株式会社クイシー・シー

HTML5プロフェッショナル認定試験
レベル2　無料技術セミナー

講師プロフィール

会社概要

株式会社ケイ・シー・シー

http://www.kcc.co.jp/

講師紹介

福田　浩之

Linux、ネットワーク、セキュリティ、HTML5、JavaScript、Javaなど幅広い分野の研修を実施。

アウトライン

- HTML5プロフェッショナル認定資格 レベル2 試験概要
- 技術解説項目

1.ストレージ

・WebStorage

2.通信

・Messaging

・WebSocket

・Server-Sent Events

3.セキュリティモデル

・クロスオリジン制約とCORS

株式会社ケイ・シー・シー

HTML5プロフェッショナル認定資格 レベル2
試験概要

HTML5プロフェッショナル認定資格の種類

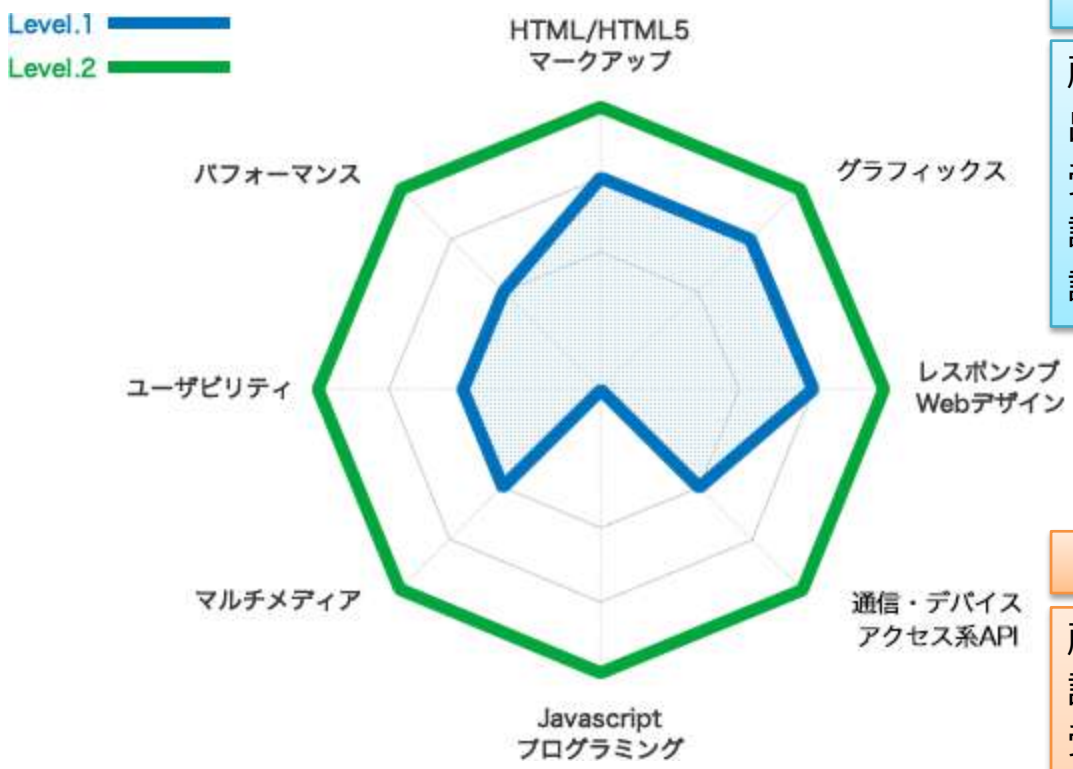
HTML 5 Level.1

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。

HTML 5 Level.2

システム間連携や最新のマルチメディア術に対応した
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。

レベル１とレベル２の資格体系



試験実施形式

コンピュータベーストテスト（CBT）で実施します。
マウスによる選択方式がほとんどですが、キーボード入力問題も多少出題されます。
※ 試験は「ピアソンVUE」より配信されています。

HTML5プロフェッショナル認定試験レベル１

所要時間：90分（アンケート等の時間を含む）
出題数：約60問
受験料：¥15,000 （税別）
認定条件： HTML5 レベル1試験の合格
認定の有意性の期限：5年間

HTML5プロフェッショナル認定試験レベル２

所要時間：90分（アンケート等の時間を含む）
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

認定の証明

認定証

認定カード

認定証・認定カードは、認定されてから2週間程度でご登録されたご住所にお届けしています。

認定者ロゴ（名刺用）

認定者ロゴは、認定後すぐに名刺等でご利用していただけます。

HTML5 レベル2　傾向と対策

- CBT（Computer Based Testing）試験
  - コンピュータを操作して問題に解答
  - 試験中、問題は何度も繰り返し参照可能
  - 試験終了と同時に結果が判明

- 試験時間の有効活用
  - 90分（機密保持契約とアンケートの時間を含む）で40～45問の問題
  - 四者択一または五者択一、複数選択
  - 問題はしっかり読む
    - あやふやな問題はチェックをつけて、後から解答する
    - 全体的に見直す時間を確保する

株式会社ケイ・シー・シー

技術解説項目

1.ストレージ

・WebStorage

ストレージ

- Web Storage
  - クライアントアプリケーションから利用可能な、ユーザエージェントがもつ永続的なデータ格納領域
- Indexed Database API
  - 高度なデータ管理（検索、トランザクション、カーソル　etc)が可能なインデックス付きのデータベースを扱うAPI
- File API
  - HTML文章からローカルファイルに直接アクセスするための機能を提供

Web Storage

- Web Storage (Second Edition)【2016/4/19　勧告】
  http://www.w3.org/TR/webstorage/

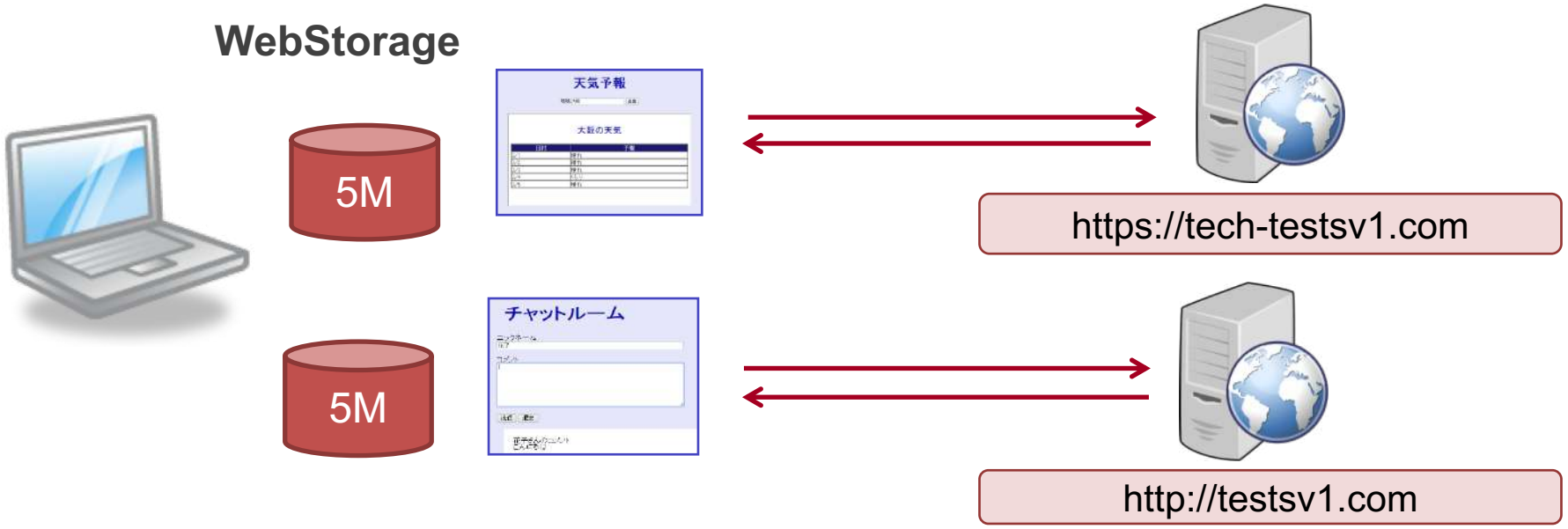
  - クライアントアプリケーションから利用可能な、ユーザエージェントがもつ永続的なデータ格納領域

    - ローカルストレージ（local storage)
      - データは永続的に保存
      - ファイルで保持する*cookie*に類似

    - セッションストレージ（session storage）
      - データは一時的に保存
      - メモリで保持する*cookie*に類似

Web Storage

- オリジンごとに5Mバイト（推奨）の容量をもつ



Web Storage

- オリジンとは『URLの「スキーム」「ホスト」「ポート」の3つの組み合わせ』

http://testsv1.com：80

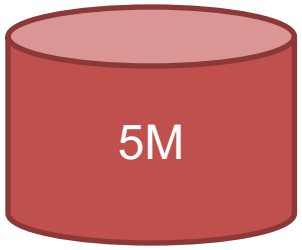
スキーム　　ホスト　　ポート番号

http://testsv1.com/dira/test.html

http://testsv1.com/dirb/appliance.html

同一オリジンとして認識

- キー、バリュー型のシンプルなデータベース



キー	バリュー
name	セマン
tel	0612345678
1	{ “name”:“セマン” , ”tel” : ”0612345678” }

Web Storage

- WindowオブジェクトのStorage属性を参照
  - ローカルストレージ
    - StorageオブジェクトはWindowオブジェクトのlocalStorage属性を用いて参照

```
window.localStrage
```

  - セッションストレージ
    - StorageオブジェクトはWindowオブジェクトのsessionStorage属性を用いて参照

```
window.sessionStrage
```

※ localStorageとsessionStorageはデータの参照や操作方法は同じです。
※windowオブジェクトは省略可能です。

Web Storage

- **Webストレージへのデータの保存**

sessionStorage.setItem(key,value)

setItem(key,value)

Webストレージにデータを保存

key　：キーとなる値（文字列）

value：バリューとなる値（文字列）

- **Webストレージに格納したデータの参照**

sessionStorage.getItem(key)

getItem(key)

指定したキーのバリューを参照

戻り値：キーに対応する値

引数に指定したキーがWebStorageになければnullを戻す

key　：キーとなる値（文字列）

Web Storage

reference その他の記述方法

- データの保存

sessionStorage.key = value
sessionStorage[key] = value

- データの参照

sessionStorage.key
sessionStorage[key]

Web Storage

- **バリューの削除**

sessionStorage.removeItem(key)

removeItem(key)

指定したキーのバリューを削除

key　：キーとなる値（文字列）

- **ストレージ内のデータをすべて削除**

sessionStorage.clear()

clear()

Webストレージ内のすべてのデータを削除

Web Storage

- **Webストレージに保存したデータの個数を参照**

sessionStorage.length

length
Webストレージに格納したデータの個数を参照

- **キーを参照**

sessionStorage.key（index）

key（index）
指定したindexのキーの値を参照
戻り値：指定したキー（文字列）
index　：順序番号

Web Storage

- storageイベント
  - WebStorageが更新されるとWindowオブジェクトに発生するイベント

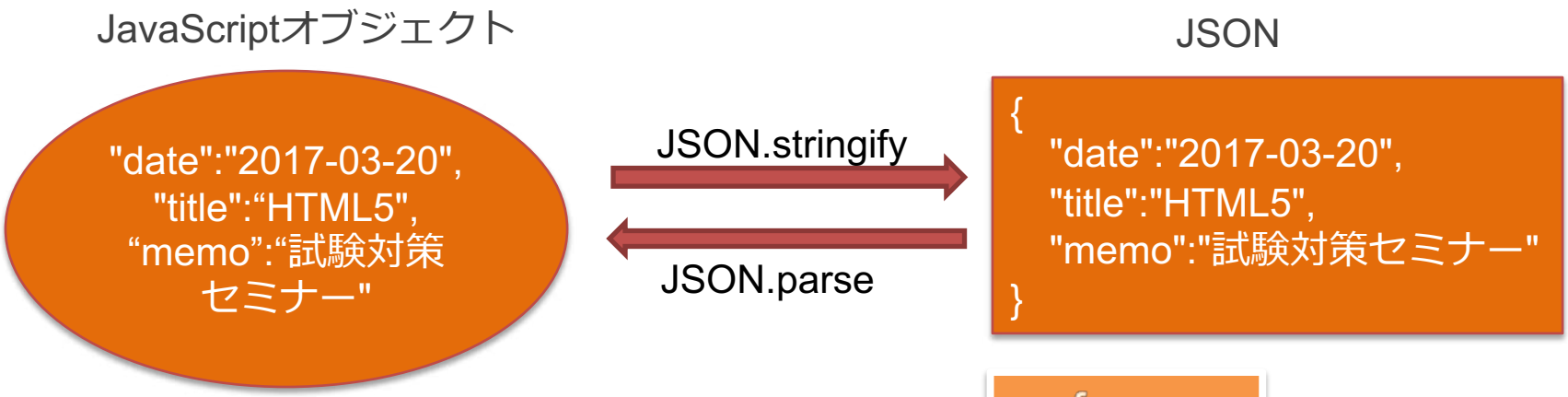
```
window.addEventListener("storage",StorageChange,false);

function StroageChange( e ){
        /* WebStorageが更新された際に行う処理 */
}
```

StorageEventオブジェクト

Web Storage

- **JavaScriptオブジェクトとJSON**
  - **JavaScriptオブジェクトは、JSONに変換して保存できる**



  - **JavaScriptオブジェクト → JSON**

```
JSON.stringify(JavaScriptオブジェクト)
```

  - **JSON → JavaScriptオブジェクト**

```
JSON.parse(JSON)
```

reference

JSON
（JavaScript ObjectNotation）
数値・文字列・配列・オブジェクトなどのデータを、文字列で表現できる軽量データ交換フォーマット

Web Storage

練習問題

WebStorageの説明として誤っているものをすべて選びなさい。

A. sessionStorageで保存したデータはブラウザを閉じると消える

B. キー、バリュー型のデータ形式で保存され、インデックスを活用し高速に検索が可能

C. http://www.testsv1.com:80とhttp://www.testsv1.com:1では異なる領域にデータが保存される

D. ローカルホスト上のファイルをBlobオブジェクトとして扱うことができる

E. localStorageで保存したデータはブラウザを閉じると消える

株式会社ケイ・シー・シー

技術解説項目

2.通信

- Messaging
- WebSocket
- Server-Sent Events

Messaging

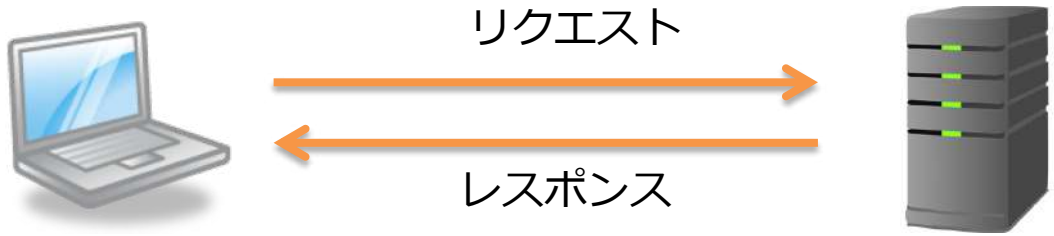
- Messaging　【2015/5/19　勧告】
  http://www.w3.org/TR/webmessaging/
  - ブラウザー上のブラウジングコンテキスト間で通信する仕組み
    - ブラウジングコンテキスト…ウィンドウやタブ等
- Messagingを利用する処理
  - クロスドキュメントメッセージング
    - 異なるオリジンと安全に通信をする仕組み
    - メインページとiframe間のデータのドキュメントメッセージング
  - チャネルメッセージング
    - チャネルを介したデータのやり取りを行う仕組み
  - サーバープッシュイベント
    - サーバーからのプッシュ通信を行う仕組み
  - WebSocket
    - リモートホストとの双方向通信を行う仕組み

WebSocket

- WebSocket（W3C）
  http://www.w3.org/TR/websockets/
  - WebSocketプロトコルを使用し、クライアント、サーバーアプリケーション間で双方向の非同期通信を行う
  - WebSoekctプロトコル（IETF）
    http://tools.ietf.org/html/rfc6455

WebSocket

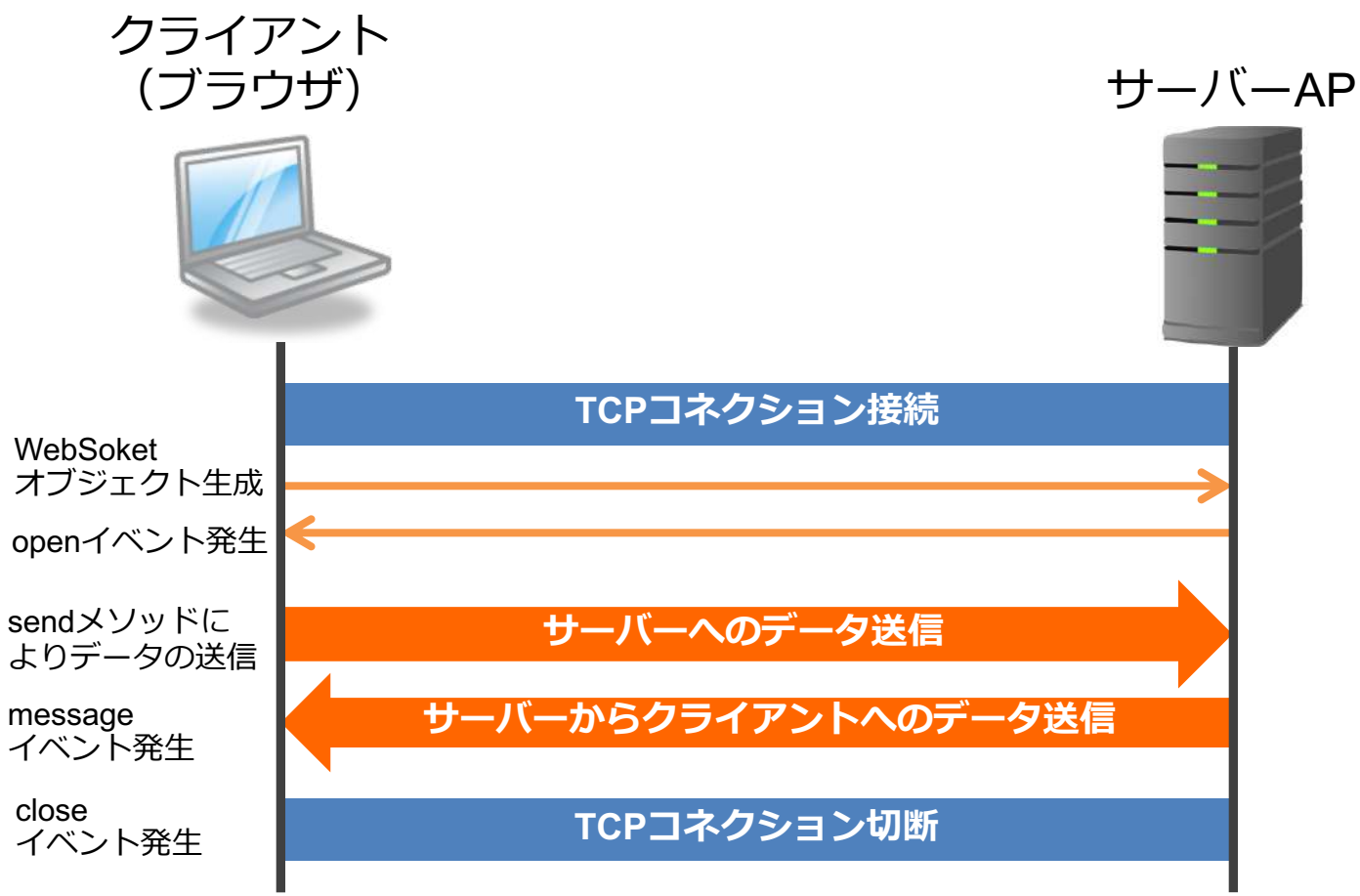
- HTTPの通信
  - リクエストとレスポンスが1：1



- WebSocketの通信
  - セッション確立後は任意のタイミングで双方向通信が可能

WebSocket

- **WebSoket通信の流れ**



WebSocket

- **WebSocketセッションの確立**
  - **WebSocketオブジェクトの生成により、サーバーアプリケーションとのセッションを確立**

```
var websocket = new WebSocket(“ws://www.testsv1.com:3000");
```

WebSocket("url",[protocols]);

WebSocketオブジェクトのコンストラクタ

url　　：サーバーアプリケーションのURL

一般的なスキームは、「ws」または「wss（セキュア通信）」

protocol：サブプロトコルの配列　※省略可

- **openイベント**
  - **接続が完了した場合、WebSocketオブジェクトにopenイベントが発生**

```
soketobj.onopen = function(){
        /* 接続が完了した際に行う処理 */
}
```

WebSocket

- **メッセージ送信**

websocket.send(data)

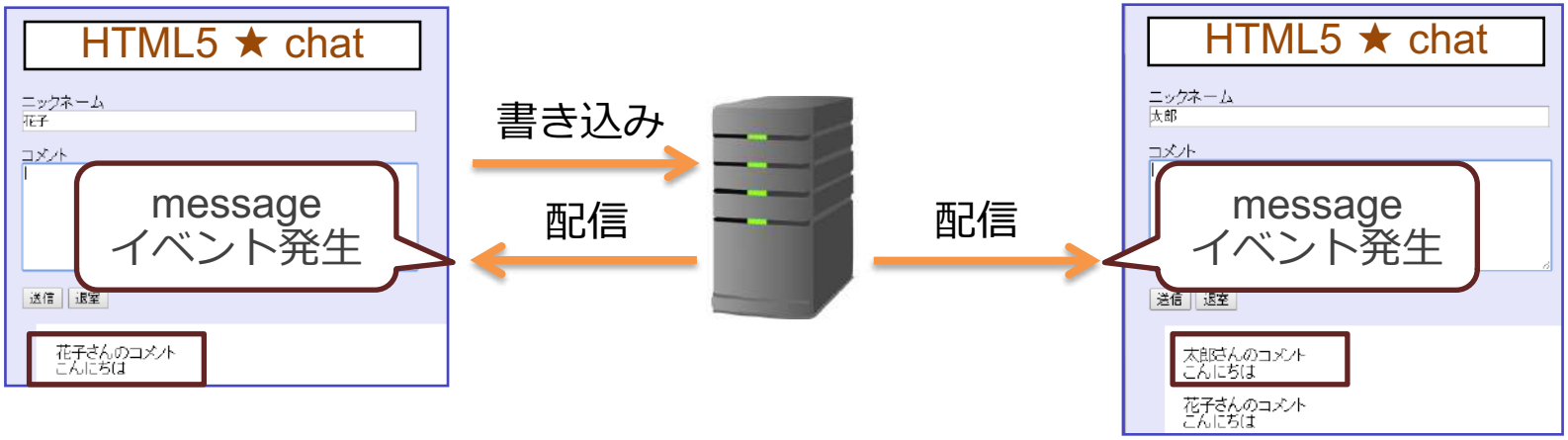
send.(data)
　　メッセージを送信
　　data　：送信データ
　　　　（文字列、Blob、ArrayBuffer、ArrayBufferView）
戻り値：true ・・・・ 成功
　　false・・・・失敗

WebSocket

- メッセージ受信
  - メッセージ受信時、WebSocketオブジェクトにmessageがイベント発生
  - 受信データの参照

```
websocket.onmessage = function(e) {
    alert(e.data);
          (略)
};
```

MessageEventオブジェクト



WebSocket

- サーバーアプリケーションとのセッション切断

websocket.close()

close([code],[reason])

サーバーアプリケーションとの接続を閉塞

code　：切断コード（数値）　※省略可

reason ：切断理由（文字列）　※省略可

- closeイベント
  - サーバーアプリケーションとのセッション切断が完了したときに発生

```
websocket.onclose=function(e){
        /* 接続を切断したときに行う処理 */
}
```

WebSocket

- サーバーアプリケーション
  - 言語はPerl　など

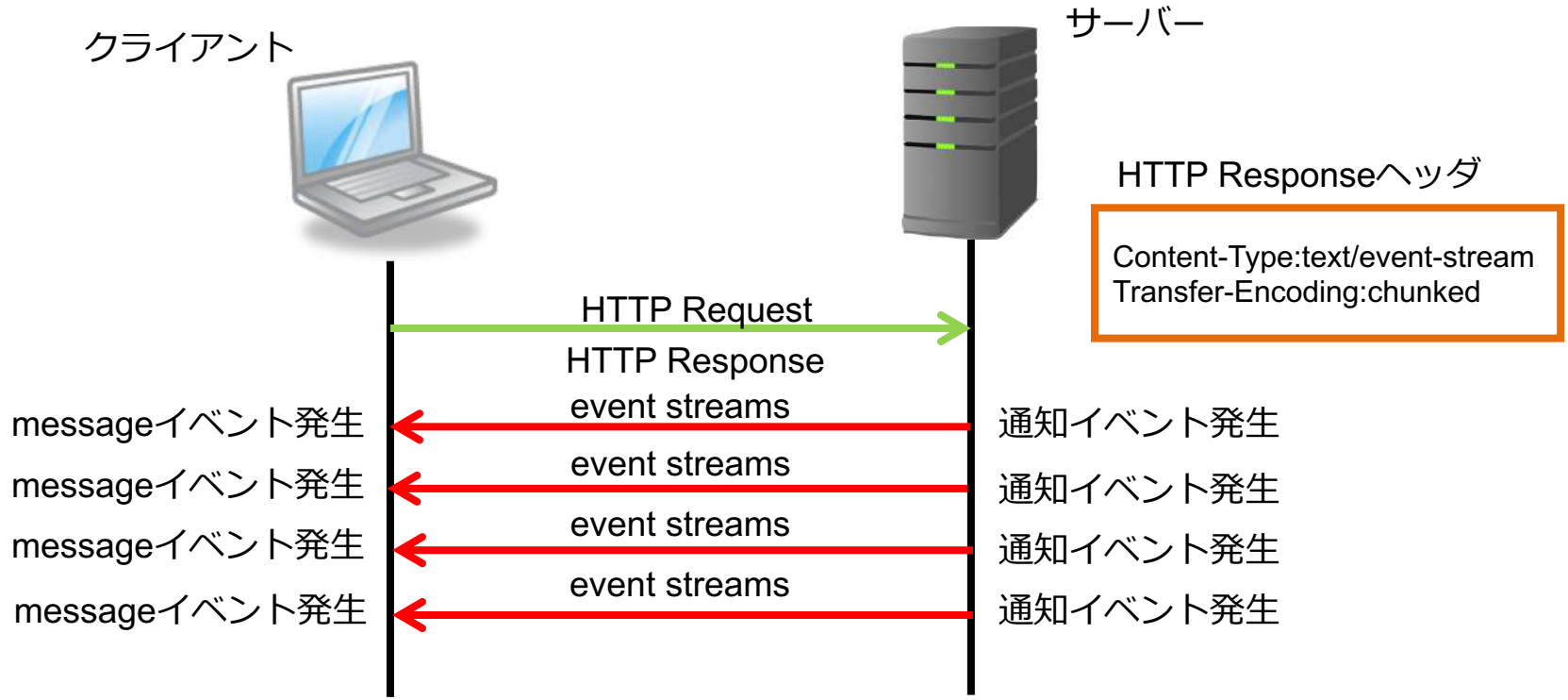
- サーバーの起動（Perl）

コマンドプロンプトから

perl サーバーアプリケーション名 daemon

ServerSentEvents

- ServerSentEvents
  - クライアントアプリケーションがサーバーアプリケーションから
    リアルタイムにメッセージを受信する仕組み
  - プッシュ型のメッセージ（イベントストリーム）を受信



ServerSentEvents

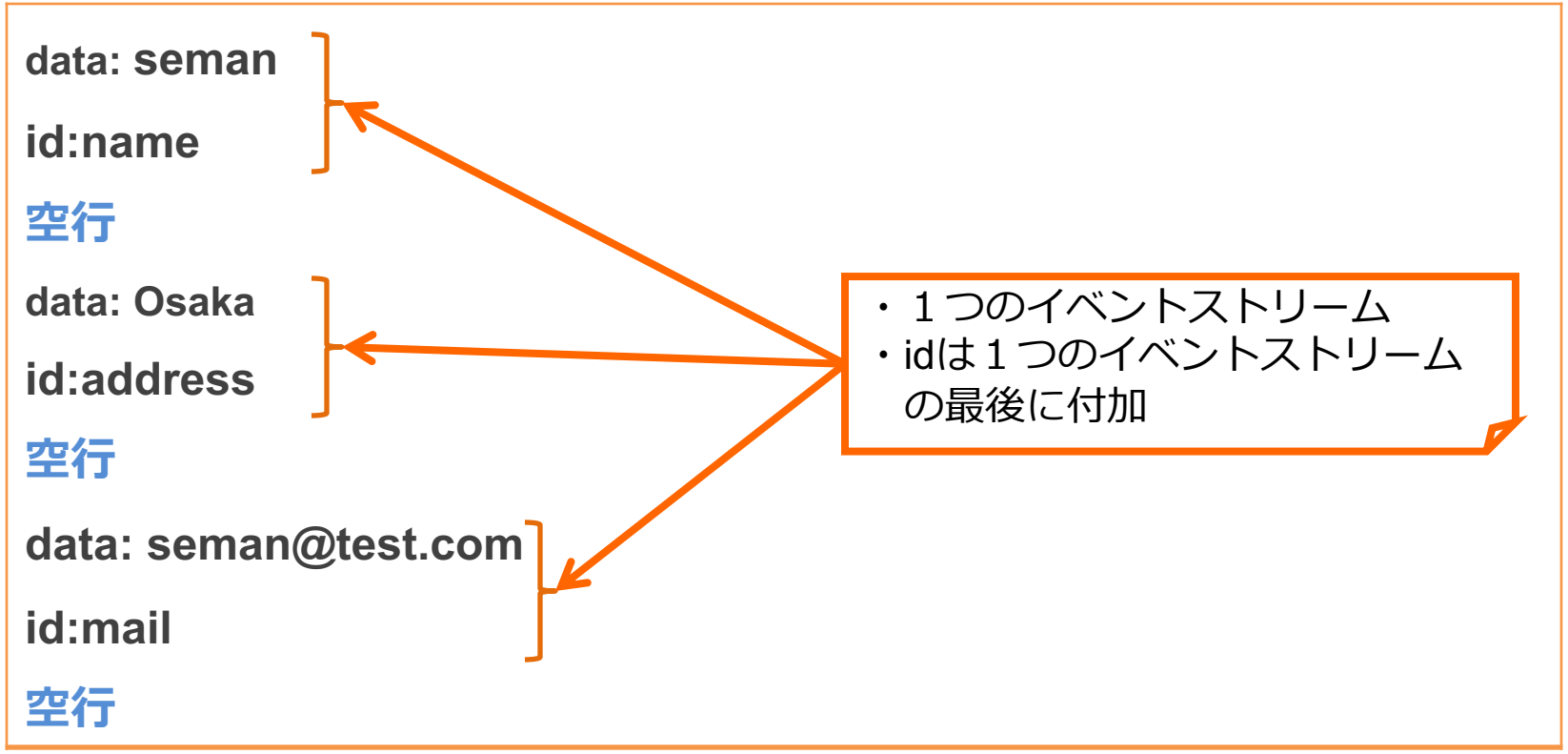
- イベントストリーム
  - サーバーアプリケーションがプッシュするメッセージ（文字列）
  - シンプルなテキストデータ
  - 文字符号化方式はUTF－８、MIMEタイプはtext/event-stream
  - イベントストリーム形式
  - 改行文字で区切られ、先頭のフィールドでテキストデータの意味を表現する

フィールド名　：　テキストデータ

フィールド名	意味
event	イベントのタイプを指定
data	メッセージデータ
id	イベントID（lastEventId）
retry	イベントストリーム再試行時間（ミリ秒）

ServerSentEvents

- イベントストリーム例



ServerSentEvents

- **ServerSentEventsの利用**
  - **ServerSentEventsを利用するためにはEventSourceオブジェクトの生成を行う**

```
var es = new EventSource(“http://www.testsv1.com/event.cgi”)
```

EventSource(url)

EventSourceオブジェクトのコンストラクタ

url ： サーバーアプリケーションのURL

- **openイベント**
  - **サーバーアプリケーションとの接続が完了した場合EventSourceオブジェクトにOpenイベントが発生**

```
es.onopen = function(e){
        /* 接続が完了した際に行う処理 */
}
```

ServerSentEvents

- messageイベント
  - イベントストリーム受信時、EventSourceオブジェクトにmessageイベントが発生
  - 受信データの参照

```
es.onmessage = function( e ){
        alert(e.data);
        (略)
}
```

MessageEventオブジェクト

ServerSentEvents

練習問題

ServerSentEventsを利用し、プッシュ型の通信を実現したい。

サーバーからクライアントに送信するHTTPResponseのヘッダー情報として正しい組み合わせはどれか。

A. Content-Type:text/event-stream
Transfer-Encoding:chunked

B. Content-Type:text/event-stream
Transfer-Encoding:compress

C. Content-Type:text/event-source
Transfer-Encoding:chunked

D. Content-Type:text/event-source
Transfer-Encoding:compress

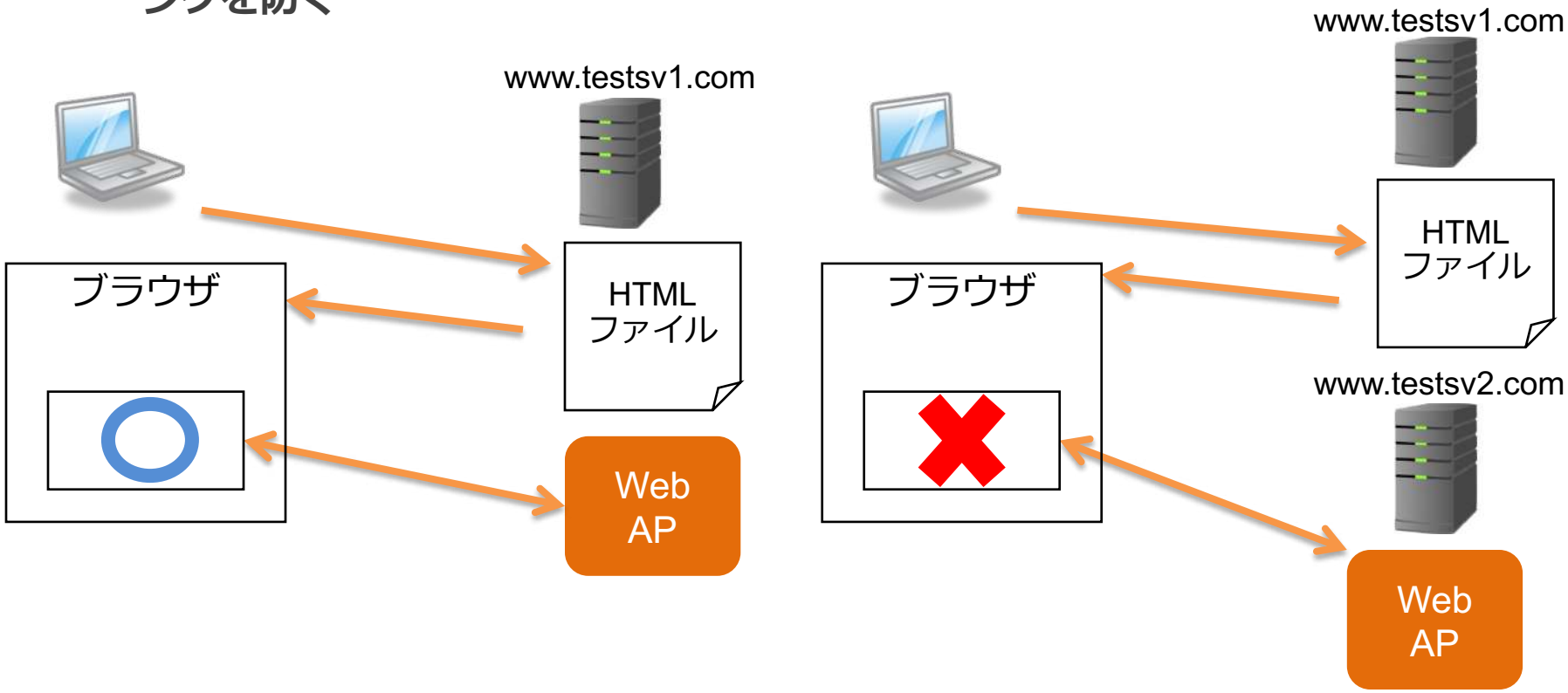
株式会社ケイ・シー・シー

技術解説項目

３.セキュリティモデル
・クロスオリジン制約とCORS

クロスオリジン制約とCORS

- クロスオリジン制約
  - セキュリティ上の理由で、そのHTMLファイルが保存されているオリジンとしかデータのやり取りが出来ない制約
  - 別のオリジンでドキュメント間の通信を禁止し、クロスサイトスクリプティングを防ぐ



クロスオリジン制約とCORS

- 代表的なクロスオリジン制約
  - XMLHttpRequest
  - Canvas
  - スクリプトによる別のoriginであるiframeやwindowに対する操作
- クロスオリジン制約の回避策
  - CORS
  - Messaging
  - JSONP

クロスオリジン制約とCORS

- CORS　【2014/1/16　勧告】
  https://www.w3.org/TR/cors/
  - サーバーがオリジンをまたぐアクセスを制御する方法を規定
  - オリジン間の安全な通信を保証

    - Access-Control-Allow-Credentials : true | false
      - クロスオリジン通信を行い認証情報、HTTPCookieを含むリクエストを許可（trueの場合）
      - *false*の場合、認証情報、HTTPCookieを含むクロスオリジン通信は禁止（デフォルト）

    - Access-Control-Allow-Origin : origin | *
      - リソースにアクセスしてよいオリジンを指定
      - ＊を指定した場合、すべてのオリジンにリソースへのアクセスを許可

LPI-JAPAN HTML5 Professional Certification

Open the Future with HTML5.