



HTML5の進化

～webアプリケーションプラットフォームとしてのHTML5～

高井 3 班 奥田 廉 小俣 晴 久保 優人 石川 怜亜琉



目次

- HTMLの変遷
- Webアプリケーションプラットフォームとは。
- タグの意味づけ
- マルチメディア対応
- ドラッグ&ドロップ対応
- GPSとの連携
- グラフィックス
- ストレージとファイル
- バックグラウンドとオフライン
- HTML5の通信関連技術

HTMLの変遷

年号	HTMLのVer	仕様
1993	HTML1.0	初のHTML。基本的なタグを定義。
1995	HTML2.0	HTML標準として公式化。フォームやテーブルなどの要素を追加。
1997年	HTML3.2	cssへの対応を開始。JavaScriptの利用が可能に。
1999年	HTML4.01	アクセシビリティ対応。CSSの完全対応。
2014年	HTML5	Webアプリケーションに対応する大幅拡張。多くのAPIの用意。

アプリケーション プログラミング インターフェース
※**API**(Application Programming Interface)・・・拡張されたオブジェクトのこと。

HTML5の変化点。

HTML5に変わり**Webアプリケーションプラットフォーム**が大きく整備された。

→HTML5で従来のデスクトップアプリケーションと遜色のないWebアプリケーションを開発できるようになった。



目次

- HTMLの変遷
- Webアプリケーションプラットフォームとは。
- タグの意味づけ
- マルチメディア対応
- ドラッグ&ドロップ対応
- GPSとの連携
- グラフィックス
- ストレージとファイル
- バックグラウンドとオフライン
- HTML5の通信関連技術

タグの意味付け・・・タグに付加的な情報を与える

- メリット

- タグに付加的な情報を加えることでコンピューターからその要素の意味を解釈しやすくなる。
- 検索エンジンなどに対して住所や電話番号の情報を与えやすくなる。

※注意：タグの意味づけの対応状況は、検索エンジンによって異なる。

<div>京都</div>

地名

<p>しおり</p>

人名



タグの意味づけの方法は三種類ある。

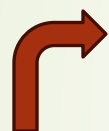
■ マイクロデータ

■ マイクロフォーマット

■ RDFa

マイクロデータ

→要素を**アイテム**として意味づけを行い、独自の属性が使われる。



・ **itemscope**属性

この要素(<div>~</div>)がアイテムであることを宣言する。

```
<div itemscope itemtype="http://data-vocabulary.org/Person">  
<p itemprop="name">しおり</p>  
</div>
```



・ **itemprop**属性

アイテムの属性の情報を指定する。



属性の値



・ **itemtype**属性

各属性が何を表すのかの取り決めが書いてる場所を指定する。

※**アイテム**・・・マイクロデータにおいて、ある特定のオブジェクトを表現する構造データの単位。

- ・離れたところのデータをアイテムに組み入れたいときは、**itemref属性**を使う。

```
<div itemscope itemref="rb">
```

```
  ⋮
```

```
</div>
```

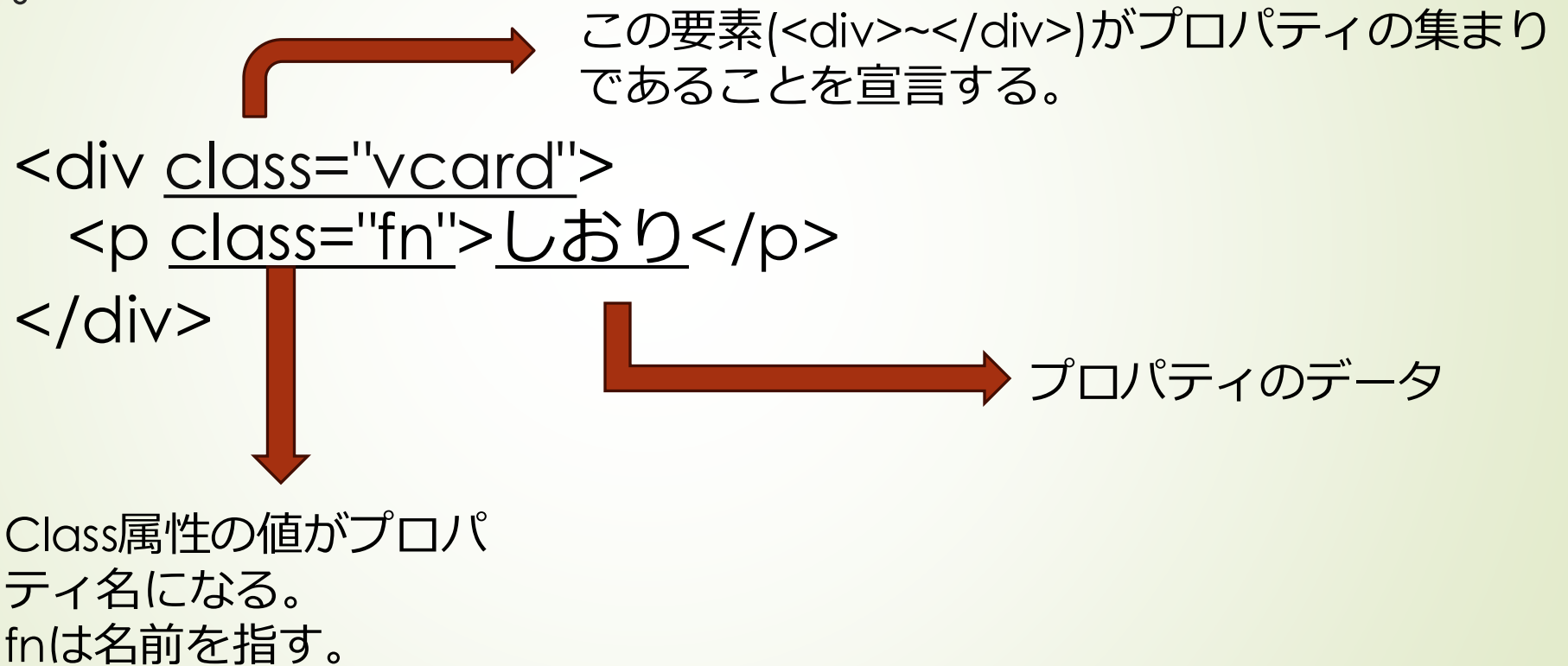
```
<p itemprop="rb">リボン</p>
```

対応付けさせる

→アイテムの内容を複数の要素に分割して記述することができる。

マイクロフォーマット

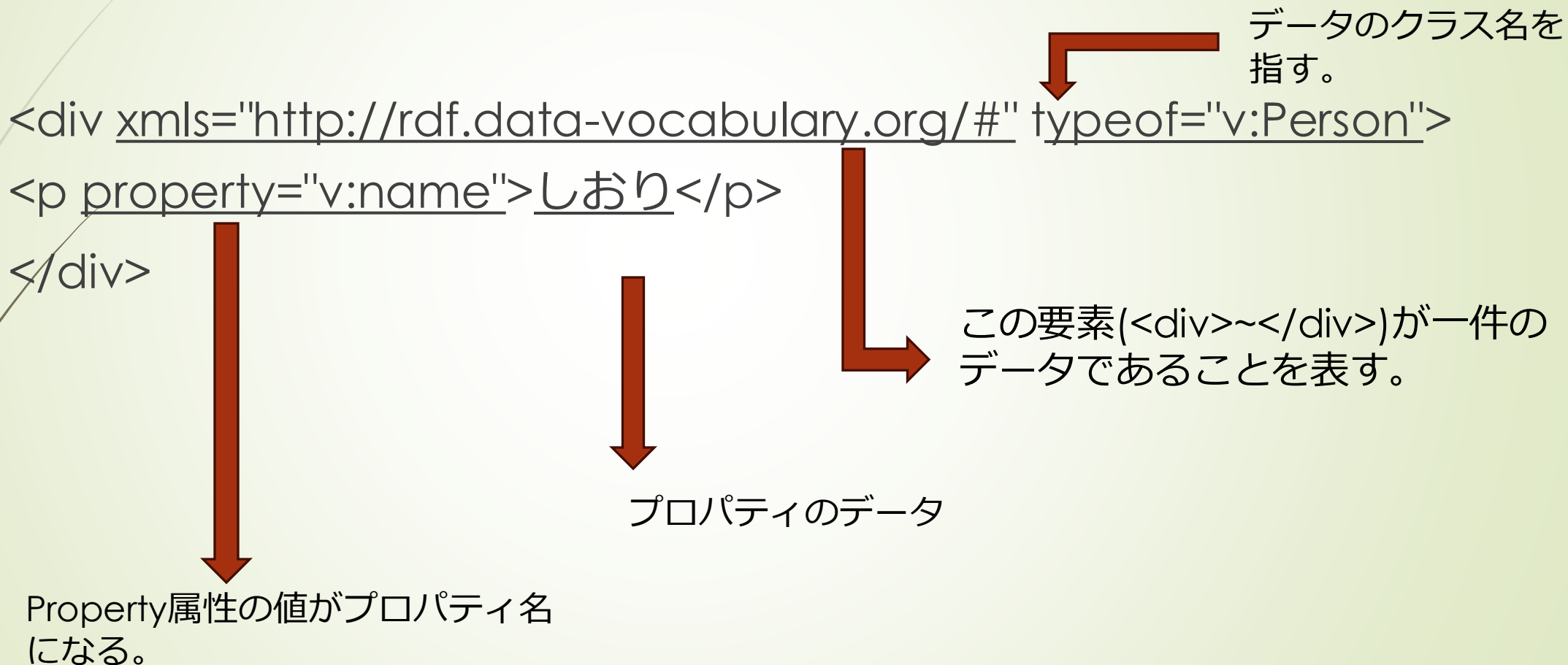
→既存の**class属性**を使って意味づけを行う。
指定する値は、マイクロフォーマットのサイトで定義されている。



※**class属性**・・・要素にクラス名を指定するグローバル属性。

RDFa

→XHTML向けのタグの意味づけ





目次

- HTMLの変遷
- Webアプリケーションプラットフォームとは。
- タグの意味づけ
- マルチメディア対応
- ドラッグ&ドロップ対応
- GPSとの連携
- グラフィックス
- ストレージとファイル
- バックグラウンドとオフライン
- HTML5の通信関連技術

マルチメディア対応

マルチメディアコンテンツとは・・・

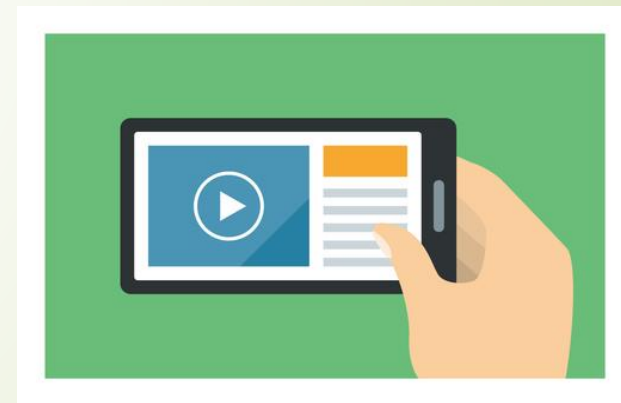
→音声や動画をウェブページで扱うための仕組みを指す。

音楽/音声

```
<audio src="mewing_cat.mp3"> </audio>
```

動画

```
<video src="running_cat.mp4"> </video>
```

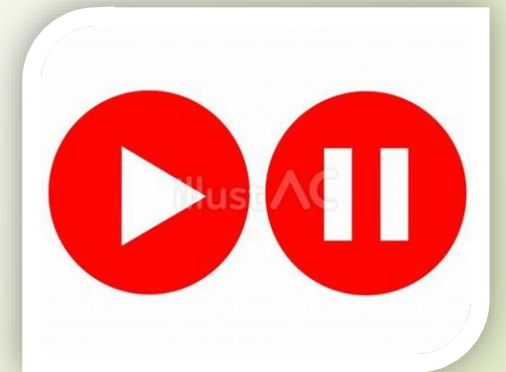


HTML5では、属性を指定することで音楽や動画の制御が可能。

<video src="running_cat.mp4" controls>再生できません</video>




属性を指定



属性には次のようなものがある。

プロパティ名	値
preload	コンテンツをあらかじめ読み込むか指定。 次の値をとる ・ none 読み込まない ・ metadata メタデータのみ読み込む ・ auto 自動(既定値)
autoplay	コンテンツを読み込み次第再生
loop	コンテンツをループ再生
controls	再生や停止などのボタンの追加
muted	消音にする
mediagroup	グループ名を指定
poster	[video要素] ビデオファイルを読み込めない場合に表示させる画像を指定
width	[video要素] ビデオの幅を指定
height	[video要素] ビデオの高さを指定



音楽や動画のファイルにはいろいろなコーデックがあり、HTML5では標準の**コーデック**を定めていない。

→様々なブラウザに対応するためsourceタグを使用して複数のコーデックを指定する。

※**コーデック**・・・音声や映像データを圧縮及び解凍するための技術やソフトウェア



```
<video control>
```

```
<source src=" running_cat.mp4">
```

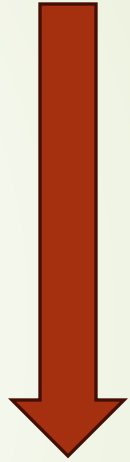
```
<source src=" running_cat.ogg">
```

```
<source src=" running_cat.webm">
```

```
<p>お使いのブラウザでは動画を再生できません。
```

```
<a href=" running_cat.mp4">ダウンロード</a></p>
```

```
</video>
```



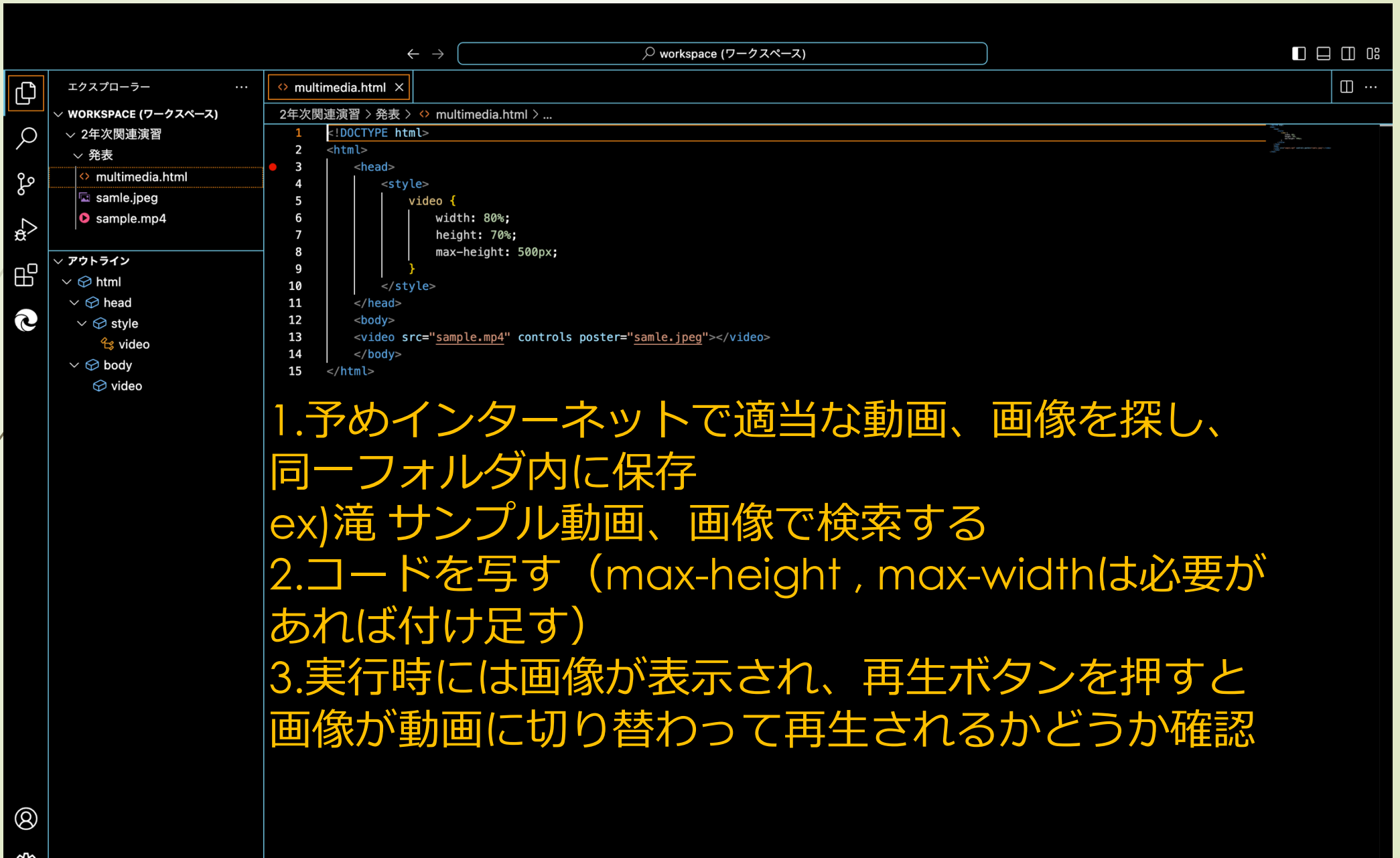
上から順に評価し、再生
できる形式が見つかった
ら、それを再生する



では、
実際にHTML5で動画を扱ってみよう！



実習1 ファイル名multimedia.html



The screenshot shows a web development IDE with a dark theme. The left sidebar contains a file explorer showing the workspace structure: WORKSPACE (ワークスペース) > 2年次関連演習 > 発表 > multimedia.html. Below this is an outline view showing the document structure: html > head > style > video > body > video. The main editor area displays the code for multimedia.html, which includes a DOCTYPE declaration, a head section with a style block for the video element, and a body section with a video element. The video element has a src attribute pointing to 'sample.mp4' and a poster attribute pointing to 'samle.jpeg'.

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <style>
5       video {
6         width: 80%;
7         height: 70%;
8         max-height: 500px;
9       }
10    </style>
11  </head>
12  <body>
13    <video src="sample.mp4" controls poster="samle.jpeg"></video>
14  </body>
15 </html>
```

1. 予めインターネットで適当な動画、画像を探し、同一フォルダ内に保存
ex) 滝 サンプル動画、画像で検索する
2. コードを写す (max-height, max-widthは必要があれば付け足す)
3. 実行時には画像が表示され、再生ボタンを押すと画像が動画に切り替わって再生されるかどうか確認

実習1 実行すると、、、

workspace (ワークスペース)

multimedia.html

2年次関連演習 > 発表 > multimedia.html > ...

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <style>
5       video {
6         width: 80%;
7         height: 70%;
8         max-height: 500px;
9       }
10    </style>
11  </head>
12  <body>
13    <video src="sample.mp4" controls poster="samle.jpeg"></video>
14  </body>
15 </html>
```

実行とデバッグ

実行とデバッグをカスタマイズするには、[launch.json ファイル](#)を作成します。

[すべての自動デバッグ構成を表示](#)

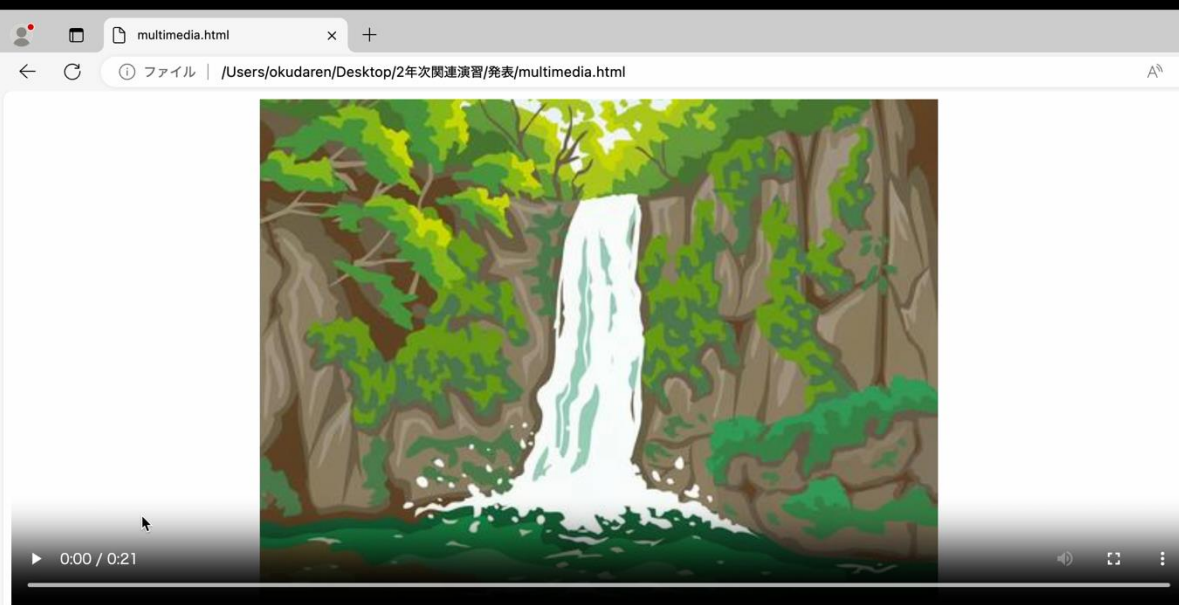
ブレークポイント

EVENT LISTENER BREAKPOINTS

- ☐ Ad Auction Worklet
- ☐ Animation
- ☐ Canvas
 - ☐ キャンバス コンテキスト...
 - ☐ WebGL エラーが発生
 - ☐ WebGL 警告が発生
- ☐ Clipboard
- ☐ Control
- ☐ Device
- ☐ DOM Mutation
- ☐ Drag / Drop
- ☐ Geolocation
- ☐ Keyboard
- ☐ Load
- ☐ Media
- ☐ Mouse
- ☐ Notification
- ☐ Parse
- ☐ Picture-in-Picture
- ☐ Pointer
- ☐ Script
- ☐ Timer
- ☐ Touch
- ☐ WebAudio
- ☐ Window

multimedia.html

ファイル | /Users/okudaren/Desktop/2年次関連演習/発表/multimedia.html





目次

- HTMLの変遷
- Webアプリケーションプラットフォームとは。
- タグの意味づけ
- マルチメディア対応
- ドラッグ&ドロップ対応
- GPSとの連携
- グラフィックス
- ストレージとファイル
- バックグラウンドとオフライン
- HTML5の通信関連技術



ドラッグ&ドロップ

HTML5ではWebページ内、またはブラウザ外からのドラッグ&ドロップができる。

要素どうしや、ブラウザと他のソフトの間で情報の受け渡しができる。


draggable属性

その要素がドラッグ可能かどうかを示す。

```
<div draggable=" ☐ ">ドラッグ&ドロップ</div>
```

true . . . ドラッグ可能

false . . . ドラッグ不可



ドラッグ&ドロップで利用するイベント

- dragstartイベント
- dragoverイベント
- dropイベント

dragstartイベント

ドロップ開始時に発生。

ここで渡したい値をセットする。



dragoverイベント

ドロップ直前に発生。



dropイベント

ドロップされたときに発生。

ここでデータを取り出す。



実習2 ファイル名drag&drop.html

workspace (ワークスペース)

drag&drop.html

2年次関連演習 > 発表 > drag&drop.html > ...

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <style>
5     #dragItem {
6       width: 100px;
7       height: 100px;
8       background-color: orange;
9       color: white;
10    }
11    #dropZone {
12      width: 250px;
13      height: 250px;
14      border: 2px solid black;
15      margin-top: 20px;
16    }
17  </style>
18 </head>
19 <body>
20   <div id="dragItem" draggable="true">ドラッグ&ドロップしてください</div>
21   <div id="dropZone"></div>
22
23   <script>
24     var ei = document.getElementById("dragItem");
25     var ez = document.getElementById("dropZone");
26
27     ei.ondragstart = function (e) {
28       e.dataTransfer.setData("text", ei.id);
29     }
30
31     ez.ondragover = function (e) {
32       e.preventDefault();
33     }
34
35     ez.ondrop = function (e) {
36       e.preventDefault();
37       var data = e.dataTransfer.getData("text");
38       var draggedElement = document.getElementById(data);
39       dropZone.appendChild(draggedElement);
40     }
41   </script>
42 </body>
43 </html>
```

- 一言一句写す

今打っているコードが何の役割を担っているのか意識してもらう

ドラッグするもの

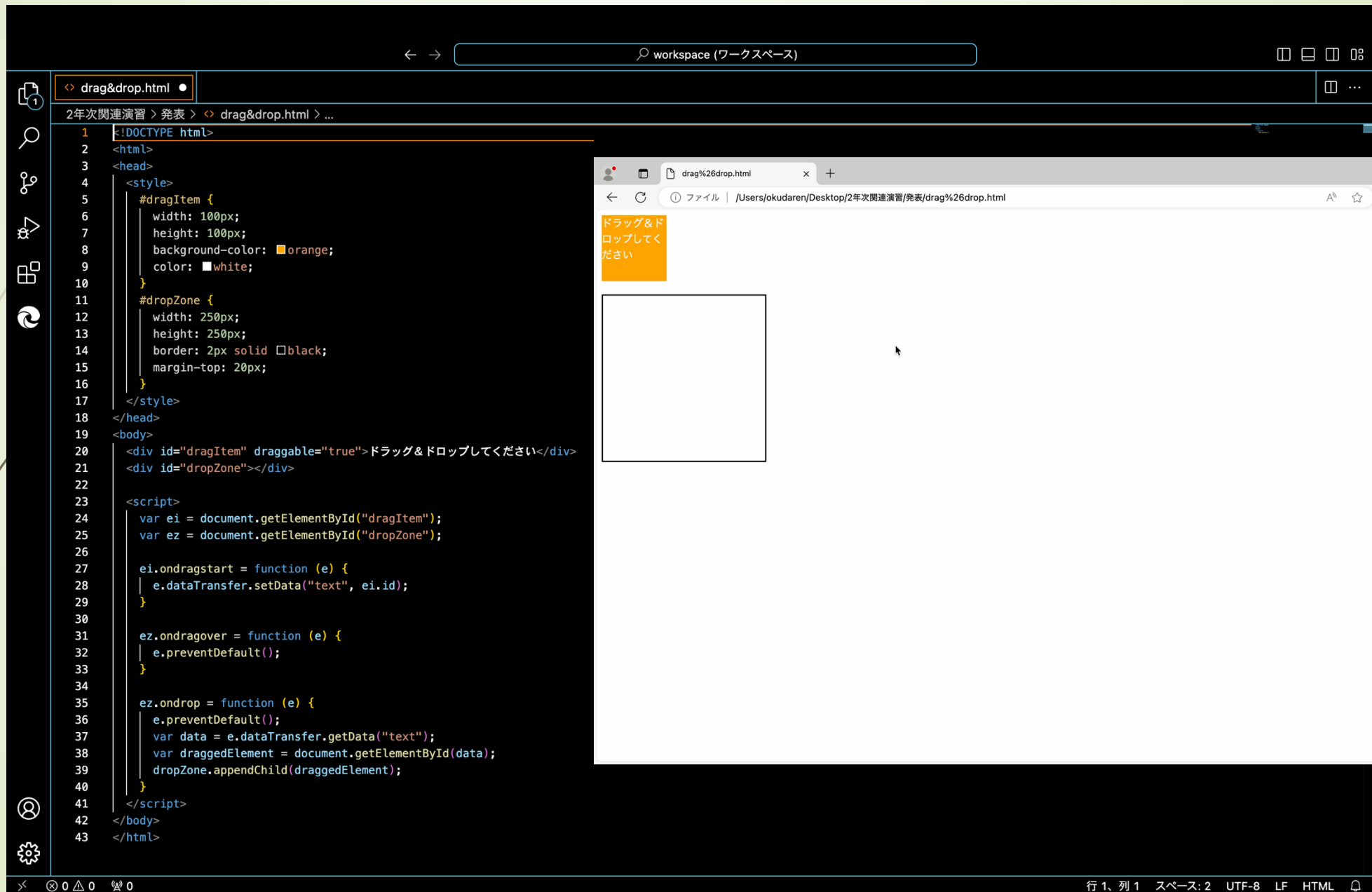
ドロップする場所

ドロップ開始時の処理

ドロップ直前の処理

ドロップ時の処理

実習2 実行すると、、、



The image shows a code editor on the left and a web browser on the right, demonstrating a drag-and-drop functionality.

Code Editor (drag&drop.html):

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <style>
5   #dragItem {
6     width: 100px;
7     height: 100px;
8     background-color: orange;
9     color: white;
10  }
11  #dropZone {
12    width: 250px;
13    height: 250px;
14    border: 2px solid black;
15    margin-top: 20px;
16  }
17 </style>
18 </head>
19 <body>
20 <div id="dragItem" draggable="true">ドラッグ&ドロップしてください</div>
21 <div id="dropZone"></div>
22
23 <script>
24   var ei = document.getElementById("dragItem");
25   var ez = document.getElementById("dropZone");
26
27   ei.ondragstart = function (e) {
28     e.dataTransfer.setData("text", ei.id);
29   }
30
31   ez.ondragover = function (e) {
32     e.preventDefault();
33   }
34
35   ez.ondrop = function (e) {
36     e.preventDefault();
37     var data = e.dataTransfer.getData("text");
38     var draggedElement = document.getElementById(data);
39     dropZone.appendChild(draggedElement);
40   }
41 </script>
42 </body>
43 </html>
```

Browser (drag%26drop.html):

The browser displays the rendered HTML. It shows a white box with the text "ドラッグ&ドロップしてください" (Drag & Drop please) and a larger white box with a black border below it. The text "ドラッグ&ドロップしてください" is highlighted in orange in the browser view.



目次

- HTMLの変遷
- Webアプリケーションプラットフォームとは。
- タグの意味づけ
- マルチメディア対応
- ドラッグ&ドロップ対応
- GPSとの連携
- グラフィックス
- ストレージとファイル
- バックグラウンドとオフライン
- HTML5の通信関連技術

Geolocation API

- ▶ HTML5ではGeolocation APIを使ってGPSを搭載した端末の位置を知ることができる。

```
navigator.geolocation.getCurrentPosition(successCallback, errorCallback);
```

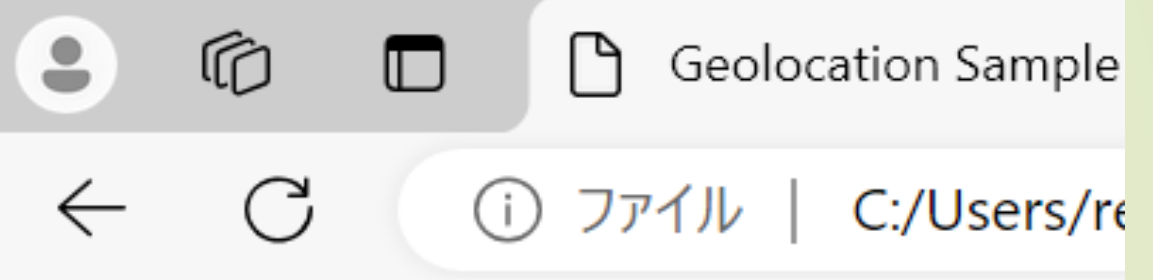
成功時に呼び出される関数

失敗時に呼び出される関数


```

<> gps.html > html
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Geolocation Sample</title>
6      <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5
7      <link rel="stylesheet" href="style.css">
8  </head>
9  <body>
10     <div class="center">
11         <div class="btn-margin">
12             <button id="btn" class="btn btn-outline-primary btn-lg">
13                 現在位置を取得する
14             </button>
15         </div>
16         <div class="txt-margin">
17             <p>緯度: <span id="latitude">???<span>度</span></p>
18             <p>経度: <span id="longitude">???<span>度</span></p>
19         </div>
20     </div>
21     <script src="geolocation.js"></script>
22 </body>
23 </html>

```



現在位置を取得する

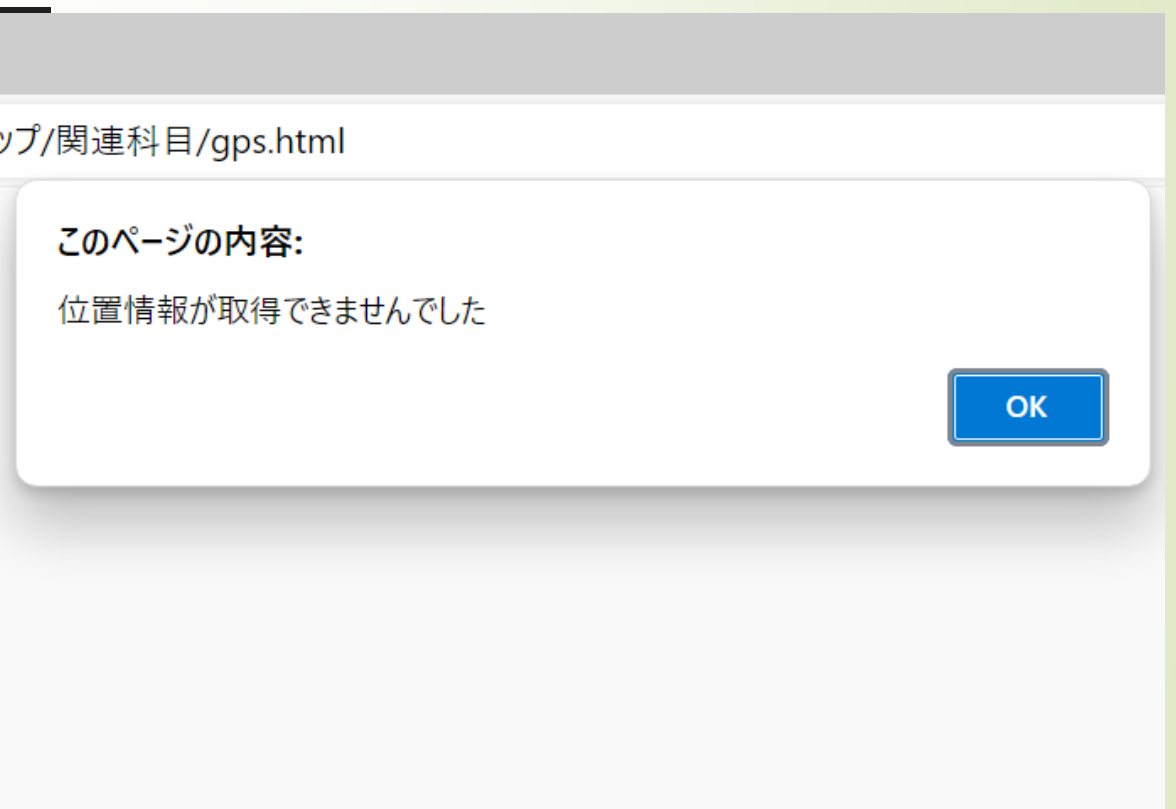
緯度: ???度

経度: ???度

成功時



失敗時





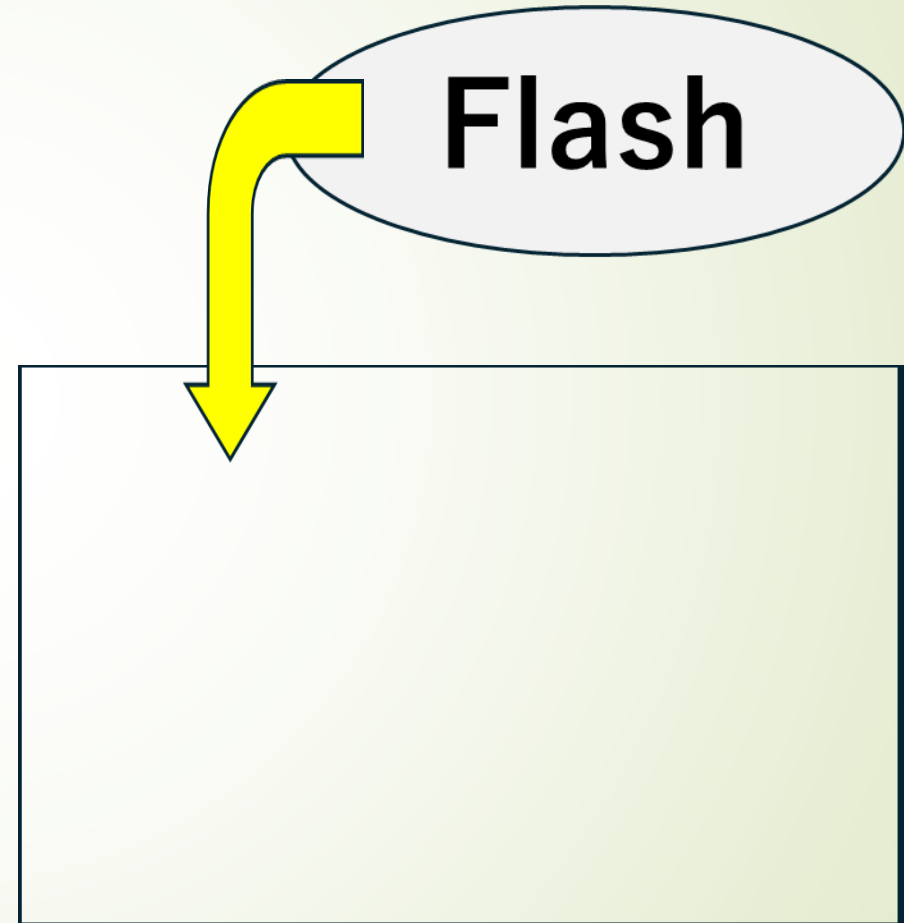
目次

- HTMLの変遷
- Webアプリケーションプラットフォームとは。
- タグの意味づけ
- マルチメディア対応
- ドラッグ&ドロップ対応
- GPSとの連携
- **グラフィックス**
- ストレージとファイル
- バックグラウンドとオフライン
- HTML5の通信関連技術

グラフィックス

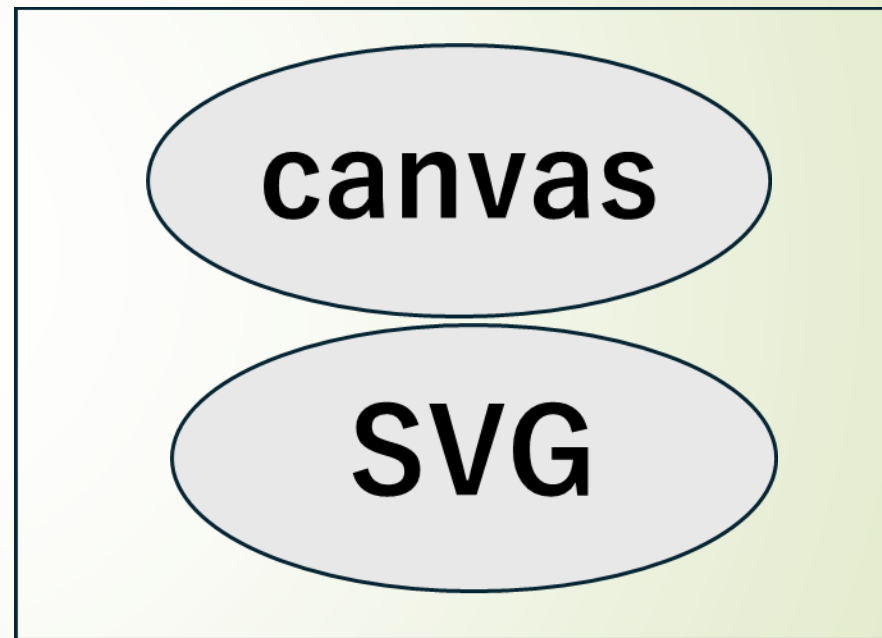
今までのHTMLではFlash
などのアドオンを使用しない
限り、サーバの処理を反映した
自由な図形を表示することは
できなかった。

アドオン(add-on)
ソフトやブラウザに機能を追加するた
めのプログラム



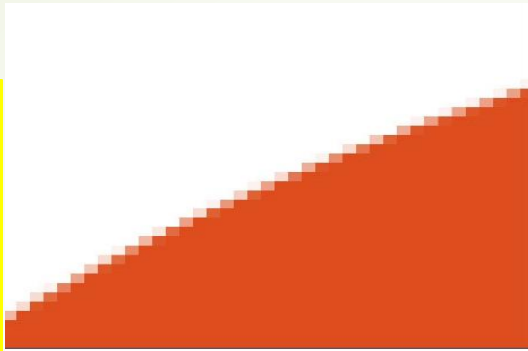
グラフィックス

HTML5で登場した
Canvasや**SVG**は、
アドオン無しに、
サーバの処理に応じて
図形を描画できる。



グラフや図形を手軽にWebページに表示できるようになった

Canvas



ビットマップ画像
(ピクセルと呼ばれる点の集合体)
で描画する。
描画方法はHTML側に絵を描く
下地となるcanvas要素を
用意して、Javascriptで描画する。

SVG



ベクタ画像
(線や面で構成された画像) で
描画する。
元々XMLやXHTMLで利用されて
いたものがHTML5で採用
された。HTML5ではsvg要素を
HTMLに直接埋め込むことが
できる。

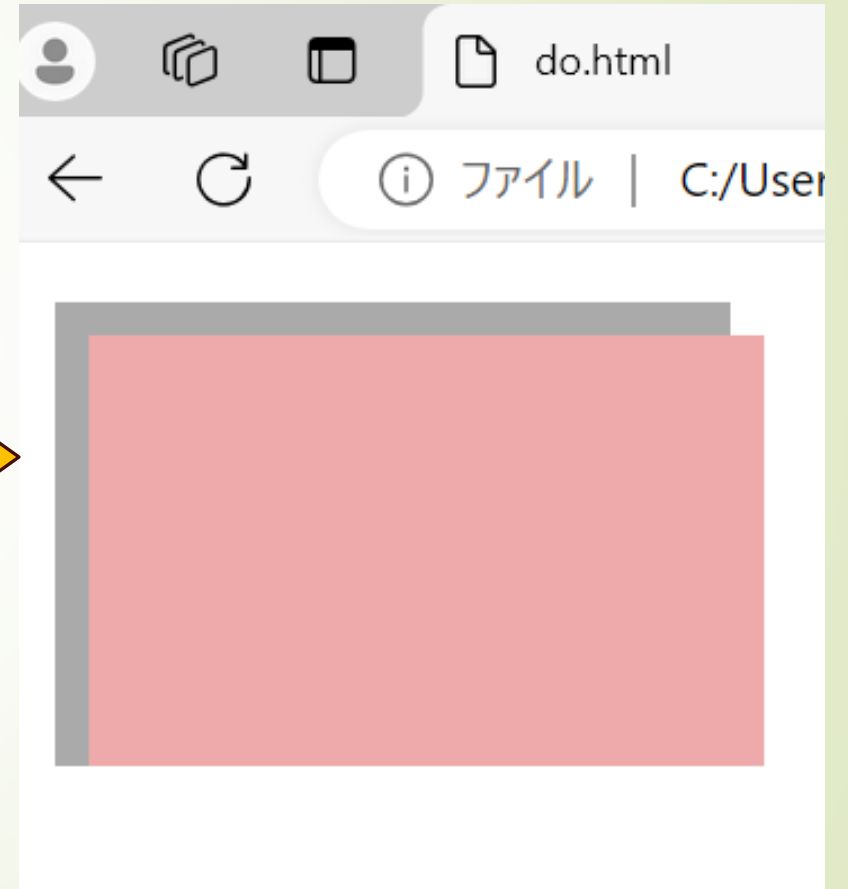
canvas

```
<canvas id="canbas1" width="300" height="300">
```

```
var canvas = document.getElementById("canvas1");  
var cvs = canvas.getContext("2d");  
cvs.fillStyle= "#aaaaaa";  
cvs.fillRect(10,200,220);  
cvs.fillStyle="#eeaaaa";  
cvs.fillRect(20,20,210,210);
```

SVG

```
<svg>  
  <rect x="10" y="10" width="200" height="200" fill="#aaaaaa" />  
  <rect x="20" y="20" width="200" height="200" fill="#eeaaaa" />  
</svg>
```



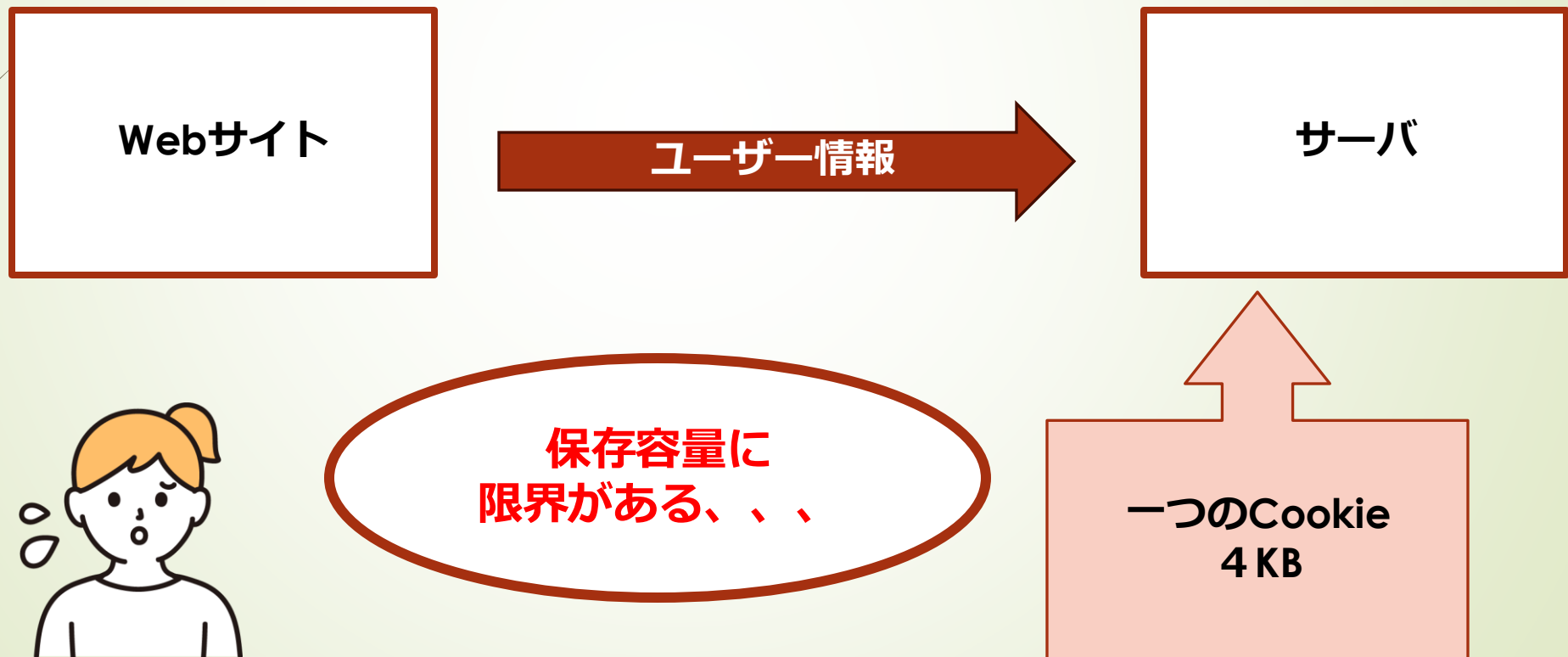


目次

- HTMLの変遷
- Webアプリケーションプラットフォームとは。
- タグの意味づけ
- マルチメディア対応
- ドラッグ&ドロップ対応
- GPSとの連携
- グラフィックス
- ストレージとファイル
- バックグラウンドとオフライン
- HTML5の通信関連技術

ストレージとファイル

➡ Cookieの制限



ストレージとファイル

HTML5でのストレージ

■ Web Storage

キーと値のペアでデータを保存する。

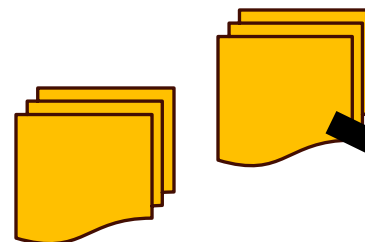
```
localStorage["sample"] = "Sample Data";  
var str = localStorage["sample"];
```

Cookieより簡単に
保存することができる！

■ Indexed Database API

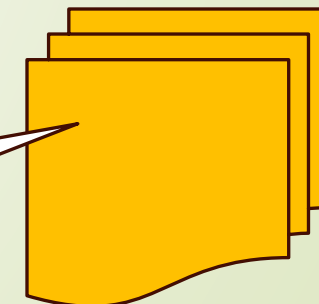
トランザクションに対応したデータベースを実現する。

データベース



いくつかの
オブジェクトストア
からできている

JavaScriptの
オブジェクトを
丸ごと保存できる



ストレージとファイル

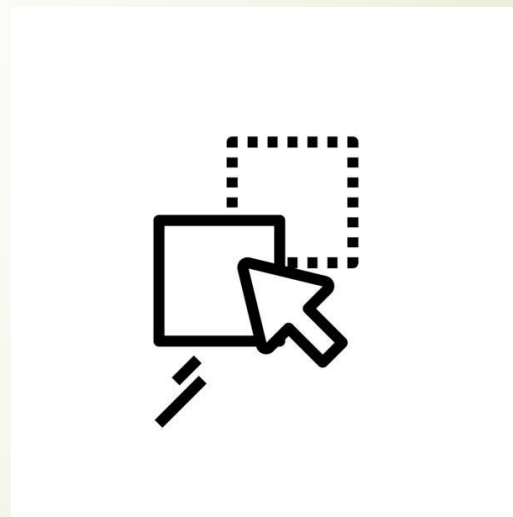
■ File APIとは、 、 、 ？

→クライアント環境のファイルにアクセスできるAPI

→ファイルの中身だけではなく、ファイル名の種類、ファイルサイズも取得できる



`<input type="file">`で
選択されたファイル



ドラッグ&ドロップ
されたファイル



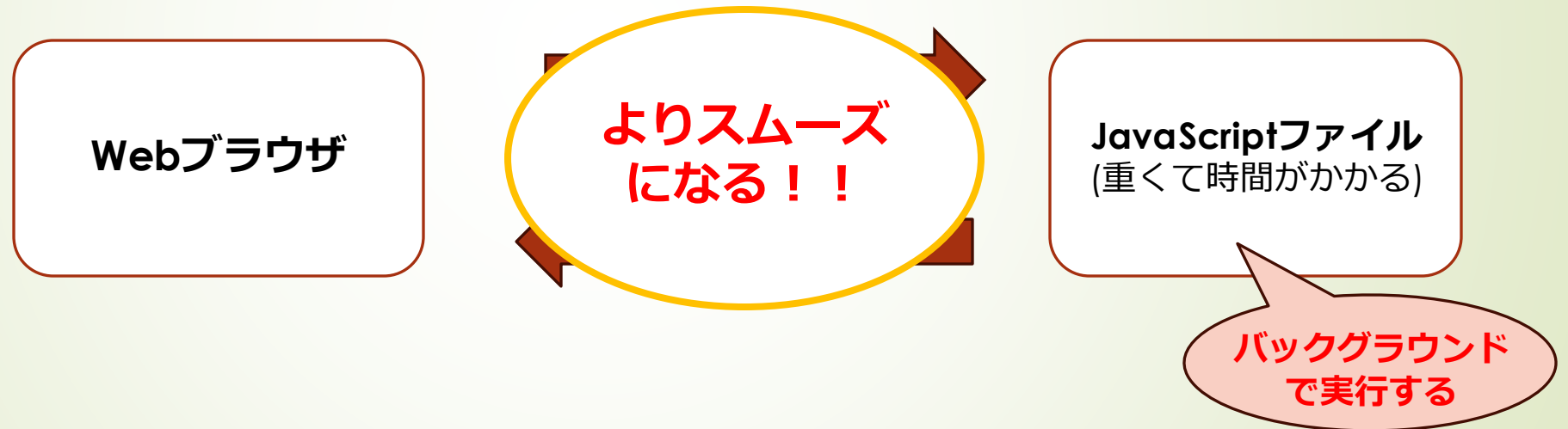
目次

- HTMLの変遷
- Webアプリケーションプラットフォームとは。
- タグの意味づけ
- マルチメディア対応
- ドラッグ&ドロップ対応
- GPSとの連携
- グラフィックス
- ストレージとファイル
- バックグラウンドとオフライン
- HTML5の通信関連技術

バックグラウンドとオフライン

■ Web Workers

→時間のかかる処理もできるようにするため、JavaScriptの処理をバックグラウンドで実行できるWeb Workersが追加された



→バックグラウンドで動作するJavaScriptのコード = **ワーカー**

バックグラウンドとオフライン

■ キャッシュマニフェスト

オフラインの状態でもWebアプリケーションを実行できるように、キャッシュを制御する**Application Cache**という仕組みがある。

→Application Cacheを利用するのに、**キャッシュマニフェスト**を作りキャッシュするものを指定することが必要

CACHE MNIFEST

```
sample.html  
sample.css  
sample.js
```

拡張子は.manifest

```
<!DOCTYPE html>  
<html manifest="sample.manifest">
```

html要素のmanifest属性を指定



目次

- HTMLの変遷
- Webアプリケーションプラットフォームとは。
- タグの意味づけ
- マルチメディア対応
- ドラッグ&ドロップ対応
- GPSとの連携
- グラフィックス
- ストレージとファイル
- バックグラウンドとオフライン
- HTML5の通信関連技術

HTML5の通信関連技術

■ Web Messaging

http://www.ank.co.jp:80 /a/b/c.html

オリジン

オリジンが異なる文書→データのやり取り、オブジェクトのアクセスに制限

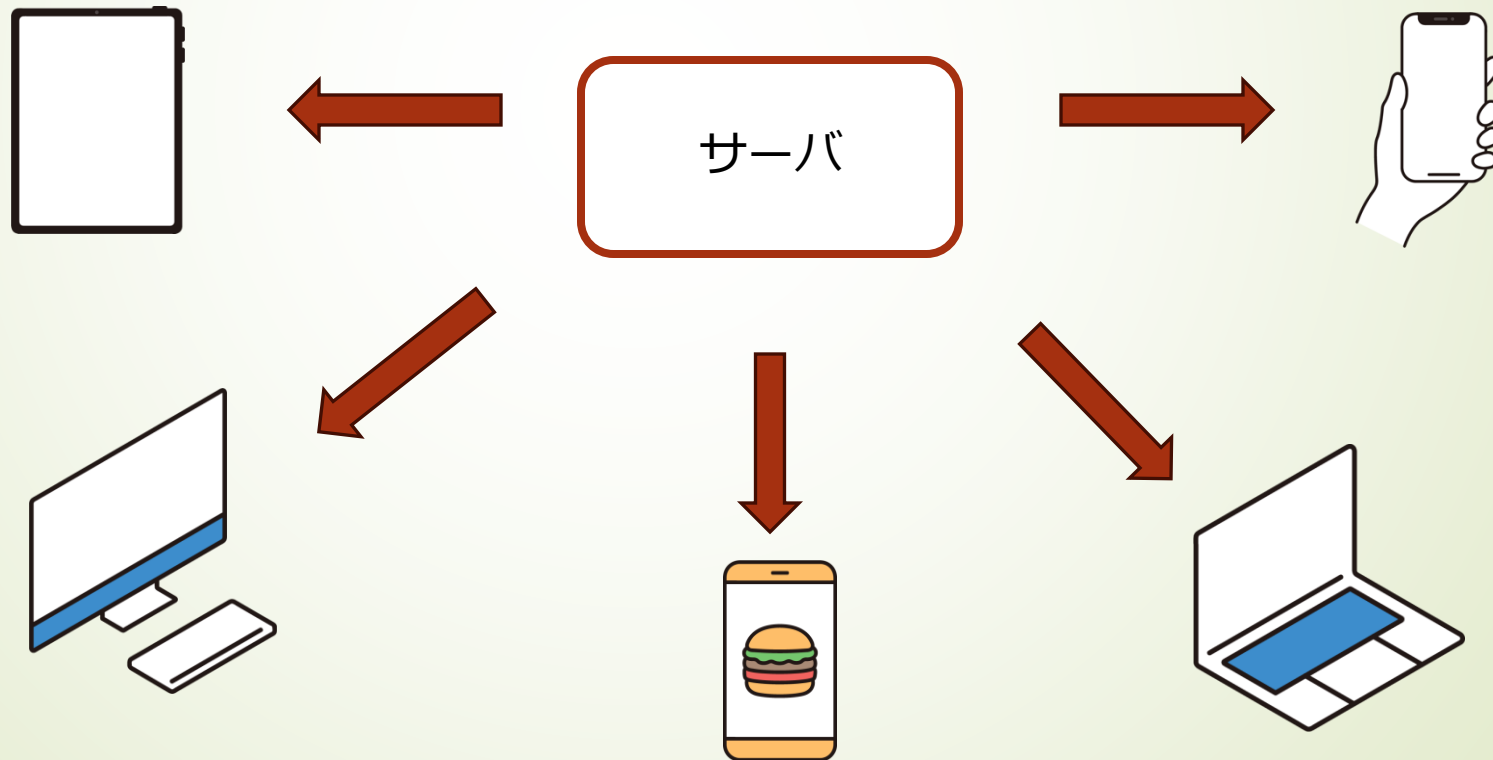
HTML5でWeb Messagingが導入

異なるオリジンのWebページ同士でも安全にデータのやりとりができるように！

HTML5の通信関連技術

■ Server-Sent Events

→ Webページのサーバからクライアントへの自動的な情報送信をするAPI

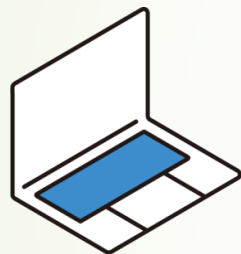


HTML5の通信関連技術

➡ Web Socket

→リアルタイムの双方向通信を手軽に実現するAPI

今までは、、、



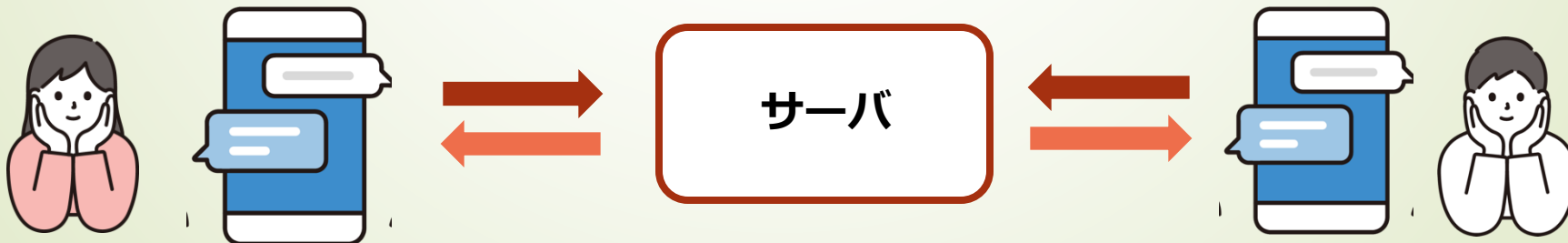
クライアント


1単位

リアルタイムで
双方向通信を行うのは
厳しい、

サーバ

→実用例：チャット





参考文献

- <https://codinghaku.com/html5-draganddrop/>
- [ベクター画像とビットマップ画像とは | Webデザイナー独学入門](#)
- [キャッシュ・マニフェスト << 基本 << HTML5入門](#)
- [Indexed Database APIの基礎 #JavaScript - Qiita](#)



ご清聴ありがとうございました。

