(value);
```

=

```
1.var value;
2.value = “Hello!”;
3.
4.console.log(value);
```

  - 関数

```
1.func();
2.
3.function func(){
4.    console.log(“Hello!”);
5.}
```

=

```
1.function func(){
2.    console.log(“Hello!”);
3.}
4.
5.func();
```

サンプル問題２

■ 以下のソースコードを実行した結果、コンソール画面に表示されるものとして、正しいものを選びなさい。

```
1.  var val = “A”;
2.
3.  function print(){
4.      console.log(val);
5.      var val = “B”;
6.      return val;
7.  }
8.
9.  console.log(print());
10. console.log(val);
```

A) A B A

B) B B A

C) Undefined B A

D) Undefined B B

解答２

■ 以下のソースコードを実行した結果、コンソール画面に表示されるものとして、正しいものを選びなさい。

```
1.  var val = “A”;
2.
3.  function print(){
4.      console.log(val);
5.      var val = “B”;
6.      return val;
7.  }
8.
9.  console.log(print());
10. console.log(val);
```

＝

```
function print(){
    var val;
    console.log(val);
    val = “B”;
    return val;
 }
```

A)　A B A

B)　B B A

C)　Undefined B A

D)　Undefined B B

５行目の変数valは、print()関数の
先頭で宣言したことになるよ！

【参考】ブロックスコープ

letで宣言した変数はブロックスコープになる

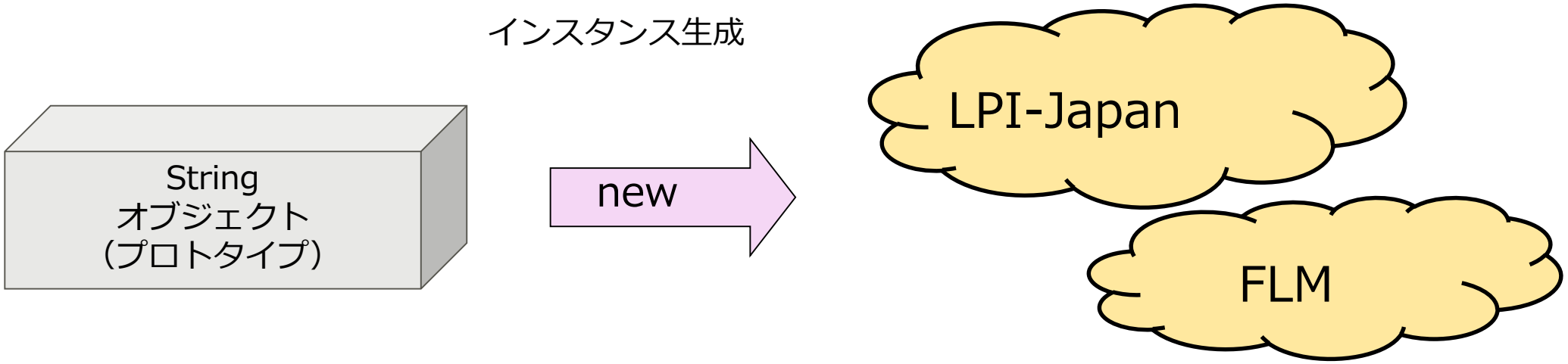
```
1. var score = 95;
2.
3. if(score >= 70){
4.     let message = “合格です”;
5. }
6.
7. console.log(message);    //エラー
```

ブロックスコープは
ブロック内だけで
参照が可能

巻き上げも起こらないため、varよりも処理が読み解きやすい。

プロトタイプベースのオブジェクト

オブジェクトからインスタンス生成



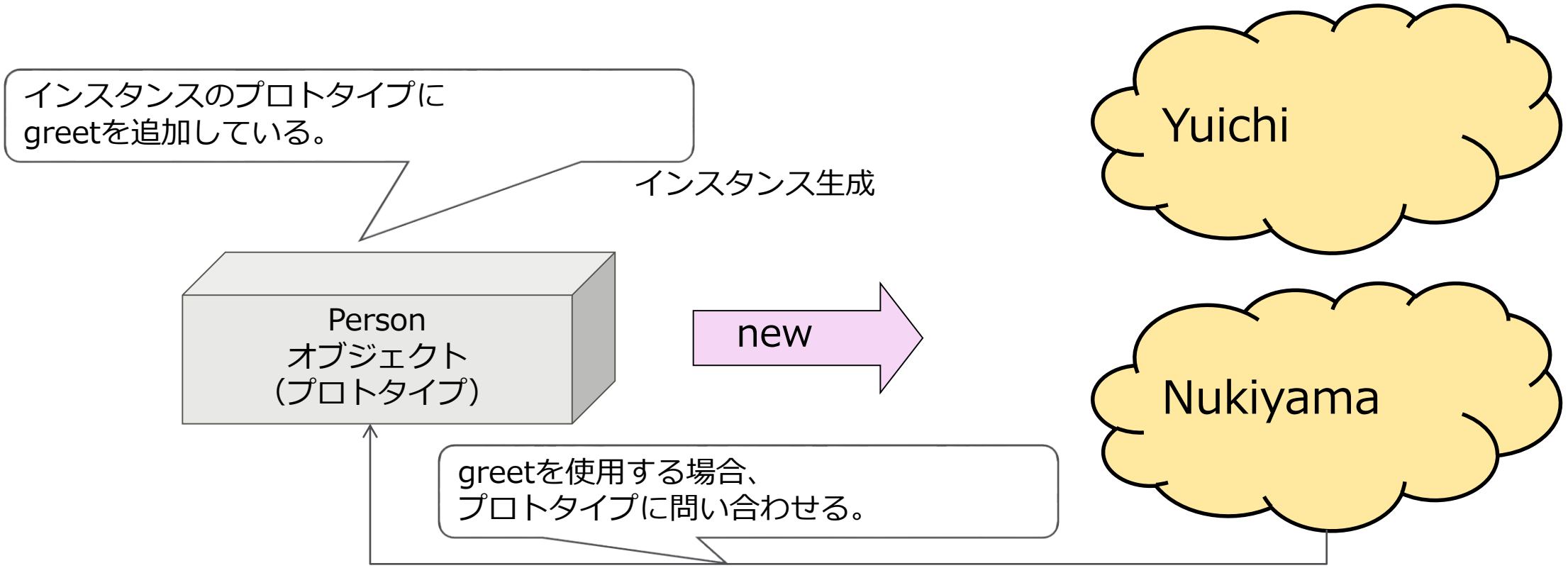
- オブジェクトは拡張可能
- インスタンスはオブジェクト（プロトタイプ）のプロパティ（データ）とメソッド（処理）を引き継ぐ。

オブジェクトの動的な拡張

```
1.  Date.prototype.getJaFullYear = function () {
2.   return this.getFullYear() + "年";
3.  }
4.  var now = new Date();
5.  console.log(now.getJaFullYear());
```

- ビルドインブジェクトも拡張可能
  →仕様がわかりにくくなるため、直接拡張はしないほうが良い

prototypeプロパティ



■ プロトタイプにメソッドを追加しないと、インスタンスごとにメソッドがコピーされてしまう

オブジェクト

```
1.  function Person(x, y) {
2.     this.name = x;
3.     this.age = y
4.  }
5.  Person.prototype.greet = function() {
6.    console.log("I'm " + this.name + this.age + " years old");
7.  }
8.
9.  var me = new Person("Yuichi", 10);
10. me.greet();
```

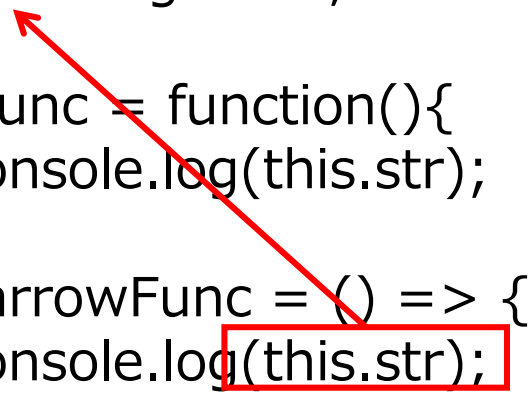
- コンストラクタは関数として定義する
- 普通の関数と見分けるため、頭文字を大文字にすることが多い

thisの値

- thisキーワードの値は**実行コンテキスト**に依存する
- **グローバル領域**では、thisは**グローバルオブジェクト**を示す
- 関数内での値は以下の５通り
  - 非strictモード
    グローバルオブジェクト（ブラウザであればwindowオブジェクト）
  - strictモード
    実行コンテキストでthisが指定されない場合、undefined
  - オブジェクトのメソッド内
    そのメソッドを持つオブジェクト
  - コンストラクタ内
    new演算子で生成されたオブジェクト
  - コールバック関数内
    コールバック関数を呼び出している関数を持つオブジェクト

【参考】ECMAScript2015のアロー関数

```
1.  var str = “global”;
2.
3.  var func = function(){
4.      console.log(this.str);
5.  }
6.  var arrowFunc = () => {
7.      console.log(this.str);
8.  }
9.
10. var obj = {
11.     str : “object”,
12.     f  : func,
13.     af : arrowFunc
14. }
15.
16. obj.f();   //objectと表示
17. obj.af(); //globalと表示
```



アロー関数内ではthisの値が
宣言時のコンテキストによって決まる。

【参考】ECMAScript2015のclassで作成

```
1. class Person2 {
2.     constructor(x, y) {
3.         this.name = x;
4.         this.age = y;
5.     }
6.     greet() {
7.         console.log("I'm " + this.name + this.age + " years old.");
8.     }
9.}
```

コードの意味としては、functionオブジェクトを使用した
オブジェクト定義と同じ。
OO言語経験者にはこちらの方が分かりやすい

サンプル問題 3

■ 以下のソースコードを実行した際に、**eオブジェクト自体に定義されている**プロパティやメソッドを**すべて**選択しなさい。

```
1.function Employee() {
2.    this.name = "foge";
3.    this.dept = "Development";
4.    this.work = function() {
5.        console.log("do test");
6.    }
7.}
8.Employee.prototype.manage = function() {
9.    console.log("manage");
10.}
11.var e = new Employee();
```

A) name
B) dept
C) work
D) manage

解答 3

■ 以下のソースコードを実行した際に、**eオブジェクト自体に定義されている**プロパティやメソッドを**すべて**選択しなさい。

```
1.function Employee() {
2.    this.name = "foge";
3.    this.dept = "Development";
4.    this.work = function() {
5.        console.log("do test");
6.    }
7.}
8.Employee.prototype.manage = function() {
9.    console.log("manage");
10.}
11.var e = new Employee();
```

manageはeオブジェクトの
プロトタイプに定義されているよ。
Object.getOwnPropertyNames()
で確認もできる。

A) dept　　C) work

B) name　　D) manage

Indexed Database API

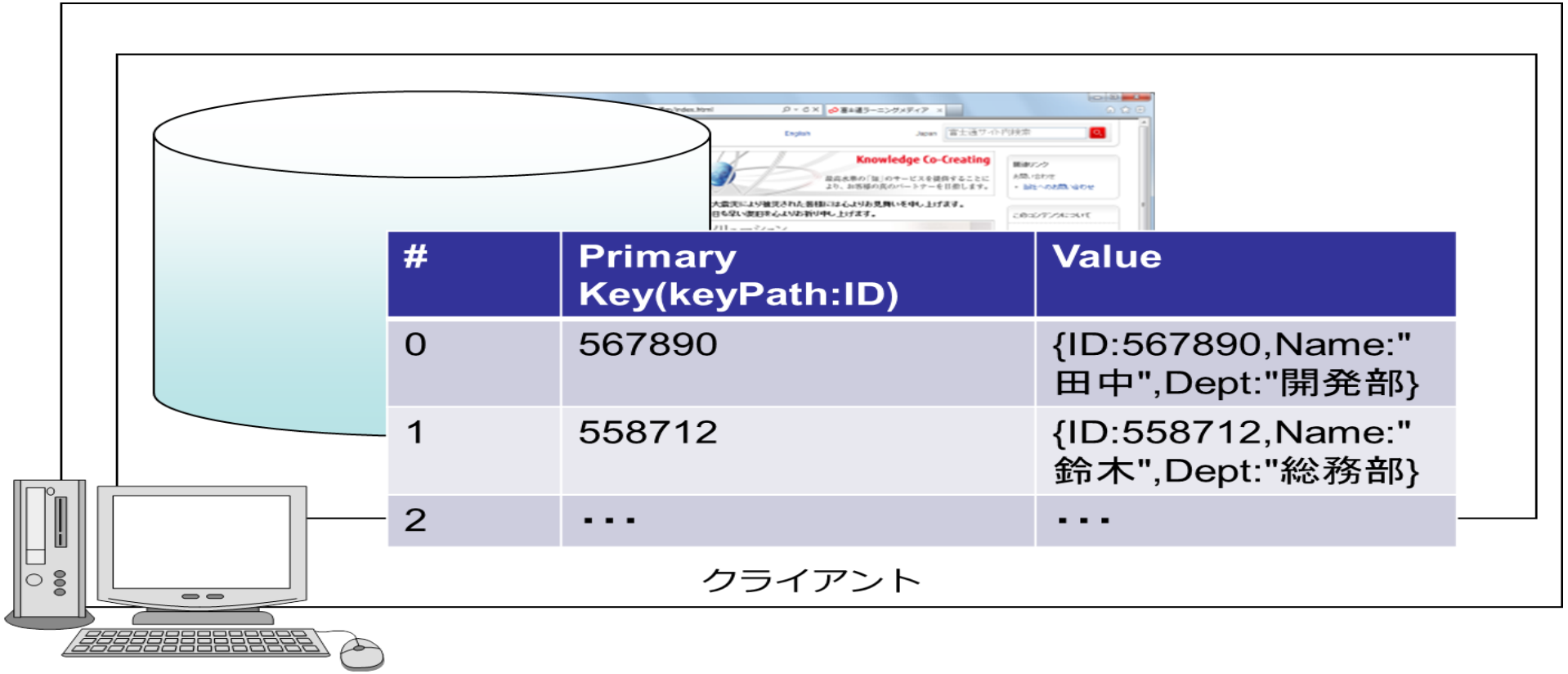
主なデータ保存系API

データをブラウザに保存できる

- Web Storage
  - ローカルストレージ
  - セッションストレージ
- Indexed Database API

Indexed Database APIとは

ブラウザにデータを保存できるデータベースに関する仕様

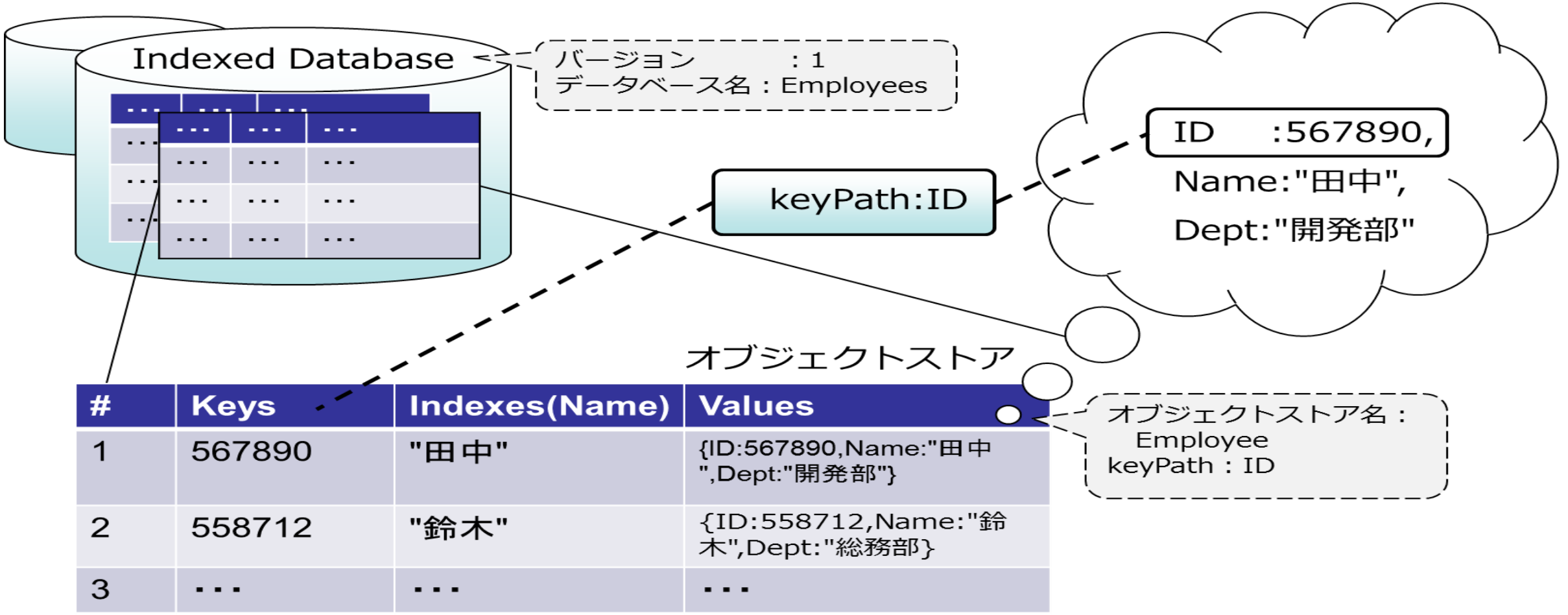


#	Primary Key(keyPath:ID)	Value
0	567890	{ID:567890,Name:"田中",Dept:"開発部}
1	558712	{ID:558712,Name:"鈴木",Dept:"総務部}
2	・・・	・・・

Indexed Database APIの特徴

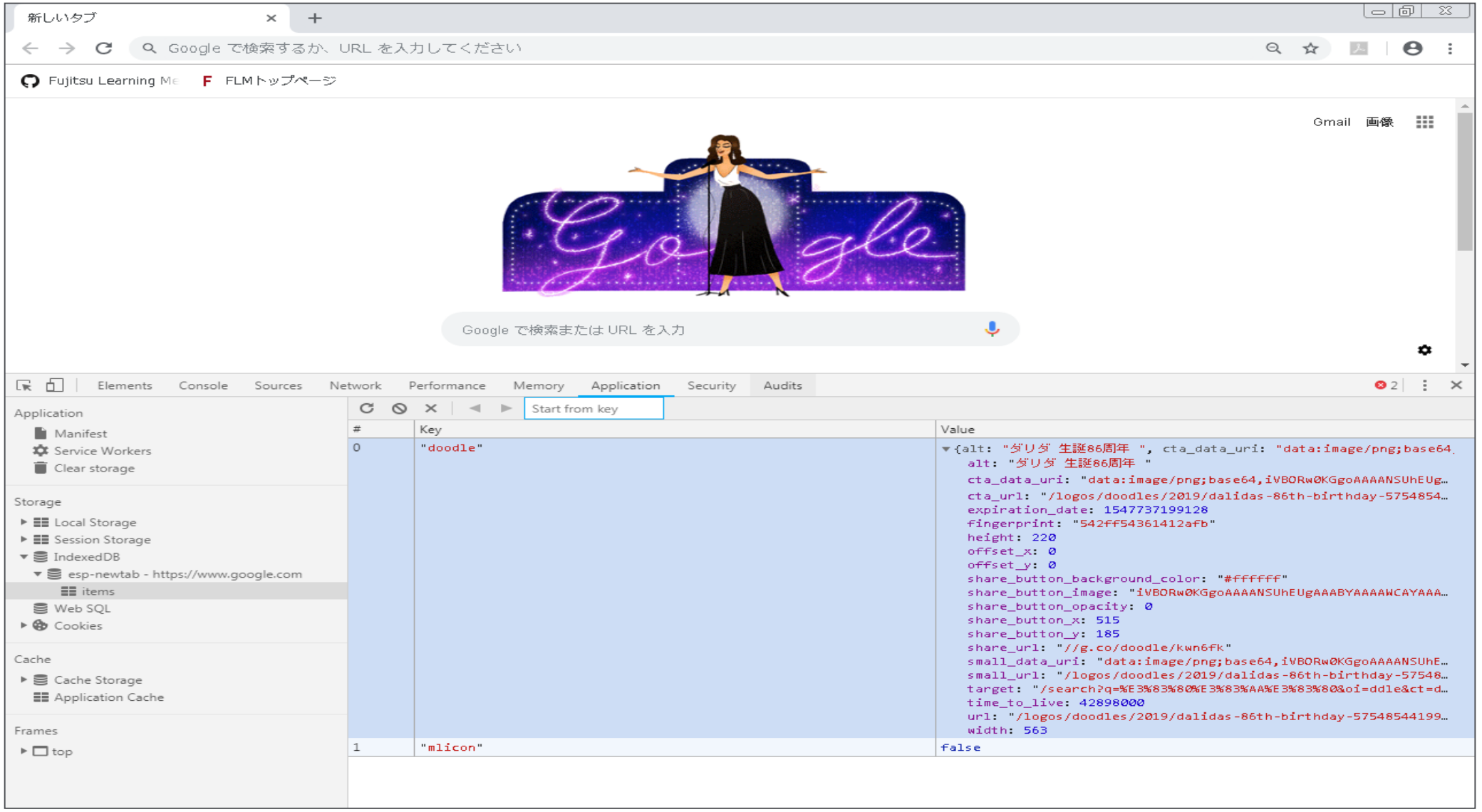
- Indexed Database 内の**オブジェクトストア**にデータを保持している
- **トランザクション**内でデータを操作できる
- **インデックス**を使用して検索できる
- **非同期**で実行される
- Indexed Database の操作に **SQL文を使用しない**
- **クロスドメイン**のアクセスが制限されている

Indexed Databaseの構造



#	Keys	Indexes(Name)	Values
1	567890	"田中"	{ID:567890,Name:"田中",Dept:"開発部"}
2	558712	"鈴木"	{ID:558712,Name:"鈴木",Dept:"総務部}
3	...	...	...

Indexed Databaseの確認



サンプル問題４

■ 以下のソースコードの説明として、正しいものを**2つ**選びなさい。

```
1. var request = indexedDB.open("zoo", 1);

2. request.onerror = function (event) {
3.     console.log("データベースに接続できませんでした");
4. };
5.
6. request.onupgradeneeded = function (event) {
7.     var db = event.target.result;
8.     var store = db.createObjectStore("animal", {keyPath: "kind"});
9. };
10.
11.request.onsuccess = function (event) {
12.    var db = event.target.result;
13.};
```

A) open()メソッドが初めて実行されると、zooデータベースが作成される

B) open()メソッドの第2引数には、生成するデータベースの数を指定している

C) データベースの作成、接続が失敗すると、エラーが発生する

D) 既に同じ名前で新しいバージョンのデータベースが存在しても、エラーは発生しない

解答４

■ 以下のソースコードの説明として、正しいものを**2つ**選びなさい。

```
1. var request = indexedDB.open("zoo", 1);

2. request.onerror = function (event) {
3.     console.log("データベースに接続できませんでした");
4. };
5.
6. request.onupgradeneeded = function (event) {
7.     var db = event.target.result;
8.     var store = db.createObjectStore("animal", {keyPath: "kind"});
9. };
10.
11.request.onsuccess = function (event) {
12.    var db = event.target.result;
13.};
```

A) open()メソッドが初めて実行されると、zooデータベースが作成される
B) open()メソッドの第2引数には、生成するデータベースの数を指定している
C) データベースの作成、接続が失敗すると、エラーが発生する
D) 既に同じ名前で新しいバージョンのデータベースが存在しても、エラーは発生しない

Indexed Database ソースコード

■ データベースへの接続、切断

メソッド名	説明
open(“DB名”,DBのバージョン)	データベースへ接続する
deleteDatabase()	データベースを削除する

■ 接続後に使用できるプロパティ、イベントハンドラ

プロパティ	説明
error	処理が失敗した場合のDOMErrorオブジェクト
result	データベースへの処理の結果

イベントハンドラ	説明
onerror	エラー発生時に発火
onsuccess	処理完了時に発火
onupgradeneeded	データベース新規作成、バージョンアップ時に発火

データ保存系APIの比較

	Cookie	Web Storage	Indexed Database	WebSQL
保存容量	4KB	5MB-10MB	5MB-10MB	5MB-10MB
保存期間	有期限	無期限/セッション	無期限	無期限
送受信	リクエストごと	JSで操作時のみ	JSで操作時のみ	JSで操作時のみ
データ形式	文字列	文字列	ネイティブ/ オブジェクト	ネイティブ/ オブジェクト
非同期	×	×	○	○
特徴	設定によっては、HTTP通信でも送受信される。	シンプルなAPIで、大容量データを保存できる。	複雑なデータを扱える。APIが複雑なため、実装が少々難しい。	SQL文ライクの文法でデータを操作できる。
備考	最も実装が進んでいる。		DBとは操作方法が異なるので注意。	仕様策定が中止されている。

サンプル問題５

Indexed DatabaseとWeb Storageの説明として、**誤っているもの**を選びなさい。

A. どちらもキーと値のペアを保存する
B. Indexed Databaseのみインデックスが使用できる
C. どちらもトランザクション処理ができる
D. どちらも同一オリジンのみデータアクセスが可能である

解答５

Indexed DatabaseとWeb Storageの説明として、**誤っているもの**を選びなさい。

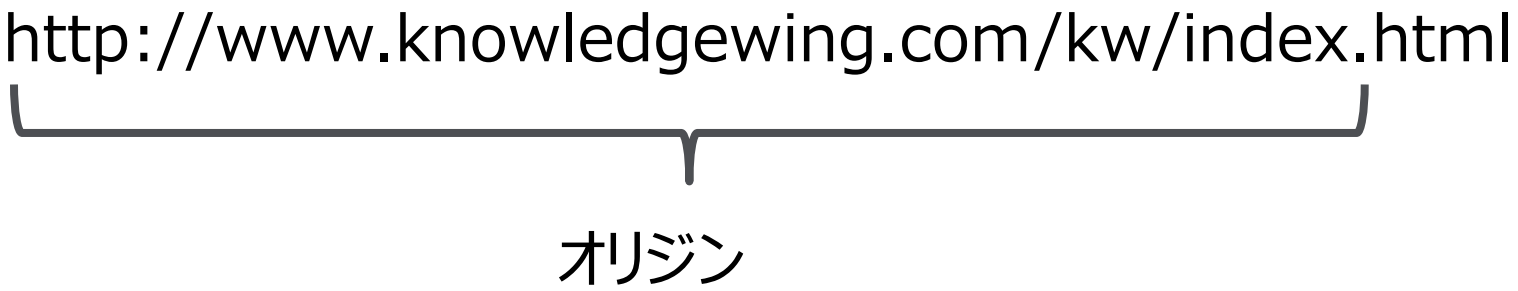
A. どちらもキーと値のペアを保存する
B. Indexed Databaseのみインデックスが使用できる
C. どちらもトランザクション処理ができる
D. どちらも同一オリジンのみデータアクセスが可能である

トランザクションは、Web Storageでは使用できないよ。

クロスオリジン制約とCORS

オリジン

URLのうち、「スキーム://ホスト名:ポート番号」までをオリジンと呼ぶ



サンプル問題６

「http://www.knowledgewing.com/」と同一オリジンのものを選びなさい。

A. http://www.knowledgewing.com:8000/
B. http://www.knowledgewing.com/kw/index.html
C. http://knowledgewing.com/
D. https://www.knowledgewing.com/

解答６

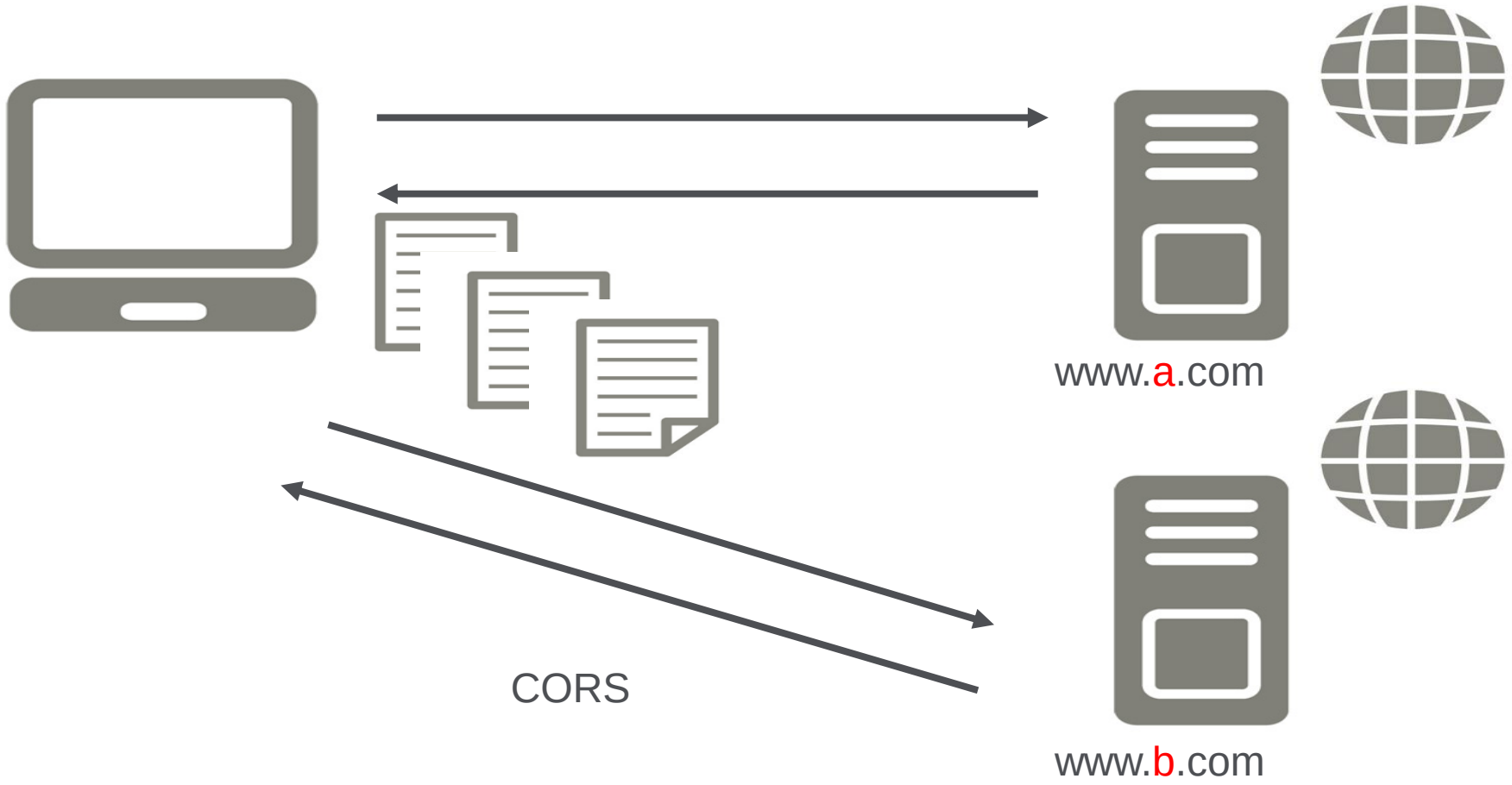
「http://www.knowledgewing.com/」と同一オリジンのものを選びなさい。

A. http://www.knowledgewing.com:8000/
B. http://www.knowledgewing.com/kw/index.html
C. http://knowledgewing.com/
D. https://www.knowledgewing.com/

ポート番号、ホスト名、プロトコルが異なるものは、同一オリジンとみなされないよ

CORS（Cross-Origin Resource Sharing）

異なるオリジンのリソースにアクセスする権限を取得する仕組み



CORSの許可

HTTPヘッダでCORSの許可設定をする

ヘッダ名	説明
Access-Control-Request-Method	リクエストで使用するリクエストメソッドをサーバに通知するリクエストヘッダ
Access-Control-Request-Headers	リクエストで使用するヘッダをサーバに通知するリクエストヘッダ
Access-Control-Allow-Origin	必須。アクセスを許可するオリジンを指定するレスポンスヘッダ。*の場合、すべてのオリジンからのアクセスを許可する
Access-Control-Allow-Methods	許可するリクエストメソッドを指定するレスポンスヘッダ
Access-Control-Allow-Headers	リクエストで使用できるヘッダを指定するレスポンスヘッダ

サンプル問題６

CORSリクエスト関連のヘッダのうち、レスポンスヘッダであるものをすべて選びなさい。

A. Access-Control-Request-Methodヘッダ
B. Access-Control-Request-Headersヘッダ
C. Access-Control-Allow-Originヘッダ
D. Access-Control-Allow-Methodsヘッダ
E. Access-Control-Allow-Headersヘッダ

サンプル問題６

CORSリクエスト関連のヘッダのうち、レスポンスヘッダであるものをすべて選びなさい。

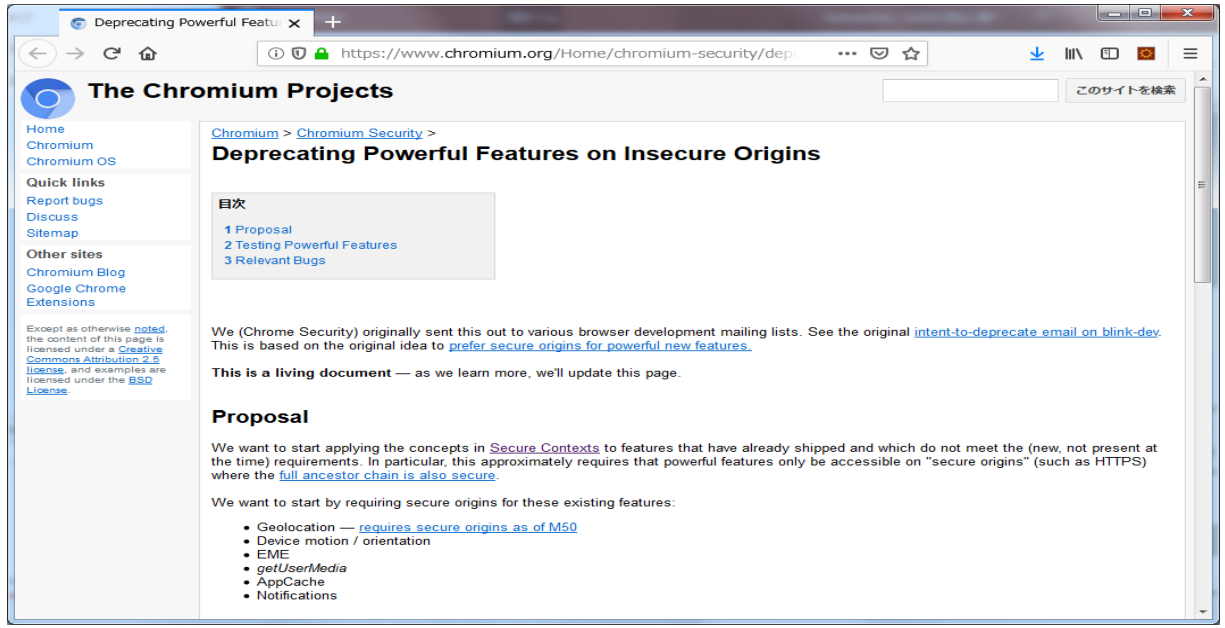
A. Access-Control-Request-Methodヘッダ
B. Access-Control-Request-Headersヘッダ
C. Access-Control-Allow-Originヘッダ
D. Access-Control-Allow-Methodsヘッダ
E. Access-Control-Allow-Headersヘッダ

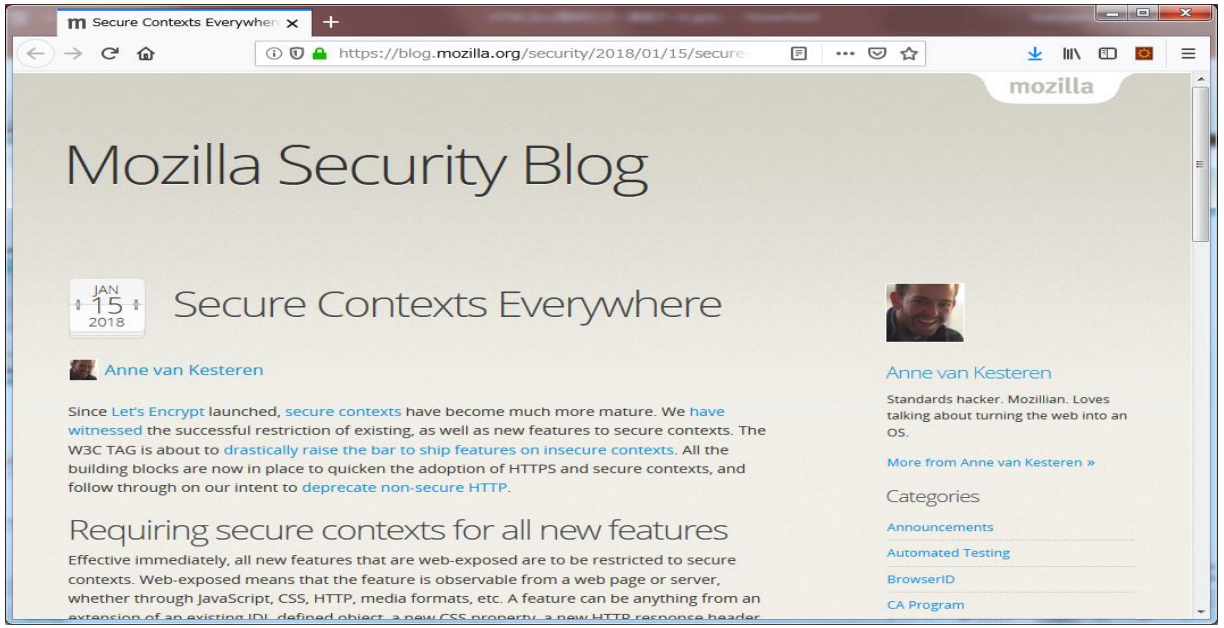
CORS許可に必要なのは
Access-Control-Allow-Originヘッダ
だけだよ！

セキュリティモデルとSSLの関係

HTTPSとJavaScript

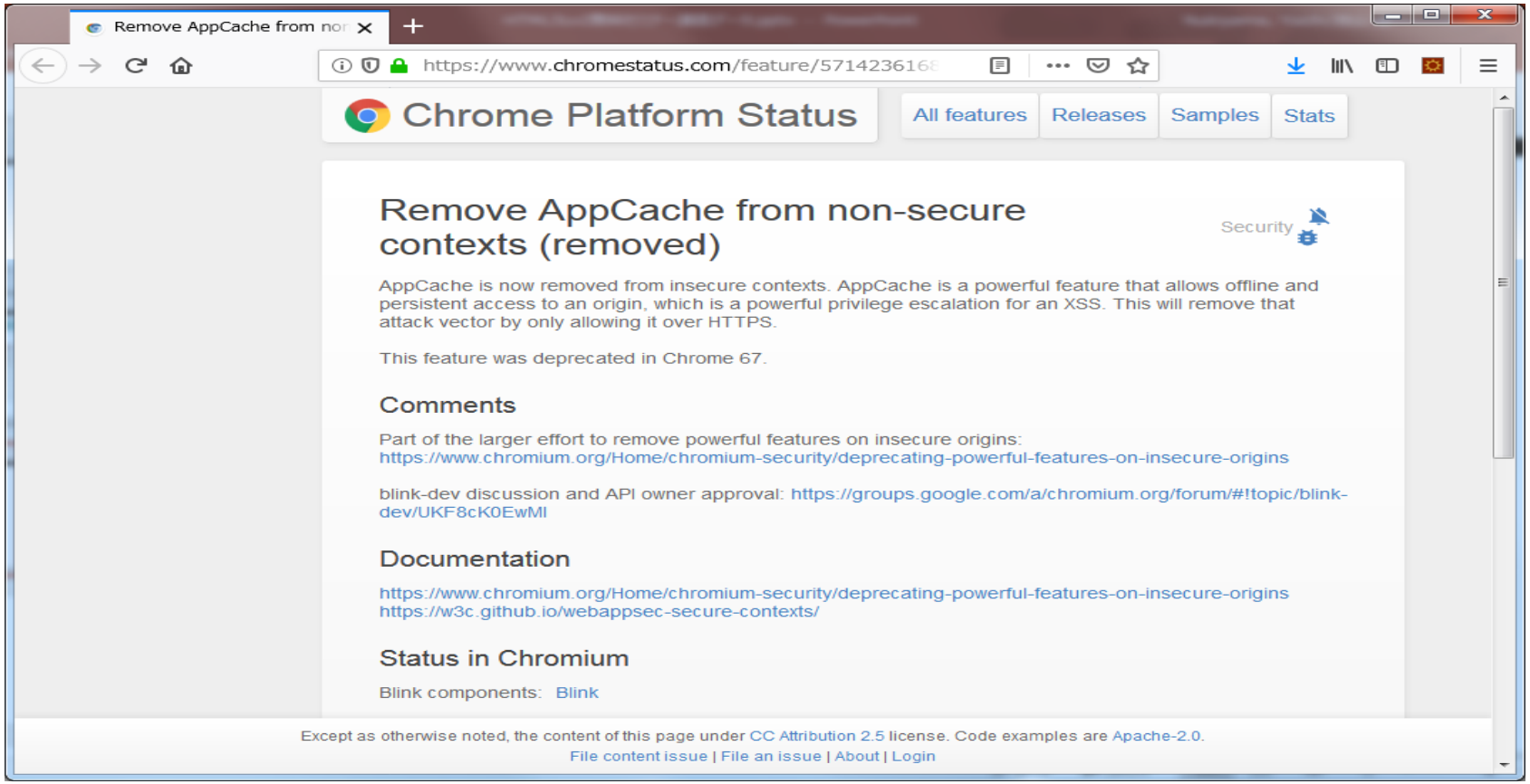
仕様、またはブラウザの設定により、一部の機能を使用する場合、HTTPSが必須に





HTTPSが必要な主なAPI

- Service Workers
- Geolocation API
- AppCache



サンプル問題７

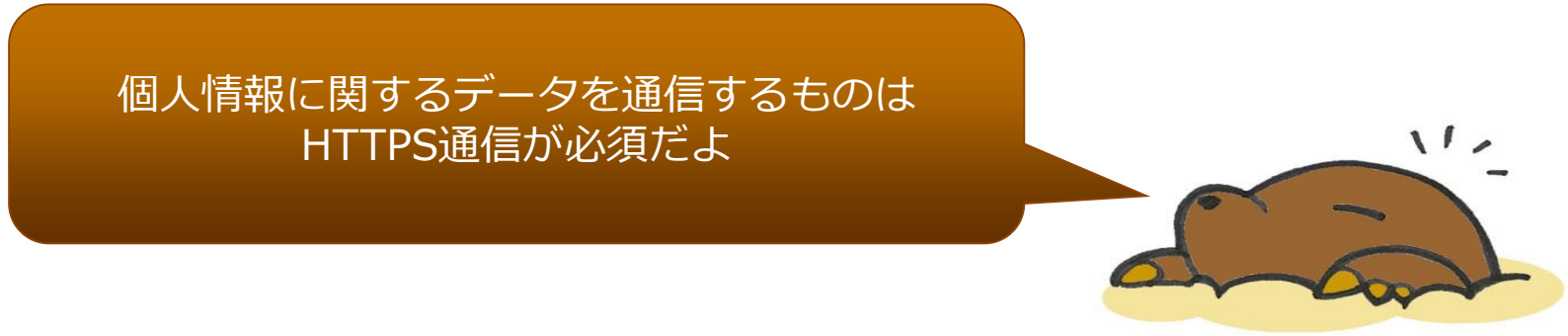
HTTPS通信でしか動作しない可能性があるAPIとして、正しいものをすべて選びなさい。

A. WebSocket
B. Geolocation API
C. DeviceOrientation Event
D. Service Workers

サンプル問題７

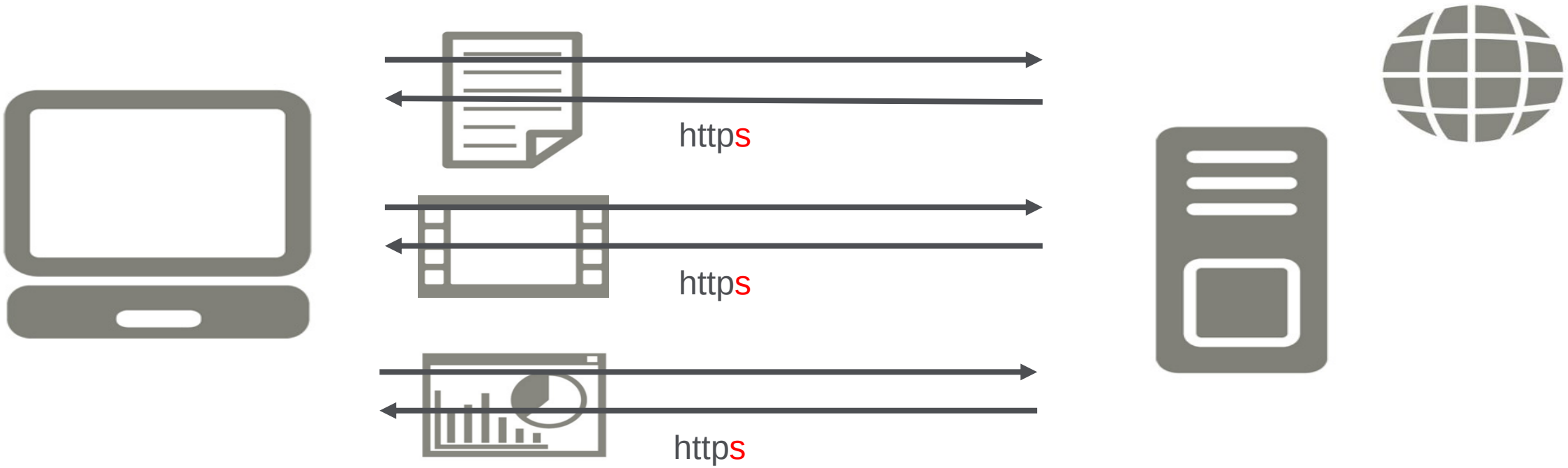
HTTPS通信でしか動作しない可能性があるAPIとして、正しいものをすべて選びなさい。

A. WebSocket
B. Geolocation API
C. DeviceOrientation Event
D. Service Workers



Secure Contexts

HTTPSなどの暗号化された通信を介して安全に提供されているコンテンツ



サンプル問題８

Secure Contextsとみなされるものをすべて選びなさい。なお、Webページのリクエスト／レスポンスにはHTTPS通信を用いているものとする。

A. Webページ内のiframe要素のsrc属性でHTTPS通信をしている
B. Webページ内でXMLHttpRequestによるHTTPS通信をしている
C. Webページ内のimg要素のsrc属性でHTTP通信をしている
D. Webページ内のscript要素のsrc属性でHTTP通信をしている

サンプル問題８

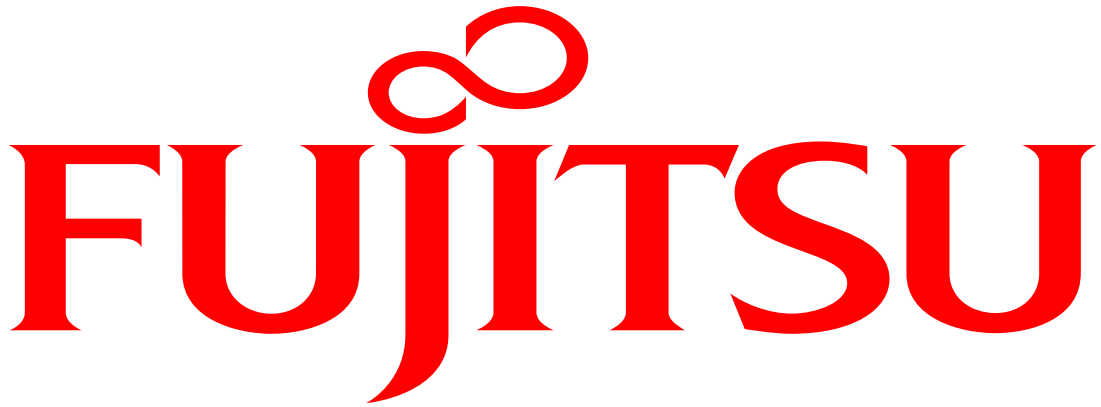
Secure Contextsとみなされるものをすべて選びなさい。なお、Webページのリクエスト／レスポンスにはHTTPS通信を用いているものとする。

A. Webページ内のiframe要素のsrc属性でHTTPS通信をしている
B. WebページでXMLHttpRequestによるHTTPS通信をしている
C. Webページ内のimg要素のsrc属性でHTTP通信をしている
D. Webページ内のscript要素のsrc属性でHTTP通信をしている

HTTPS通信しているかがポイントだね

LPI-JAPAN HTML5 Professional Certification

Open the Future with HTML5.



shaping tomorrow with you