HTML5 & JavaScript

Webアプリ入門

河西 朝雄著

最新のHTML5技術をJavaScriptを使ってプログラミングすることで
斬新なWebアプリを簡単に作ることができます。

KASAI.SOFTWARELAB

定価1,500円（税込）

Web アプリ入門

（HTML5+JavaScript）

最新の HTML5 技術を JavaScript を使ってプログラミングすることで斬新な Web アプリを簡単に作ることができます。

河西　朝雄著

KASAI.SOFTWARELAB

定価 1,500 円（税込）

はじめに

HTML5 は WHATWG（Web Hypertext Application Technology Working Group）が中心になり開発を進めてきましたが、現在は W3C と WHATWG が共同で仕様策定をしています。しかし W3C と WHATWG で策定する HTML5 の仕様には違いがあり両者で共通する仕様は以下です。

・Semantic Elements
・Multimedia Elements
・HTML5 Forms
・Event model & APIs
・Offline Events
・Drag & Drop API
・HTML5 Parser

上に示した仕様以外に以下の仕様があります。WHATWG ではこれらの仕様を HTML5 の仕様に含めていますが、W3C では HTML5 関連技術という位置づけにあります。() 内は W3C の仕様の名称です。

・Canvas 2D Graphics Context(HTML Canvas 2D Context)
・Microdata(HTML5 Microdata)
・Cross-document messaging,Channel messaging(HTML5 Web Messaging)

これらの他に、W3C と WHATWG の仕様に定めれていませんが、現実のブラウザで採用されている SVG、Webstrage、Indexed Database、WebWorks、WebSocket などの HTML5 関連技術があります。

本書は厳密な意味での HTML5 にとらわれずに、HTML5 関連技術を含めた技術の中で、JavaScript との関連性の深い技術を以下の章立てで詳しく紹介します。

HTML/CSS の言語仕様の概要を Appendix1、JavaScript の言語仕様の概要を Appendix2 に示しますので参考にしてください。また 12 章、13 章、14 章ではサーバー側のプログラムを PHP で記述しています。PHP の言語仕様の概要を Appendix3 に示しますので参考にしてください。

5 章　Geolocation（ジオロケーション）
6 章　センサー
7 章　タッチイベント
8 章　ドラッグ＆ドロップ
9 章　File API
10 章　Web Storage
11 章　Indexed Database
12 章　XMLHttpRequest
13 章　WebSocket
14 章　Server-Sent Events（EventSource）
15 章　Web Messaging
16 章　アプリケーションキャッシュ
17 章　WebWorkers によるバックグラウンド処理
18 章　応用アプリ

Android や iPhone などのスマートフォンや iPad などのタブレット端末向けのアプリは、これらの端末のローカル記憶装置にインストールされて動作します。通常 Android 系アプリは Java で開発し、iPhone/iPad(iOS)系アプリは Objective-C で開発します。このような開発をした場合、Android 系アプリは、iPhone/iPad(iOS)では使用できず、逆に iPhone/iPad(iOS)系アプリは Android では使用できません。このように機種依存するアプリをネイティブアプリと呼びます。

これに対し、ブラウザ上で動作するアプリを Web アプリと呼びます。Web アプリは Android 系、iPhone/iPad(iOS)系といった機種依存はなく動作します。近年は JavaScript などにより動的なホームページ（Web ページ）が増えていますが、HTML5（および HTML5 関連技術）と組み合わせることにより、単なる Web ページではなく、そこで目的に応じて何らかの仕事が行えるようなアプリケーション（Web アプリ）が簡単に作れるようになります。

本書のプログラムはパソコンでもスマートフォンや iPad などのタブレット端末でも動作します。ただしマウスイベント関連はタブレット端末では動作しません。逆に Geolocation（ジオロケーション）、センサー、タッチイベントはパソコンでは動作しません。

また、パソコンでもタブレット端末でも適正な画面サイズで動作させるためには、<meta>タグに viewport を指定する必要があります。タブレット端末でのみ動作する 5 章～7 章のプログラムと 18 章の応用アプリにはこの<meta>タグを入れてあります。詳細は 18 章を参照してください。

2013 年 8 月　河西　朝雄

目次

1章　キャンバス

HTML5 ではグラフィックス描画を行うための領域を<canvas>タグで指定することができます。この領域に対し、直線、四角、円（円弧）、ベジェ曲線などの基本図形やテキスト、イメージなどを描画することができます。直線、円（円弧）、ベジェ曲線はパスに対する描画メソッドを使って一旦パスに対し描画を行い、その後 stroke メソッドあるいは fill メソッドを使ってパス情報をキャンバスに描画します。基本図形、テキスト、イメージに対し平行移動、回転、スケールなどの座標変換を施すことができます。また、グラデーションや影を付けたり、指定領域をクリップするこができます。

1-1　<canvas>タグ

HTML5 ではグラフィックス描画を行うための領域を<canvas>タグで指定することができます。width と height 属性にキャンバスの幅と高さをピクセル単位で指定します。

```
<canvas id="canvas" width="400" height="400"></canvas>
```

「注」ピクセル単位を明示するには"400px"とします。

```
<canvas id="canvas" width="400px" height="400px"></canvas>
```

1.　<canvas>領域への描画手順

<canvas>タグで指定した領域にグラフィックス描画を行うにはキャンバスオブジェクトを取得し、さらにそのキャンバスオブジェクトから実際にグラフィックス描画を行うためのコンテキストオブジェクトを取得します。getContext に指定できる引数は現在「2d:2 次元グラフィックス」だけです。このコンテキストオブジェクトに対し strokeRect メソッドや fillRect メソッドを使って図形を描画します。

```
var canvas = document.getElementById("canvas");
if(canvas.getContext){
    var context = canvas.getContext("2d");
    // context に対し描画メソッドを適用する
}
```

2.　矩形領域の描画

矩形領域を描画するメソッドとしては以下の 3 種類があります。引数はいずれも(x,y)位置を矩形領域の左上隅座標、w を矩形領域の幅、h を矩形領域の高さとします。

矩形領域描画メソッド	機能
strokeRect(x,y,w,h)	枠線のみの矩形領域を描画色で描く。
fillRect(x,y,w,h)	矩形領域の中を塗りつぶし色で塗る。
clearRect(x,y,w,h)	矩形領域を白色でクリアする。

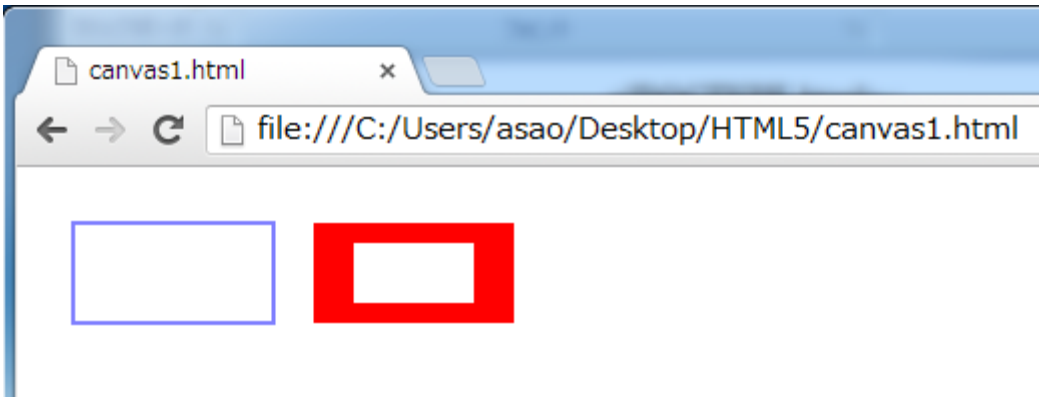
描画色と塗りつぶし色は strokeStyle プロパティと fillStyle プロパティで指定します。指定する色は"blue"のような色名、"#0000ff"のような 16 進数の RGB 値、"rgb(0,0,255)"のような rgb 関数が指定できます。

```
context.strokeStyle = "blue"; //  輪郭の色
context.fillStyle = "red";      //  塗る色
```

以下は青の輪郭だけの矩形領域、赤で塗った矩形領域、赤で塗った矩形領域の一部をクリアした矩形領域を描画するものです。

・canvas1.html

```
<!DOCTYPE html>
<html>
<body>
<canvas id="canvas" width="400" height="400"></canvas>
<script type="text/javascript">
	var canvas = document.getElementById("canvas");
	if(canvas.getContext){
		var context = canvas.getContext("2d");
		context.strokeStyle = "blue"; //  輪郭の色
		context.fillStyle = "red";	//  塗る色
		context.strokeRect(20, 20, 100, 50);
		context.fillRect(140, 20, 100, 50);
		context.clearRect(160, 30, 60, 30);
	}
</script>
</body>
</html>
```



2章　SVG

SVGは「Scalable Vector Graphics」の略で、図形を点の座標とそれを結ぶ線や面の方程式のパラメータで表したベクターグラフィックスです。これに対し、図形を点(ドット)の羅列として表現する方式をビットマップグラフィックス(ラスタグラフィックス)といいます。ベクターグラフィックスは図形を拡大・縮小したり変形したりしても、輪郭の処理などがその都度行われ、解像度に見合った画質が維持されるという特徴があります。

HTML5でのSVGは<svg>タグでベクターグラフィックス領域を定義し、<line>、<rect>、<text>などの基本図形タグを使って描画を行います。各図形を座標変換したり、各図形を描くパスを設定することができます。また、グラデーションやフィルターをかけることができます。

SVGでは<animate>タグ、<animateTransform>タグ、<animateMotion>タグを用いて簡単にアニメーションを行うことができます。

2-1　基本図形

SVG で描画できる基本図形は以下の 9 種類でそれぞれのタグと属性を用いて記述します。

基本図形のタグ	指定する属性
<line>	x1,y1：始点　x2,y2：終点。
<rect>	x,y：左上隅座標　width,height：幅、高さ　rx,ry：4 隅の円弧の半径。
<circle>	cx,cy：中心座標　r：半径。
<ellipse>	cx,cy：中心座標　rx,ry：x 軸方向半径、y 軸方向半径。
<polyline>	points：各点の座標。
<polygon>	points：各点の座標。
<path>	d：パスコマンド。
<image>	x,y：イメージの左上隅座標　width,height：幅、高さ xlink:href：イメージファイルの URL。
<text>	x,y：テキストの左下隅座標　font-size：フォントサイズ。

図形を描画する際に各図形タグに共通の属性（描画色、塗る色、線幅など）として以下があります。

各図形タグに共通の属性	機能
stroke	描画色。デフォルトは黒。
stroke-width	描画幅。デフォルトは 1。
fill	塗る色。デフォルトは黒。none を指定すると塗らない。
opacity	透明度。0.0～1.0。

SVG で使用できる座標の単位として以下があります(CSS と同じ単位です)。単位を省略した場合「px」とみなされます。本書では単位を省略してありますが、ピクセル単位を明示するには「px」を指定してください。

単位	意味
px	ピクセル。
pt	ポイント（1.25px）。
pc	パイカ（15px）。
mm	ミリメートル（3.543307px）。
cm	センチメートル（35.43307px）。

in	インチ（90px）。
em	現在のフォントの大きさ。
ex	現在のフォントでの文字 x の高さ。
%	ビューポートに対する割合。

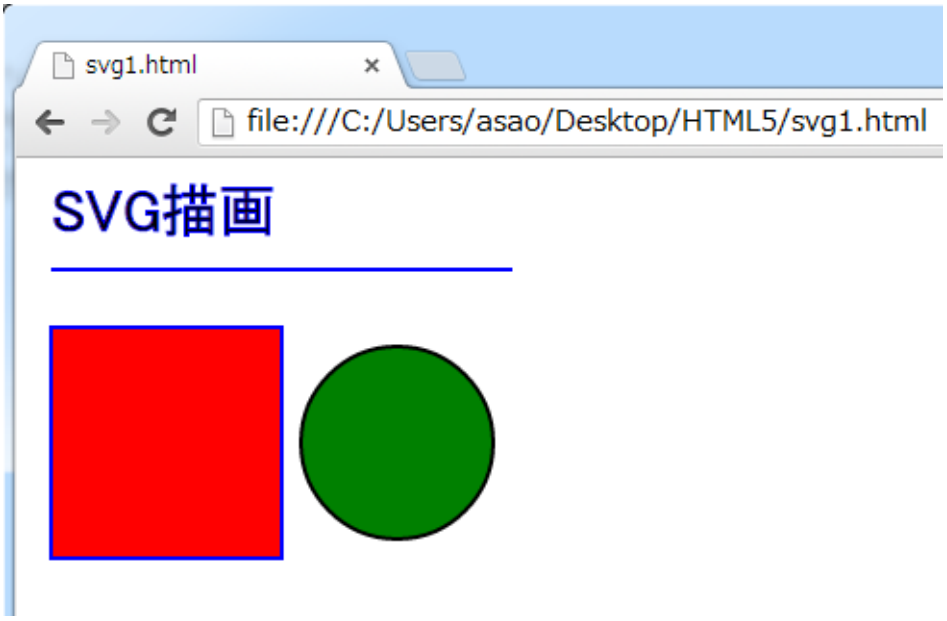
以下はテキスト、直線、四角、円を描くものです。

・svg1.html

```
<!DOCTYPE html>
<html>
<body >
<svg width="400" height="400">

<text x="10" y="30" stroke="blue" font-size="30">SVG 描画</text>
<line x1="10" y1="50" x2="250" y2="50" stroke="blue" stroke-width="2" />
<rect x="10" y="80" width="120" height="120" fill="red" stroke="blue" stroke-width="2"
/>
<circle cx="190" cy="140" r="50" fill="green" stroke="black" stroke-width="2" />

</svg>
</body>
</html>
```



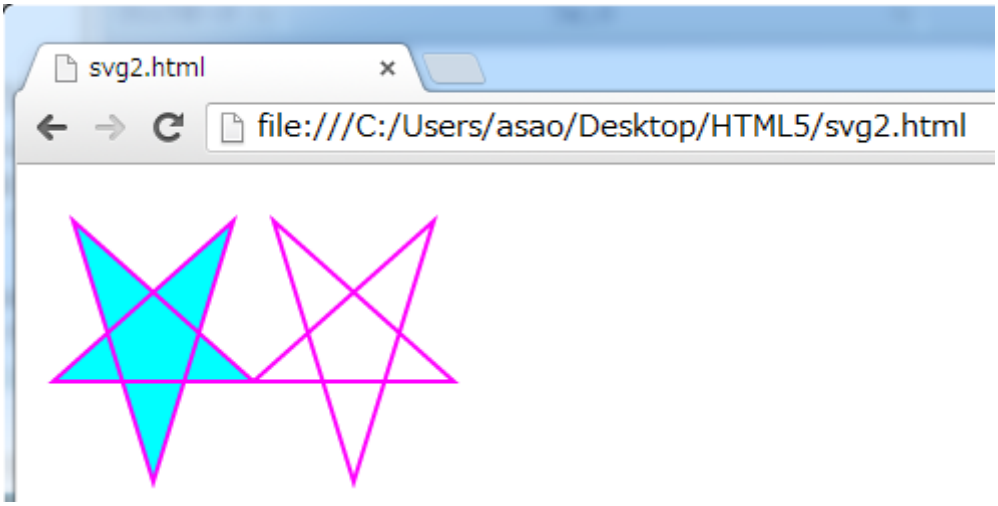
以下は星型のポリゴンを polygon と polyline を使って描くものです。polygon は始点を終点とみなします。polyline は最後の終点の座標を指定しないと図形は閉じません。

・svg2.html

```
<!DOCTYPE html>
<html>
<body >
<svg width="400" height="400">

<polygon points="10 100 100 20 60 150 20 20 110 100" fill="cyan" stroke="magenta" stroke-width="2" />
<polyline points="110 100 200 20 160 150 120 20 210 100 110 100" fill="none" stroke="magenta" stroke-width="2" />

</svg>
</body>
</html>
```



3章　ビデオとオーディオ

ビデオ（動画ファイル）は<video>タグを使って再生することができます。
オーディオ（音楽ファイル）は<audio>タグを使って再生することができます。
<video>タグも<audio>タグも src 属性に再生するファイルの URL,type 属性にファイルの MIME タイプを指定します。

再生できるビデオフォーマットとオーディオフォーマットはブラウザごとに異なります。

再生中に一定間隔でアップデートするリスナーは「timeupdate」を指定します。この中で currentTime プロパティを使って現在の再生時間を取得することができます。

3-1　<video>タグによる動画再生

<video>タグを用いて動画ファイルを再生することができます。

1.　<video>タグ

src 属性に「動画ファイルの URL」、type 属性に「ビデオフォーマット」を指定します。controls 属性を指定することでビデオプレーヤーのコントロール（再生、一時停止、シーク、音量設定など）が表示されます。

<video controls src="動画ファイルの URL" type="video/ビデオフォーマット"></video>

<video>タグに指定できる属性として以下があります。

<video>タグの属性	機能
src	動画ファイルの URL。
type	動画ファイルのビデオフォーマット。
controls	ビデオプレーヤーのコントロール（再生、一時停止、シーク、音量設定など)。
loop	繰り返し再生。
autoplay	ビデオの再生の自動開始。
preload	事前読み込み。auto、metadata、none を指定。
muted	消音（ミュート)。
poster	ビデオプレーヤーに表示されるプレースホルダー画像。
width	ビデオプレーヤーの幅(ピクセル単位)。
height	ビデオプレーヤーの高さ(ピクセル単位)。

「注」preload 属性

preload 属性を指定（preload または preload="auto"）すると、ウェブページを読み込んだ時点で動画をバックグラウンドで読み込みます。

preload="metadata"を指定すると、動画全体ではなく、動画のサイズ・最初のフレーム・トラックリスト・再生時間などの動画のメタデータのみを取得します。

preload="none"（デフォルト）を指定すると事前読み込みはしません。ユーザーが動画をあまり必要としていないことが想定される場合や、ウェブサーバに余分な負担を掛けたくない場合に指定すると良いでしょう。

2. <video>タグで再生できるビデオフォーマット

<video>タグで再生できるビデオフォーマットはブラウザにより異なります。主なビデオフォーマットとして以下があります。

ビデオフォーマット	意味	MIME タイプ
WebM	動画コーデック「VP8」をベースとして、Google が開発した著作権フリーの動画フォーマット。	video/webm
MP4	QuickTime 動画フォーマットをベースに ISO/IEC の MPEG-4 で規定されている動画フォーマット。	video/mp4
3GP	MPEG-4 をベースに 3GPP (Third Generation Partnership Project)が規定する第 3 世代携帯電話向け動画フォーマット。	video/3gpp

上の表の 3 種類のビデオフォーマットに対する各ブラウザごとのサポート状況を示します。ブラウザのバージョンによりサポート状況が異なる場合もあります。

	webm	mp4	3gp
Mobile Safari	○	○	○
Mobile Chrome	○	○	○
Internet Exploler	△	○	×

・video1.html

```
<!DOCTYPE html>
<html>
<body>

<h3>WebM</h3>
<video src="movie1.webm" type="video/webm" width="400" height="300" controls>
</video>

<h3>MP4</h3>
<video src="movie2.mp4" type="video/mp4" width="400" height="300" controls>
</video>

<h3>3GP</h3>
```

```
<video src="movie3.3gp" type="video/3gpp" width="400" height="300" controls>
</video>

</body>
</html>
```

・Mobile Chrome(Android)

4 章　Google Maps

Google Maps API の「http://maps.google.com/maps/api/js」を読み込むことで簡単に Google Maps を表示し、制御することができます。通常のロードマップや航空写真を表示することができます。地図の表示位置を移動したり、ズームを変えたりなどの操作ができます。マップに独自のイベントリスナーを付けることができます。マーカーや吹き出しを付けることもでます。地図の上にサークル、ポリライン、ポリゴンなどの図形を重ね合わせることもできます。

4-1　最もシンプルな地図の表示

Google Maps を表示する最もシンプルな手順を以下に示します。

1.　Google Maps API の読み込み

```
<script type="text/javascript" src="http://maps.google.com/maps/api/js?sensor=false">
</script>
```

「http://maps.google.com/maps/api/js」は、Google Maps API を使うのに必要な全てのシンボルを読み込む JavaScript ファイルを示しています。

sensor パラメータは、ユーザーの位置を決めるのに(GPS のような)センサーを使うかどうかを示す true/false のどちらかを指定します。

2.　Google Maps を表示する領域の設定

タグなどで、Google Maps を表示する領域を設定します。画面一杯に地図を表示したい場合は「100%」を指定します。

```
<div id="map" style="width:100%;height:100%;"></div>
```

ただし、Google Maps に関しては、 の親要素の高さを以下のように明示しておかないと%指定の場合の高さは 0px となってしまいます。

```
<style type="text/css">
    html,body {
        height: 100%;
    }
</style>
```

JavaScript コードで画面のサイズに指定するには以下のようにします。ただし画面のサイズは初期状態のサイズですので、画面がリサイズされても地図のサイズはリサイズされません。

```
var obj=document.getElementById("map");
obj.style.width=document.documentElement.clientWidth+"px";
obj.style.height=document.documentElement.clientHeight+"px";
```

「注」Google Maps 利用時には画面サイズを document.body.clientWidth 、document.body.clientHeight では取得できません。

3. マップオブジェクトの生成

マップの表示領域 obj とマップの各種設定値を指定した options を引数にして「google.maps.Map(obj,options);」でマップオブジェクトを生成します。

```
var latlng = new google.maps.LatLng(35.681099,139.767084); // 東京駅
var options = {
    zoom: 16,              // ズーム
    center: latlng,        // 中心
    mapTypeId: google.maps.MapTypeId.ROADMAP // 地図の種類
};
map = new google.maps.Map(obj,options);
```

options で指定するマップの各種設定値は zoom にズーム値(1:最縮小~21:最拡大)、center にマップの中心位置(緯度、経度を google.maps.LatLng コンストラクタで生成したもの)、mapTypeId にマップの種類（下表）を指定します。

マップの種類	意味
ROADMAP	通常のロードマップ。
SATELLITE	航空写真。
HYBRID	航空写真の上に有名な地物(道路や街の名前など）を重ねて表示。
TERRAIN	山や川などの地形的特徴を持つ地図。

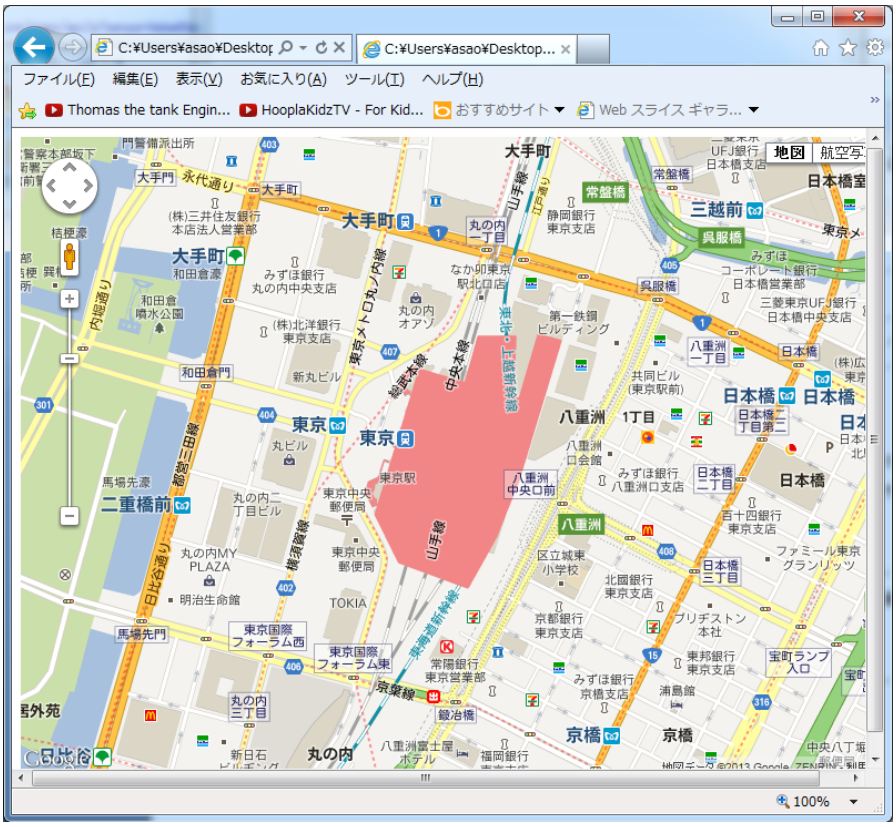
以下は東京駅を中心にズーム 16 で表示したものです。

・map1.html

```
<!DOCTYPE html>
<html>
<head>
<style type="text/css">
    html,body {
        height: 100%;
    }
</style>
<script type="text/javascript" src="http://maps.google.com/maps/api/js?sensor=false">
</script>
<script type="text/javascript">
    var map;
    function dispMap(){
```

```
        // マップ表示領域の取得
        var obj=document.getElementById("map");

        // マップ作成
        var latlng = new google.maps.LatLng(35.681099,139.767084); // 東京駅
        var options = {
            zoom: 16,               // ズーム
            center: latlng,       // 中心
            mapTypeId: google.maps.MapTypeId.ROADMAP // 地図の種類
        };
        map = new google.maps.Map(obj,options);
    }
</script>
</head>
<body onLoad="dispMap()">
<div id="map" style="width:100%;height:100%;"></div>
</body>
</html>
```



5 章　Geolocation（ジオロケーション）

Geolocation API は、W3C が仕様策定を進める HTML5 関連規格であり、JavaScript で位置情報を取得できるように標準化されています。ケータイやスマホのような GPS 対応の携帯端末向けのウェブサイトだけではなく、一般的なブラウザで閲覧するいわゆる PC サイトでもユーザーの位置情報を利用したコンテンツを提供することが可能になります。

W3C(World Wide Web Consortium)は World Wide Web で使用される各種技術の標準化を推進する為に設立された標準化団体です。

パソコンでもタブレット端末でも適正な画面サイズで動作させるためには、<meta>タグに viewport を指定する必要があります。タブレット端末でのみ動作する 5 章～7 章のプログラムと 18 章の応用アプリにはこの<meta>タグを入れてあります。詳細は 18 章を参照してください。

5-1　navigator.geolocation オブジェクト

ケータイやスマホなどは GPS を搭載しているため、高精度な緯度・経度を取得することができます。しかし、デスクトップパソコンには GPS が搭載されていないので通常は位置情報を取得できませんが、ブラウザによっては、Wi-Fi 位置データと IP アドレス情報に基づいて、ブラウザが実行されているデスクトップパソコンの経度と緯度を特定することができるものもあります。

geolocation API は navigator オブジェクトの子オブジェクトである geolocation オブジェクトを通じて提供されます。geolocation API が使用できるかどうかは以下のコードで調べることができます。

```
if (navigator.geolocation) {
    alert("Geolocation API を使用できます");
}
else {
    alert("Geolocation API を使用できません");
}
```

位置情報を取得するには getCurrentPosition メソッドを実行し、引数で指定したコールバック関数内で位置情報を取得します。

```
navigator.geolocation.getCurrentPosition(success);
function success(pos) { //  コールバック関数
    // pos.coords.latitude と pos.coords.longitude で緯度・経度を取得
}
```

コールバック関数の引数 pos から取得できるプロパティは以下です。pos.coords.latitude と pos.coords.longitude で緯度・経度を取得することができます。

引数 pos のプロパティ	機能
coords.latitude	緯度（-90 度～90 度）。
coords.longitude	経度（-180 度～180 度）。
coords.altitude	高度（m）。
coords.accuracy	緯度・経度の誤差（m）。
coords.altitudeAccuracy	高度の誤差（m）。
coords.heading	方角（0 度～360 度）。
coords.speed	速度（m/秒）。

timestamp	位置情報が取得できたときのタイムスタンプ（1970 年からの経過ミリ秒)。new Date(pos.timestamp)で Date オブジェクトを生成。

「注」コールバック関数

ある関数の呼び出し時の実引数に関数へのポインタを指定することで、呼び出し元の関数では指定された関数を内部的に呼び出して使うことができます。このように呼び出し元の関数の手助けをする関数をコールバック関数と呼びます。

以下は現在位置に関する緯度、経度、高度、緯度・経度の誤差、高度の誤差、方角、速度、タイムスタンプなどの情報を表示するものです。

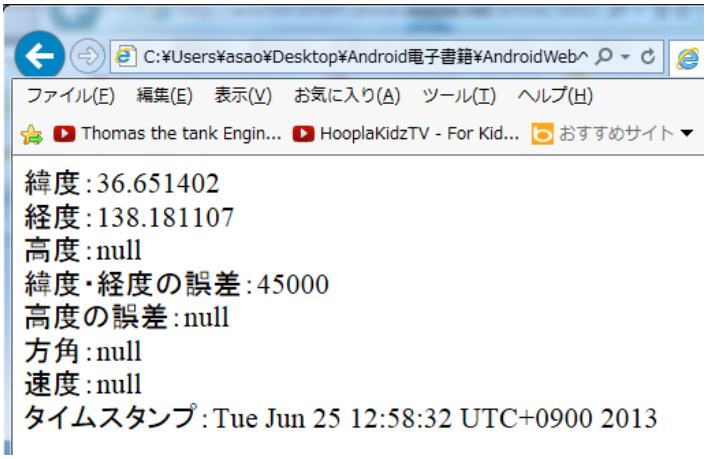
・geo1.html

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0,
maximum-scale=1.0, user-scalable=no" />
<script type="text/javascript">
    navigator.geolocation.getCurrentPosition(success);
    function success(pos) {
        document.getElementById("msg").innerHTML =
            "緯度："+pos.coords.latitude+"<br />"+
            "経度："+pos.coords.longitude+"<br />"+
            "高度："+pos.coords.altitude+"<br />"+
            "緯度・経度の誤差："+pos.coords.accuracy+"<br />"+
            "高度の誤差："+pos.coords.altitudeAccuracy+"<br />"+
            "方角："+pos.coords.heading+"<br />"+
            "速度："+pos.coords.speed+"<br />"+
            "タイムスタンプ："+new Date(pos.timestamp)+"<br />";
    }
</script>
</head>
<body>
<div id="msg"></div>
</body>
</html>
```

パソコンおよびスマホでの各ブラウザの実行結果を以下に示します。

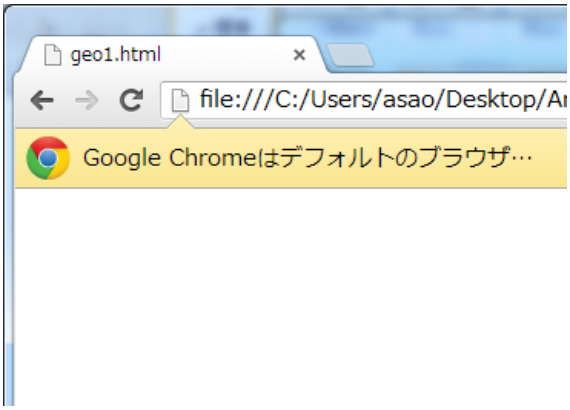
・パソコン Internet Exploler

得られる緯度・経度は県庁所在地またはプロバイダの所在地のようです。



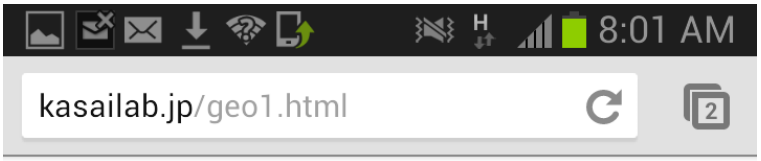
・パソコン Chrome

パソコン Chrome では位置情報を取得できません。



・Android Chrome

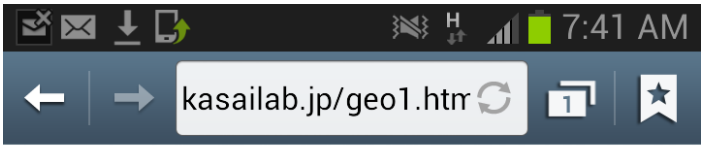
「GPS 機能を使用」を選択した場合。



緯度：36.00405596
経度：138.14256955
高度：809.2999877929688
緯度・経度の誤差：25
高度の誤差：null
方角：null
速度：0
タイムスタンプ：Wed Jun 26 2013 08:00:32
GMT+0900 (JST)

・Android Safari

Android Safari は「GPS 機能を使用」を選択した場合は情報を取得できませんので、以下は「無線ネットワークを使用」を選択した場合の結果です。



緯度：36.0034482
経度：138.1438619
高度：null
緯度・経度の誤差：3814
高度の誤差：null
方角：null
速度：null
タイムスタンプ：Wed Jun 26 2013 07:40:23 (

6章　センサー

HTML5で使用できるセンサーは方位センサー（deviceorientation）と加速度センサー（devicemotion）です。

方位センサーをイベントリスナーに追加するには「deviceorientation」を指定します。event.alphaで方位を、event.betaでピッチを、event.gammaでロールを取得します。

加速度センサーをイベントリスナーに追加するには「devicemotion」を指定します。event.accelerationIncludingGravity.x、event.accelerationIncludingGravity.y、event.accelerationIncludingGravity.zでx,y,z方向の重力加速度を取得します。

ブラウザの向きが変化した時のイベントをイベントリスナーに追加するには「resize」を指定します。window.orientationで向きを取得します。

パソコンでもタブレット端末でも適正な画面サイズで動作させるためには、<meta>タグにviewportを指定する必要があります。タブレット端末でのみ動作する5章～7章のプログラムと18章の応用アプリにはこの<meta>タグを入れてあります。詳細は18章を参照してください。

6-1　方位センサー

方位センサーをイベントリスナーに追加するには「deviceorientation」を指定します。event.alpha で方位を、event.beta でピッチを、event.gamma でロールを取得します。

```
window.addEventListener("deviceorientation", function(event) {
    // event.alpha で方位を取得;
    // event.beta でピッチを取得;
    // event.gamma でロールを取得;
});
```

スマホを飛行機に見立てたときに、機首の上げ下げがピッチで、主翼の先端を上下するのがロールです。飛行機が飛んでいる方角が方位（アジマス）です。

スマホの上端（短辺）を機首と考えたときの方位、ピッチ、ロールの値は以下のような範囲になります。ブラウザの向きとは関係なく取る値は同じです。

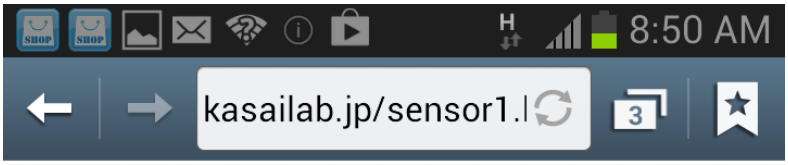
引数 event のプロパティ	機能
alpha	方位。 上端が西の時に 0、南の時に 90、東の時に 180、北の時に 270。
beta	ピッチ。 上端が上の時 90、水平のとき 0、上端が下の時-90。
gamma	ロール。 水平の時 0、左長辺が上の時 90、左長辺が右水平位置のとき 180、左長辺が下の時 270(-90)。

以下は方位センサーを使って方位、ピッチ、ロールを取得して表示するものです。

・sensor1.html

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0,
maximum-scale=1.0, user-scalable=no" />
<script type="text/javascript">
    window.addEventListener("deviceorientation", function(event) {
        var orientation = event.alpha;
```

```
            var pitch = event.beta;
            var roll= event.gamma;
            var result=document.getElementById("result");
            result.innerHTML = "方位(°)<br />"+
                "方位（東西南北:0～360）: "+ orientation.toFixed(1) +"<br />" +
                "ピッチ（x 軸回りの回転角度:-90～90）: "+ pitch.toFixed(1) +"<br />" +
                "ロール（y 軸回りの回転角度：-90～270）: "+ roll.toFixed(1);
    });
</script>
</head>
<body>
<div id="result"></div>
</body>
</html>
```



方位(°)
方位（東西南北:0 ~ 360）：286
ピッチ（x軸回りの回転角度:-90 ~ 90）：1.1
ロール（y軸回りの回転角度：-90 ~ 270）：-2.2

7章　タッチイベント

マウスイベントではタッチパネルのタッチ操作を捕捉できません。タッチ操作を捕捉するには touchstart、touchend、touchmove などのタッチ専用のイベントを使用します。タッチイベントはマウス操作を行うパソコンでは動作しません。onClick イベントはタッチ動作でもマウス操作でも動作します。

タッチイベントは addEventListener で組み込むことができます。touches 配列にタッチしている指のオブジェクトが格納されています。touches[0]で 1 本目の指、touches[1]で 2 本目の指のオブジェクトが取得できます。touches オブジェクトの clientX/clientY プロパティでタッチ位置の座標を取得できます。

デフォルトで与えられているタッチイベントでは直接、タップ、ダブルタップ、ロングタップを区別できませんので、これらを区別する方法を説明します。またピンチ、スクロール、ムーブ、フリングなどの操作を判別する方法を説明します。タッチイベントを利用した応用サンプルとして、羅針盤と相性占いを紹介します。

パソコンでもタブレット端末でも適正な画面サイズで動作させるためには、<meta>タグに viewport を指定する必要があります。タブレット端末でのみ動作する 5 章～7 章のプログラムと 18 章の応用アプリにはこの<meta>タグを入れてあります。詳細は 18 章を参照してください。

7-1　タッチイベントの種類とプロパティ

タッチイベントとして以下があります。タッチイベントは addEventListener で組み込むことができます。

タッチイベントの種類	機能
touchstart	タッチ開始時に発生。
touchend	タッチ終了時に発生。
touchmove	タッチ移動時に発生。
touchcancel	システム割り込み（着信、アラームなど）によるキャンセルで発生。

touches 配列にタッチしている指のオブジェクトが格納されています。touches[0]で 1 本目の指、touches[1]で 2 本目の指のオブジェクトが取得できます。touches オブジェクトには以下のプロパティがあります。

touches のプロパティ	機能
identifier	タッチポイントの ID。
clientX/clientY	クライアント領域（viewport）に対する座標。
pageX/pageY	ページ全体（html 要素）に対する座標。
screenX/screenY	画面の表示領域に対する座標。
target	イベントの発生元。

touchstrat イベントを組み込むには以下のようにします。デフォルトのタッチ動作を無効にするには「event.preventDefault()」を実行します。

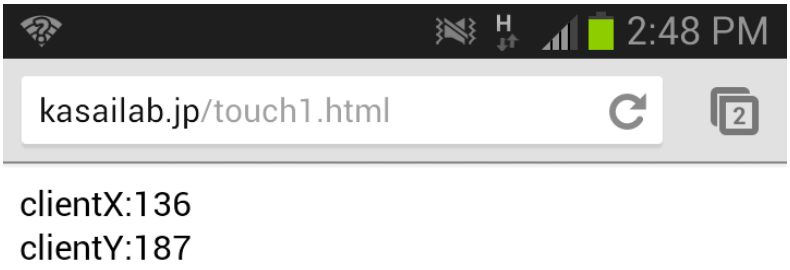
```
document.addEventListener("touchstart", function(event){
    event.preventDefault();
    // event.touches[0].clientX と event.touches[0].clientY でタッチ位置の座標
});
```

以下はタッチムーブ位置の x,y 座標を表示します。

・touch1.html

```
<!DOCTYPE html>
<html>
<head>
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0,
maximum-scale=1.0, user-scalable=no" />
<script type="text/javascript">
	document.addEventListener("touchmove", function(event){
		event.preventDefault();
		var result = document.getElementById("result");
		result.innerHTML =
			"clientX:"+ event.touches[0].clientX+ "<br />" +
			"clientY:"+ event.touches[0].clientY;
	});
</script>
</head>
<body>
<div id="result"></div>
</body>
</html>
```

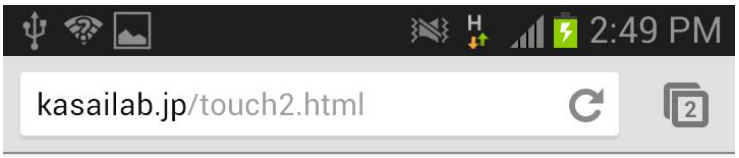


以下はタッチ位置にイメージを移動するものです。

・touch2.html

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0,
maximum-scale=1.0, user-scalable=no" />
<script type="text/javascript">
	document.addEventListener("touchmove", function(event){
		event.preventDefault();
```

```
        var target=document.getElementById("img1");
        target.style.left=event.touches[0].clientX+"px";
        target.style.top=event.touches[0].clientY+"px";
    });
</script>
</head>
<body>
<img id="img1" src="play.png" style="position:absolute;left:50px;top:50px" />
</body>
</html>
```



8章　ドラッグ&ドロップ

ドラッグ&ドロップは mousedown や mouseup などのイベントで実現することができましたが、 HTML5 ではドラッグ&ドロップ専用の新しいイベントや新しいメソッド・属性が追加されています。ドラッグ&ドロップはドラッグ操作とドロップ操作の 2 種類の操作を組み合わせて行います。

draggable 属性を「true」に設定するとドラッグが可能に、「false」に設定するとドラッグが禁止になります。ドラッグ関連イベントとして ondragstart、ondrag、ondragend が、ドロップ関連イベントとして ondragenter、ondragleave、ondragover、ondrop があります。「event.dataTransfer」オブジェクトの setData メソッドと getData メソッドを使えばドラッグ操作とドロップ操作という異なる操作の間でデータをやり取りすることができます。ドロップされたファイルオブジェクト（複数のファイル）は「event.dataTransfer.files」で取得できます。

ドラッグ&ドロップ関連のイベントはマウスイベントの仲間なので、スマホやタブレットなどのタッチパネルでは動作しません。

8-1　draggable 属性

draggable 属性を「true」に設定するとドラッグが可能に、「false」に設定するとドラッグが禁止になります。draggable 属性を指定しないとデフォルト解釈になります。<img>や<a>タグはデフォルトで「true」、他の要素は「false」と解釈されます。

以下はそれぞれの要素に draggable 属性を指定したものです。ドラッグ可能なものは、ドラッグを開始するとドラッグ対象物の薄い色の画像（ゴースト画像）が移動します。ドロップターゲットは設定していないので、ブラウザのクライアント画面にはドロップはできません。ただし、ブラウザのアドレスバーやデスクトップはシステム側でドロップターゲットとして設定されているのでドロップできます。

・drag1.html

```
<!DOCTYPE html>
<html>
<head>
<style>
div.drag {
  width:300px; height:150px; margin:20px; background-color:#CCFF66; border:1px solid #00cc00;
}
</style>
</head>
<body >
<div class="drag" draggable="true">
コンテナはドラッグできますが、イメージは単独でドラッグできません。<br />
<img src="play.png" draggable="false" />
</div>
<div class="drag" draggable="false">
コンテナはドラッグできませんが、イメージは単独でドラッグできます。<br />
<img src="pause.png" draggable="true" />
</div>
<a href="http://www.yahoo.co.jp">Yahoo!へのリンクもドラッグできます。</a>
</body>
</html>
```


drag1.html
file:///C:/Users/asao/Desktop/HTML5/drag1.html
コンテナはドラッグできますが、イメージは単独でドラッグできません。
コンテナはドラッグできますが、イメージは単独でドラッグできません。
コンテナはドラッグできませんが、イメージは単独でドラッグできます。
Yahoo!へのリンクもドラッグできます。

9 章　File API

File API を用いることにより、JavaScript からクライアント側のファイルにアクセス（読み込み専用）することが可能です。ただし、セキュリティを考慮して、アクセス可能なのは、利用者が<input type="file">タグを使って意識的に選択したファイル、あるいはドラッグ＆ドロップしたファイルのみに限られます。ファイル名を指定してファイルオープンするということはできません。

File API を用いてファイルにアクセスするには FileReader オブジェクトをインスタンス化し、readAsXXX メソッドを使ってファイル読み込みを開始します。ファイルの読み込みが完了すると onloaded イベントハンドラが呼び出されますので、そこで event.target.result を使って結果を取得します。

9-1　ファイルオブジェクト

ファイルにアクセスするためには、アクセスするファイルを選択し、それにより取得できた File オブジェクトを使用します。

1.　ファイルの選択方法

ファイルを選択する方法として以下の 2 種類があります。いずれも files オブジェクトに File オブジェクト配列が取得できます。個々の File オブジェクトは files[0]、files[1]、・・・で取得できます。

①<input type="file">タグを使う方法

```
var files = event.target.files;
```

②ドラッグ&ドロップを使う方法

```
var files = event.dataTransfer.files;
```

2.　ファイル情報の取得

File オブジェクトの主なプロパティとして以下があります。

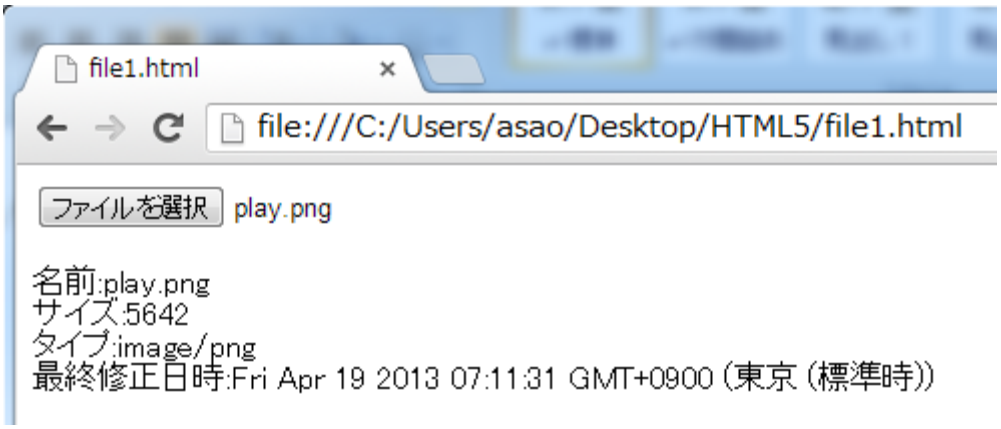
File オブジェクトのプロパティ	機能
name	ファイル名。
type	MIME タイプ。
size	サイズ（バイト）。
lastModifiedDate	最終修正日時。

以下は選択したファイルの情報を表示します。選択したファイルは一つと仮定しています。

・file1.html

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript">
    function disp(event) {
        var file = event.target.files[0]; // 選択されているファイルは 1 つと仮定
        if (file) {
            var msg = "名前:" + file.name + "<br />";
```

```
                msg += "サイズ:" + file.size + "<br />";
                msg += "タイプ:" + file.type + "<br />";
                msg += "最終修正日時:" + file.lastModifiedDate + "<br />";
                document.getElementById("msg").innerHTML = msg;
            }
        }
</script>
</head>
<body>
<input type="file" onchange="disp(event);">
<p id="msg"></p>
</body>
</html>
```



10 章　Web Storage

HTML5 では、クッキーに代わるデータ保存の仕組みとして、「Web Storage」と呼ばれる機能を利用できます。Web Storage とクッキーの機能の差異は、以下のとおりです。

機能	クッキー	Web Storage
保存容量	4KB。	5MB。
データの有効期限	あり。	なし。
セキュリティ	すべてのリクエストに対してサーバにデータを自動送信するのでセキュリティが低い。	データを利用する時のみ送信するのでセキュリティが高い。

Web Storage は Web ページで使用するデータをブラウザ外に「Key-Value」型で保存する機能で、保存場所の違いにより以下の 2 つの種類があります。

・セッションストレージ（sessionStorage）

ブラウザとサーバーが接続状態にある間だけ、ブラウザが使用しているメモリー内にデータを保管します。ブラウザが閉じればその内容は消えます。sessionStorage オブジェクトを使用してデータの保存と読みとりを行います。

・ローカルストレージ（localStorage）

ローカルボリューム（ハードディスク）にデータを保管します。ブラウザが閉じてもその内容は消えません。localStorage オブジェクトを使用してデータの保存と読みとりを行います。

セッションストレージとローカルストレージはデータの保管場所が異なること以外は同じです。この章では主にローカルストレージについて解説しています。プログラム中の localStorage を sessionStorage に置き換えるだけで、セッションストレージでも同様な処理が行えます。

WebStorage を動作させる Web ページをローカルから起動しても動作しません。読者が管理するサイトがあれば、そこに Web ページを置くか、Apatche などのサーバー環境が構築されていれば「http://localhost」に Web ページを置いて起動します。「http://localhost」については「12 章　XMLHttpRequest」を参照してください。

10-1　ストレージへのデータの保存と読みとり

Web Storage で扱うデータは「キー、値」のペアになります。データを保存するには setItem メソッドを使って以下のようにします。

localStorage.setItem(キー,値);

指定したキーの値を読み取るには getItem メソッドを使って以下のようにします。

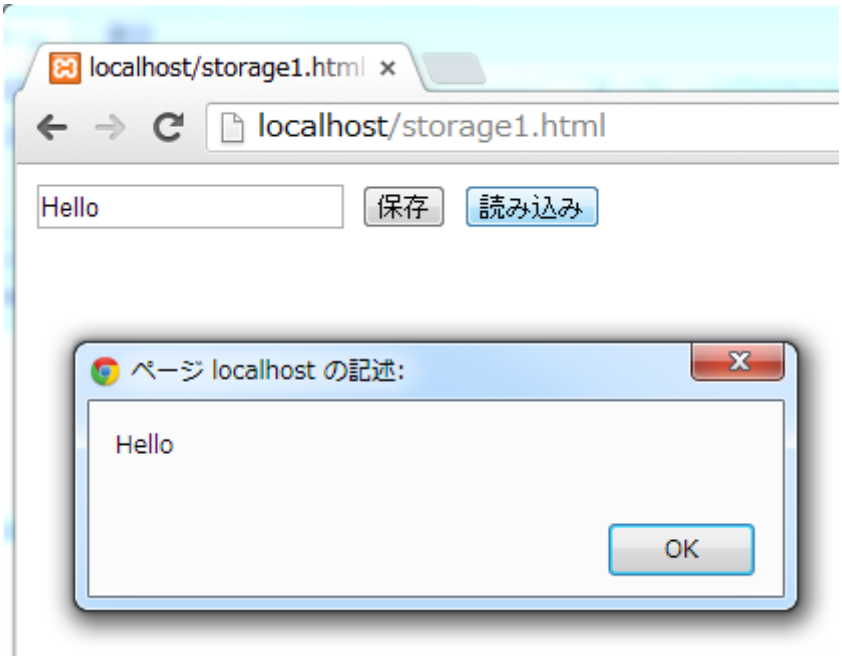
変数 = localStorage.getItem(キー);

以下は<input type="text">に入力された文字を「message」というキーで保存し、読みとります。

・storage1.html

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript">
	function saveStorage()
	{
		var target = document.getElementById("input");
		var str = target.value;
		localStorage.setItem("message",str);
	}
	function loadStorage()
	{
		var msg = localStorage.getItem("message");
		alert(msg);
	}
</script>
</head>
<body>
<input type="text" id="input">
<input type="button" value="保存" onclick="saveStorage();">
<input type="button" value="読み込み" onclick="loadStorage();">
```

</body>
</html>


localhost/storage1.html
localhost/storage1.html
Hello
保存
読み込み
ページ localhost の記述:
Hello
OK

11 章　Indexed Database

いくつかのブラウザでは当初、Web SQL Database を使ったデータベースを行っていました。ところが、W3C では Web SQL Database を非推奨とし、ローカルボリュームへのデータ保存については Web Storage と Indexed Database を使用するように推奨することになりました。

Indexed Database は SQL に依存しないデータベースです。SQL のクエリーを書くのではなくメソッドを呼び出してデータアクセスできるように設計されています。Indexed Database を利用するにはインデックスデータベースをオープンし、オブジェクトストアを介してデータベースにアクセスします。オブジェクトストアとは「キー、値」型のオブジェクトで、「キー」を元にデータにアクセスします。オブジェクトストア内にインデックスを設定し、カーソルを使用すると、オブジェクトストア内のデータをインデックス検索できます。インデックスの範囲（レンジ）を指定して検索することもできます。

11-1　データベースのオープン

Indexed Database を使ってデータベースをオープンする手順は以下です。

1.　IDBFactory の取得

IDBFactory はデータベースのオープン、削除などを行うためのオブジェクトクラスです。W3C の仕様では indexedDB という名前ですが、ブラウザにより先頭に「moz」や「ms」などのプリフィックスが付く場合がありますので、||演算子で列挙します。以下により indexedDB に IDBFactory 型のオブジェクトが取得されます。

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

2.　IDBRequest の取得

上で取得した indexedDB に対し「open」メソッドを呼び出し、指定のデータベースを開きます。オープンできるとそのリクエストが IDBRequest 型オブジェクトとして openRequest に取得されます。

```
var openRequest = indexedDB.open("データベース名"[,バージョン]);
```

3.　onupgradeneeded イベントハンドラの設定

onupgradeneeded イベントハンドラはデータベースが新たに作成される時またはバージョンが更新されたときに呼びだされます。引数の「event.target.result」にデータベースにアクセスするためのオブジェクトが渡されます。createObjectStore メソッドを使ってオブジェクトストアを作成します。引数にはオブジェクトストアの名前とキーの名前を指定します。

```
openRequest.onupgradeneeded = function(event) {
    db = event.target.result;
    var store = db.createObjectStore("mystore", { keyPath: "mykey"});
}
```

4.　onsuccess イベントハンドラの設定

2 回目のオープン以後に呼び出されます。引数の「event.target.result」にデータベースにアクセスするためのオブジェクトが渡されます。onupgradeneeded イベントハンドラのようなオブジェクトストアの作成処理は行いません。

```
  openRequest.onsuccess = function(event) {
        db = event.target.result;
  }
```

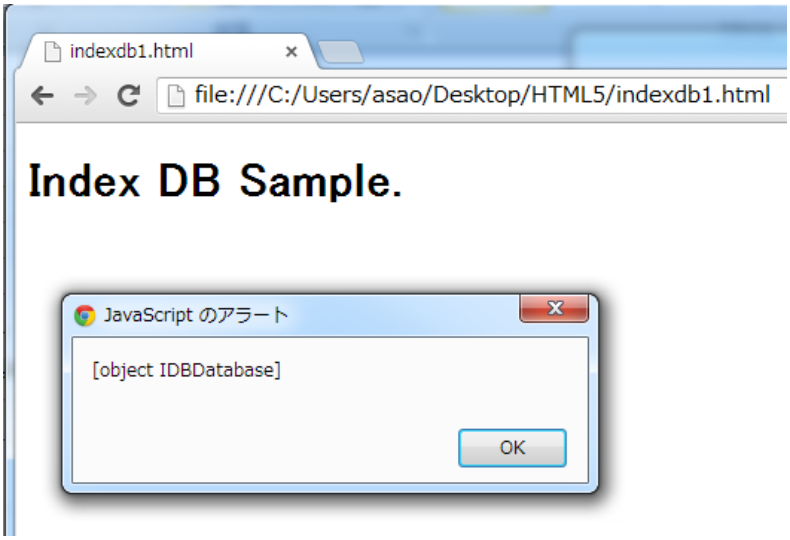
以下は「mydb」という名前でバージョンを「1.0」とするデータベースをオープンします。オブジェクトストアを「mystore」という名前で生成し、そこで使用するキーの名前を「mykey」とします。indexedDB が null なら、使用しているブラウザは Indexed Database に未対応というメッセージを表示します。

・indecdb1.html

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript">
    function initial(){
        var db;
        var indexedDB = window.indexedDB || window.mozIndexedDB ||
window.msIndexedDB;
        if (indexedDB) {
            // indexedDB.deleteDatabase("mydb");   mydb 削除
            var openRequest = indexedDB.open("mydb", 1.0);
            openRequest.onupgradeneeded = function(event) {
                db = event.target.result;
                var store = db.createObjectStore("mystore", { keyPath: "mykey"});
            }
            openRequest.onsuccess = function(event) {
                db = event.target.result;
                alert(db);
            }
        }
        else {
            alert("このブラウザでは Indexed DataBase API は使えません。");
        }
    }
</script>
</head>
```

```
<body onload="initial();">
<h1>Index DB Sample.</h1>
</body>
</html>
```

「注」データベースを削除するには「indexedDB.deleteDatabase("mydb");」とします。開発段階で、データベースの内容がごちゃごちゃになってしまったら一度データベースを削除するか別の名前のデータベースをオープンしてください。



12 章　XMLHttpRequest

HTTP（HyperText Transfer Protocol）は Web サーバとクライアント(Web ブラウザなど)がデータを送受信するのに使われるプロトコルです。クライアントがサーバにリクエストメッセージを送信して結果を受け取るリクエスト-レスポンス型のプロトコルです。

XMLHttpRequest は、JavaScript を使用してクライアントとサーバー間の HTTP リクエストによるデータ転送を行うための API です。Web ページから HTTP リクエストを送り、HTML や XML のデータを受信する際にページ遷移を伴う必要がなく、非同期に通信を行うことができます。

XMLHttpRequest は Microsoft によって開発されましたが、W3C により標準化が進められていて現在の仕様は XMLHttpRequest Level 2(XHR2)と呼ばれるものです。XMLHttpRequest は HTML5 の仕様ではありませんが、HTML5 の技術が実装されていくのにあわせてブラウザが対応してきた「HTML5 の周辺技術」という位置づけになります。XMLHttpRequest は Ajax で非同期通信を行うための標準的な方式の一つです。XMLHttpRequest は Same-Origin Policy のため、異なるドメインのリソースへのアクセスが出来ません。ただし XMLHttpRequest Level 2 ではこの制限が解除されています。

この章の例は Apatche サーバーを稼働状態にして、「localhost」が使用できる状態で実行します。ファイルは「http://localhost」に置き、ブラウザのアドレスバーにたとえば「http://localhost/ ajax1.html」などを指定して実行します。

サーバー側のプログラムは PHP で記述しています。PHP については「Appendix3　PHP」を参照してください。

12-1　XMLHttpRequest を動作させるサーバー環境

XMLHttpRequest を動作させるにはサーバー環境を構築する必要があります。ここでは XAMPP（ザンプ、エグザンプ）を使用します。

1.　XAMPP とは

XAMPP は Web アプリケーションの開発に必要なフリーソフトウェアをひとつのパッケージとしてまとめたもので、以下の 4 つの主要ソフトで構成されています。

・Apache：HTTP Server
・MySQL：SQL データベースサーバ
・PHP：Web プログラミング言語
・Perl：Web プログラミング言語

通常サーバーサイドプログラミングはサーバー用とクライアント用に 2 台のコンピュータが必要になります。また Web アプリケーション開発を行うためには Apache、MySQL、PHP などのソフトをそれぞれ個別にインストールしなければならず、さらに相互の設定が複雑でした。このため個人レベルでサーバーサイドプログラミングを行うには敷居が高い状態でした。XAMPP を利用すると以下のような利点があります

・XAMPP でインストールすれば Web アプリケーション開発に必要なソフトが一括してインストールでき、各種設定をしなくてもすぐに稼働できる環境になります。

・XAMPP 環境では 1 台のパソコン上にサーバーとクライアントを実現することができるため、Web アプリケーションの開発が容易になり、個人レベルで気軽に使用することができます。

2.　XAMPP のインストール

XAMPP は apachefriends.org から提供されています。Windows 版は次のサイトから無料でダウンロードできます。

http://www.apachefriends.org/jp/xampp-windows.html

「インストーラ」を選択してダウンロードします。

ダウンロード

XAMPP

XAMPP for Windowsには、3つの種類が用意されています:

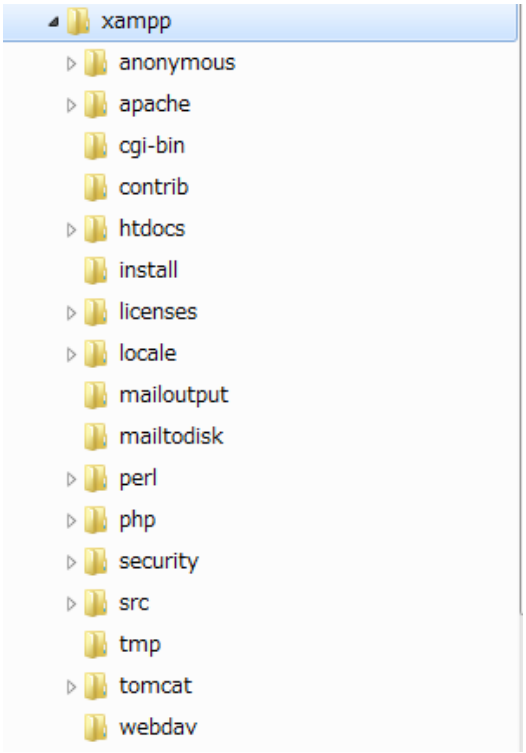
インストーラ
簡単で安全: 快適なインストーラ内蔵のXAMPP
ZIPアーカイブ
こだわり派向け: 通常のZIPアーカイブでのXAMPP
7zipアーカイブ
こだわり派で時間の無い人向け: 7zipアーカイブでのXAMPP

注意:
ファイルを展開すると、誤ってウイルス警告が検出される場合があります。

XAMPP Windows版 1.8.1, 2012/9/30

バージョン	サイズ	内容
XAMPP Windows版 1.8.1		Apache 2.4.2, MySQL 5.5.27, PHP 5.4.7, OpenSSL 1.0.1c, phpMyAdmin 3.5.2.2, XAMPP Control Panel 3.1.0, Webalizer 2.23-04, Mercury Mail Transport System v4.62, FileZilla FTP Server 0.9.41, Tomcat 7.0.30 (with mod_proxy_ajp as connector), Strawberry Perl 5.16.0.1 Portable For Windows 2000, XP, Vista, 7.
インストーラ	99 MB	インストーラ MD5 checksum: 2c067c31725fda3c71c6d43483b4df4c
ZIP	184 MB	ZIP アーカイブ MD5 checksum: 924e9cdc0fc49984e0c4916aa8f31c18
7zip	84 MB	7zip アーカイブ MD5 checksum: 462f6bc3c9e96a8c9228927ff8e0d217

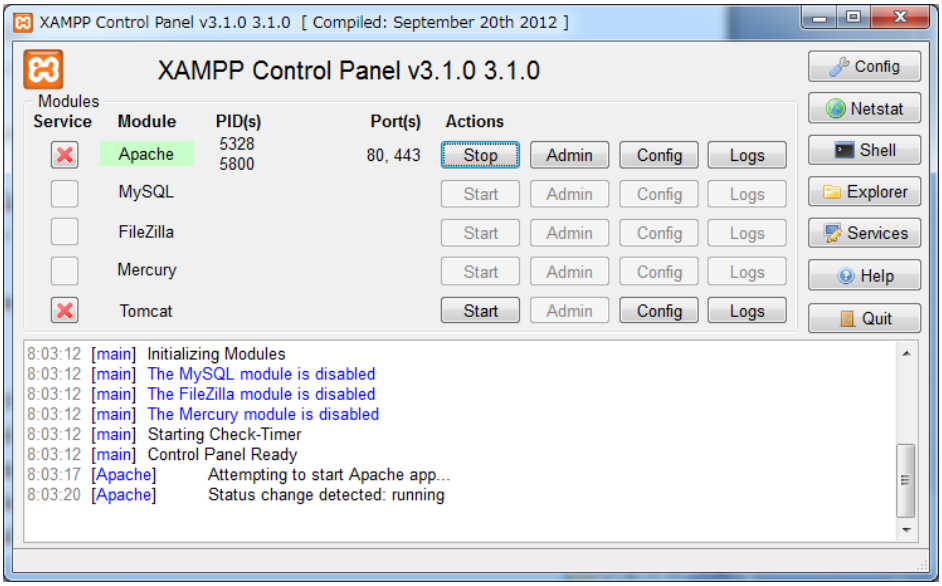
以下のような xampp フォルダに展開されます。



- xampp
  - anonymous
  - apache
  - cgi-bin
  - contrib
  - htdocs
  - install
  - licenses
  - locale
  - mailoutput
  - mailtodisk
  - perl
  - php
  - security
  - src
  - tmp
  - tomcat
  - webdav

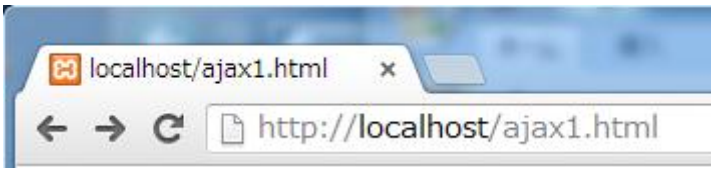
3. XAMPP の管理

「スタート」―「XAMPP Control Panel」で XAMPP コントロールパネルが開くので各モジュールの起動、停止、状態チェックなどができます。Apatche サーバーを起動状態にします。



4. ソースの保管場所と localhost へのアクセス

ソースファイルは「xampp/htdocs」フォルダに置きます。この場所がサーバーの「localhost」になります。「localhost」は使用しているパソコンで立ち上げているサーバーへアクセスする場合の IP アドレスです。「localhost」の具体的な IP アドレスは「127.0.0.1」です。「xampp/htdocs」フォルダに配置された ajax1.html にアクセスするには「http://localhost/ajax1.html」とします。



「xampp/htdocs」フォルダの下にサブフォルダ「data」を作成し、そこに test.html を配置した場合の URL は「http://localhost/data/test.html」となります。

13 章　WebSocket

WebSocket は、インターネットの標準化団体である W3C と IETF（The Internet Engineering Task Force）が定めるサーバーとブラウザとの間の双方向通信用の技術規格です。WebSocket は、サーバーとブラウザ間に「ソケット」接続を確立する API を定義しています。クライアントとサーバーの間に持続的接続があり、どちらの側からでも、いつでもデータの送信を開始できるのが特徴です。

HTML5 では JavaScript を使って WebSocket を利用することができます。現在は W3C の HTML5 仕様から分離されてしまいましたが、次世代のウェブアプリケーションの中核を担うテクノロジーとして期待されています。

WebSocket で送受信が可能なフォーマットは文字列（テキスト）、Blob、ArrayBuffer の 3 つです。Blob と ArrayBuffer は JavaScript でバイナリデータを扱うためのフォーマットで、そのまま send メソッドに指定するだけで送信できます。

WebSocket は、HTTP ではなく独自のプロトコルを利用して通信を行うため、Web サーバーとは別に専用のサーバーを用意しなければいけません。

この章の例は Apatche サーバーを稼働状態にして、「localhost」が使用できる状態で実行します。ファイルは「php-websocket-master」フォルダ下の「server」、「server/lib/WbSocket/Application」、「client」などにに置き、ブラウザのアドレスバーにたとえば「http://localhost/ php-websocket-master/client/socket1.html」などを指定して実行します。

サーバー側のプログラムは PHP で記述しています。PHP については「Appendix3　PHP」を参照してください。

13-1　WebSocket を動作させるサーバー環境

WebSocket を利用したアプリケーション開発には、WebSocket の機能を盛り込んだサーバサイド実装のプロダクトが必要です。主なプロダクトとして以下があります。

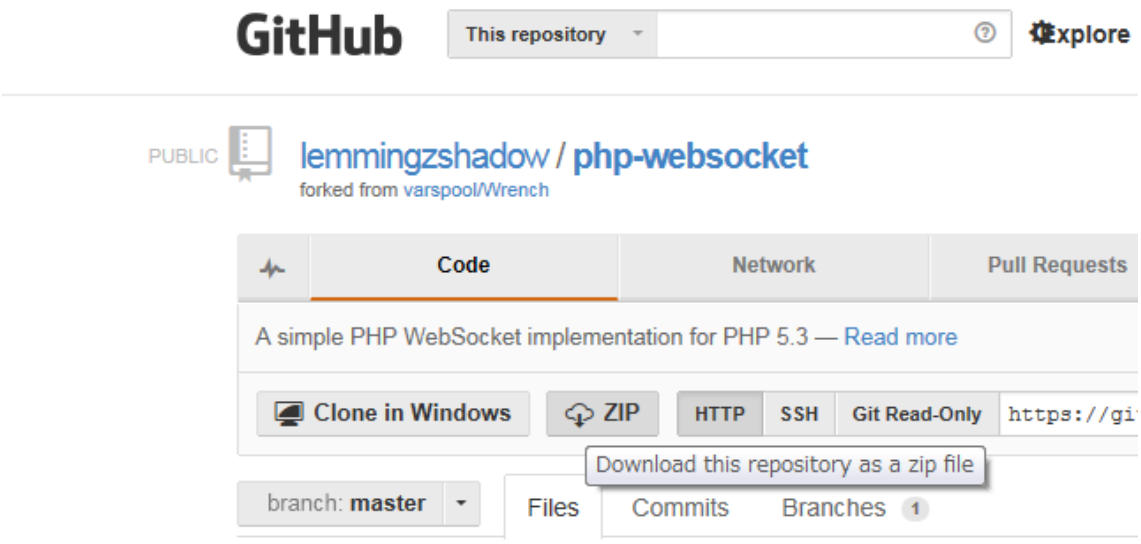
プロダクト	概要
Jetty	Java で作成された Web サーバーで、WebSocket もサポートしている。
Kaazing WebSocket Gateway	Java で作成された WebSocket サーバー。
node.js	サーバサイド JavaScript。
PHP Websocket Class	PHP の Websocket サーバーライブラリ。
SuperWebSocket	.NET で作成された WebSocket サーバー。
WCF WebSockets	WebSocket 機能が拡張された WCF（Windows Communication Foundation）。

本書では「PHP Websocket Class」を使用します。「PHP Websocket Class」を使用するにあたり PHP や Apache サーバーなどの環境を構築する必要があります（12 章参照）。

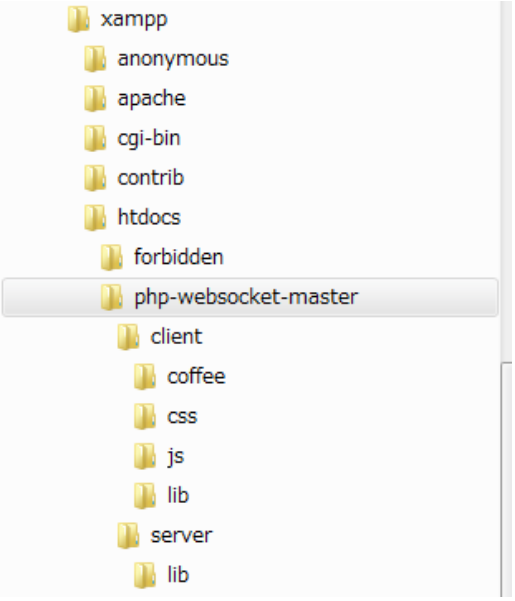
PHP Websocket Class は次のサイトからライブラリを無料でダウンロードできます。

https://github.com/lemmingzshadow/php-websocket

以下の画面の「ZIP」からダウンロードして展開します。



展開されたフォルダー「php-websocket-master」を適当な場所に配置します。ここでは Apache サーバーの localhost にあたる「xampp/htdocs」に配置します。



14 章　Server-Sent Events（EventSource）

Server-Sent Events は HTTP でサーバーPush を簡単に行うための API です。通常の HTTP 通信においては、クライアントからサーバーへリクエストを送り、サーバーからレスポンスが帰ってくるとそれで終了します。Server-Sent Events では、HTTP 通信でサーバーから応答しても、接続を終了せずに維持します。こうすることで、その接続を利用してサーバーからメッセージを継続的に送ることができます。ただしデータはサーバーから一方的に送るだけで、クライアントからサーバーにはデータを送ることはできません。

クライアント側は EventSource オブジェクトを使ってサーバーに接続しデータを受信します。サーバー側は「text/event-stream」という MIME タイプでレスポンスを返します。

この章の例は Apatche サーバーを稼働状態にして、「localhost」が使用できる状態で実行します。ファイルは「http://localhost」に置き、ブラウザのアドレスバーにたとえば「http://localhost/ source1.html」などを指定して実行します。サーバー側のプログラムは PHP で記述しています。PHP については「Appendix3　PHP」を参照してください。なお、Server-Sent Events のレスポンスの文字コードは必ず UTF-8 でなければならないという制約があるため、この章のサーバー側に置く PHP ファイルは UTF-8 形式で保存してください。

14-1　EventSource オブジェクトによるデータ授受の概要

EventSource オブジェクトを接続先の url を引数にしてインスタンス化すると、サーバー側イベントの受信と解析がバックグラウンドで自動的に開始されます。EventSource ではクロスドメイン通信（異なるドメイン間の通信）を許可していませんので、url には同一ドメインの URL しか指定できません。

```
var EventSource = window.EventSource || window.MozEventSource;
var source = new EventSource("event1.php");
```

イベントを送信するサーバサイドのスクリプトでは、MIME タイプが「text/event-stream」での応答が必要です。

```
header("Content-Type: text/event-stream");
```

サーバー側では以下のように「data:」で始まる「データ」を送信します。空白行が送信データの終わりとして認識されますので、テキストの最後は「¥n¥n」とします。

```
echo "data:データ¥n¥n";
```

クライアント側はデータ受信すると onmessage プロパティで指定されたイベントハンドラが呼び出され、「event.data」で受信データを取得することができます。レスポンスの文字コードは必ず UTF-8 でなければなりません。

```
source.onmessage = function(event){
    // event.data で受信データを取得
};
```

以下は$girls 配列のデータを送信します。クライアント側は取得したデータをテキストエリアに表示します。「終わり」というテキストを受信したら、サーバーとの接続を解除します。

・event1.php(サーバー)

```
<?php
    header("Content-Type: text/event-stream");
    header("Cache-Control: no-cache");
```

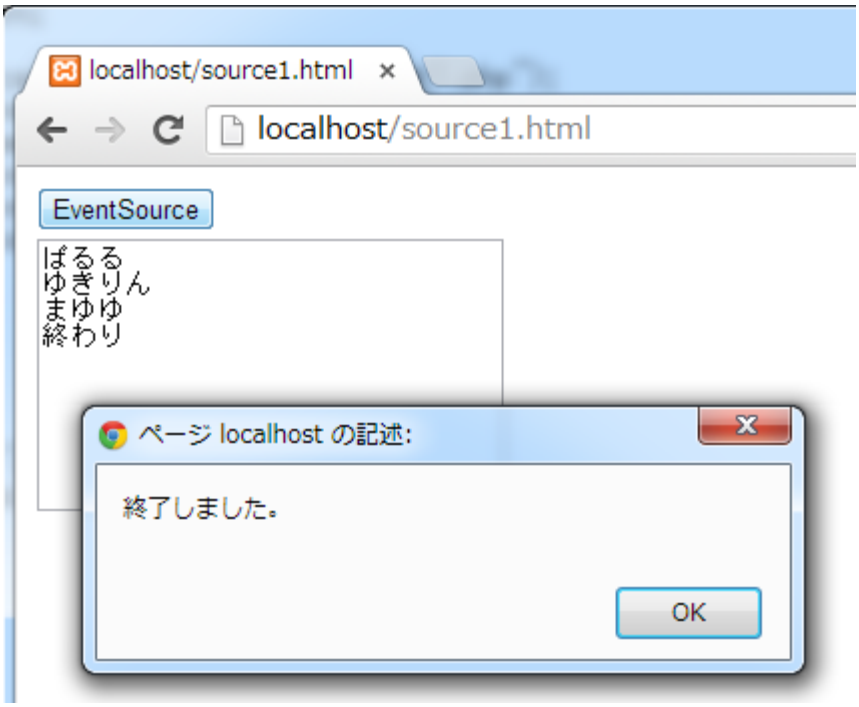
```
    $girls = array("ぱるる","ゆきりん","まゆゆ","終わり");
    foreach ($girls as $girl){
        echo "data:".$girl."¥n¥n";
    }
    flush();
?>
```

「注」Server-Sent Events のレスポンスの文字コードは必ず UTF-8 でなければならないという制約があるあめ、この章のサーバー側に置く PHP ファイルは UTF-8 形式で保存してください。

・source1.html(クライアント)

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript">
    var EventSource = window.EventSource || window.MozEventSource;
    function doAction(){
        if (!EventSource){
            alert("EventSource が利用できません。");
            return;
        }
        var source = new EventSource("event1.php");
        source.onmessage = function(event){
            document.getElementById("msg").value+=event.data+"¥n";
            if (event.data == "終わり"){
                event.target.close();
                alert('終了しました。');
            }
        };
    }
</script>
</head>
<body>
<button onclick="doAction();">EventSource</button><br />
<textarea id="msg" rows="10" cols="30"></textarea>
```

</body>
</html>


localhost/source1.html
localhost/source1.html
EventSource
ぱるる
ゆきりん
まゆゆ
終わり
ページ localhost の記述:
終了しました。
OK

15 章　Web Messaging

一般にコードによる他のウィンドウやフレームへのアクセスは、セキュリティ上の理由から、同じオリジンのドキュメントの間だけに制限されています。しかし、異なるオリジンのドキュメント間でも相手を限定するなら、メッセージのやり取りを行えるようにした方が良い場合があります。この問題を解決するのが、Web Messaging です。

Web Messaging は異なるドメイン間や異なるウィンドウ間で、通信を行うための 2 つの API を定義しています。一つはクロスドキュメントメッセージング（cross-document messaging ）で postMessage メソッドと onmessage イベントハンドラで行います。もう一つはチャンネルメッセージング(channel messaging)で MessageChannel オブジェクトを用いて行います。

この章の例は Apatche サーバーを稼働状態にして、「localhost」が使用できる状態で実行します。ファイルは「http://localhost」と「http://kasailab.jp/」（筆者の Web サイト）に置きます。

15-1　クロスドキュメントメッセージング

クロスドキュメントメッセージングは、異なるオリジンの間での通信を安全に行うことを目的とした API で、そのために、メッセージの送受信を専用のメソッドとイベントだけに限定し、送受信時に相手先のオリジンを確認できるようになっています。ここでは<iframe>に対しクロスドキュメントメッセージングを行う方法を説明します。

1.　<iframe>タグ

<iframe>は、「インラインフレーム」のことで、HTML のテキスト中にフレームを埋め込むための要素です。通常の分割で表示するフレームセット<frameset>とは違い、<body>内の好きな場所に指定したサイズの窓を表示することが可能です。

<iframe>タグに指定する属性として以下があります。

<iframe>タグの属性	機能
frameborder	フレーム境界線の有無（yes,no,auto,1,0）。
bordercolor	境界線の RGB 値。
scrolling	スクロールの有無（yes,no,auto）。
marginwidth	フレーム内の左右マージン（ピクセル数）。
marginheight	フレーム内の上下マージン（ピクセル数）。
width	インラインフレームの横幅（ピクセル数）。
height	インラインフレームの高さ（ピクセル数）。
align	フレームに回り込む要素の位置（top,middle,bottom,left,right）。

フレームのウインドウオブジェクトは次の 2 つの方法で取得できます。

```
var iframe = window.frames[0];
```

または、

```
var iframe = document.getElementById("inframe");
```

2.　クロスドキュメントメッセージングを行う手順

iframe の利用側と iframe 側でのクロスドキュメントメッセージングを行う手順を説明します。

・iframe の利用側

<iframe>の src に利用する iframe の URL を指定します。同じローカルホストなら src="http://localhost/iframe1.html"

異なるドメインなら、たとえば、
src="http://kasailab.jp/iframe1.html"
などを指定します。

iframe 側にデータを送信するには postMessage メソッドを使用します。

```
var iframe = document.getElementById("inframe");
iframe.contentWindow.postMessage("送信データ","送信先のオリジン");
```

送信先のオリジンには「http://kasailab.jp/」や「http://localhost」を指定します。ワイルドカードの「*」を指定すると「すべてのサイト」に対し送信可能となります。この場合、悪意のある Web サイトに対して不用意にメッセージを送信してしまう危険が伴いますので、特定のオリジンに限定した方が良いでしょう。

・iframe 側
message イベントハンドラを設定し、window オブジェクトの message イベントを監視することで、メッセージの受信を行えます。

```
window.addEventListener("message", function(event){
    // event.data でデータを取得
});
```

message イベントハンドラの引数に渡される MessageEvent オブジェクトのプロパティは以下です。

MessageEvent のプロパティ	機能
data	送られて来たデータ。
origin	送信元のオリジン。クロスドキュメントメッセージングと Server-Sent Events で使用。
lastEventId	メッセージイベントの ID。Server-Sent Events で使用。
source	送信元の window オブジェクト。クロスドキュメントメッセージングで使用。
ports	送信元から送られてきた MessagePort オブジェクトの配列。チャネルメッセージングで使用。

以下の例は iframe の利用側から現在時間を iframe 側に送信し、iframe に送信元のオリジンと受信データを表示します。message1.html は「http://localhost」に、iframe1.html は「http://kasailab.jp/」（筆者の Web サイト）にあるものとしています。別のドメインを設定できない環境ではどちらも「http://localhost」に置いてください。この章のその他の例も同様です。

・message1.html

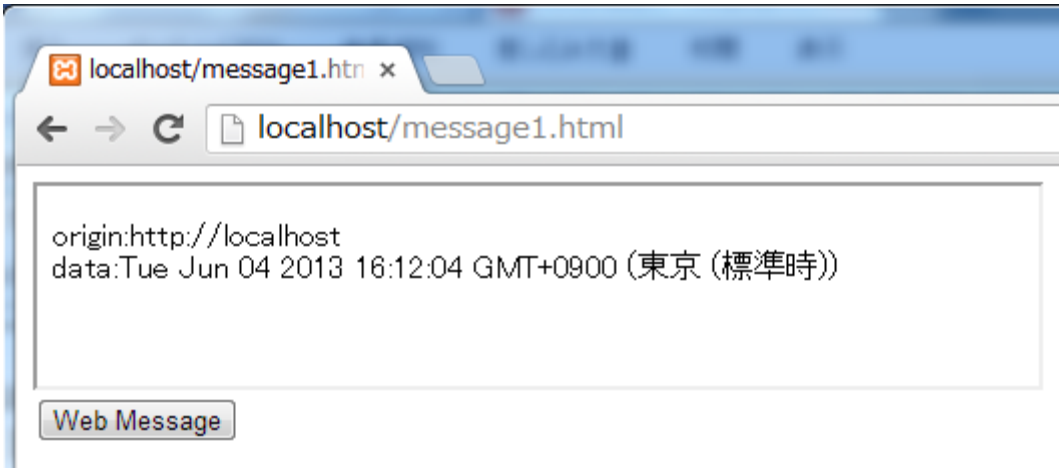
```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript">
    function doAction(){
        var val = new Date();
        var iframe = document.getElementById("inframe");
        iframe.contentWindow.postMessage(val,"http://kasailab.jp/");
    }
</script>
</head>
<body>
<iframe id="inframe" src="http://kasailab.jp/iframe1.html" frameborder="1"
width="500px" height="100px"></iframe><br />
<button onclick="doAction();">Web Message</button>
</body>
</html>
```

・iframe1.html

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript">
    window.addEventListener("message", function(event){
        document.getElementById("msg").innerHTML =
"origin:"+event.origin+"<br />data:"+ event.data;
    });
</script>
</head>
```

<body>

</body>
</html>



「注」iframe1.html をローカルホストに置いた場合、Chrome では更新したファイルが反映されないことがあります。この場合<iframe>の src の URL に「?=」で始まるパラメータを追加すると更新されます。「=」の後の文字はなんでもかまいません。

・message1.html
iframe.contentWindow.postMessage(val,"http://localhost/");

・iframe1.html
<iframe id="inframe" src="http://localhost/iframe1.html?=2013-06-11"

16 章　アプリケーションキャッシュ

Web アプリをオフラインで動作させるためにはデータをキャッシュしておく必要があります。各ブラウザにはキャッシュ機能がついていますが、統一性がなく、アプリケーションをキャッシュするには機能も低いです。そこで HTML5 では、ApplicationCache インターフェースによりアプリケーションキャッシュを統一的に行えるようにしています。

ブラウザがキャッシュする必要があるリソースをキャッシュ・マニフェストファイルに記述しておきます。JavaScript からアプリケーションキャッシュを制御するには applicationCache オブジェクトを使用します。

この章の例は Apatche サーバーを稼働状態にして、「localhost」が使用できる状態で実行します。ファイルは「http://localhost」に置き、ブラウザのアドレスバーにたとえば「http://localhost/appcache1.html」などを指定して実行します。

16-1　アプリケーションキャッシュの使い方

アプリケーションキャッシュを使うにはキャッシュ・マニフェストファイルを使用して、キャッシュされたページの更新を行います。

1.　キャッシュ・マニフェストファイル

キャッシュ・マニフェストファイルは、オフライン・アクセスに備えてブラウザがキャッシュする必要があるリソースを列挙したテキストファイルです。

キャッシュ・マニフェストファイルは拡張子を、「.appcache」とし、文字コードは「UTF-8」を使用します。キャッシュ・マニフェストファイルは、1 行目に「CACHE MANIFEST」という文字列を記述し、以後の行に以下のセクションに対し内容を記述します。

セクション	内容
#	コメント。
CACHE:	キャッシュしたいファイルを指定します。ここで指定したファイルは、アクセス時にすべてダウンロードされ、ローカルに保存されます。
NETWORK:	必ずオンラインから情報を取得したいファイルを指定します。ここで指定したファイルはローカルに保存されません。
FALLBACK:	アクセスに失敗した場合、代わりに使用するファイルを指定します。

たとえばテスト Web ページの「appcache1.html」とそこで使われているイメージファイルの「play.png」をキャッシュするには manifest1.appcache マニフェストファイルに以下のように記述します。

・manifest1.appcache

```
CACHE MANIFEST
#VERSION 1
CACHE:
appcache1.html
play.png
```

アプリケーションキャッシュを利用する Web ページの「appcache1.html」では、<html> タグの manifest 属性にマニフェストファイル名を指定します。

・appcache1.html

```
<!DOCTYPE html>
<html manifest="manifest1.appcache">
・
・
</html>
```

マニフェストファイルは、MIME タイプを「text/cache-manifest」で提供する必要があります。この MIME タイプを Apache で提供するには、「.htaccess」設定ファイルを作成し以下の内容をに追加します。

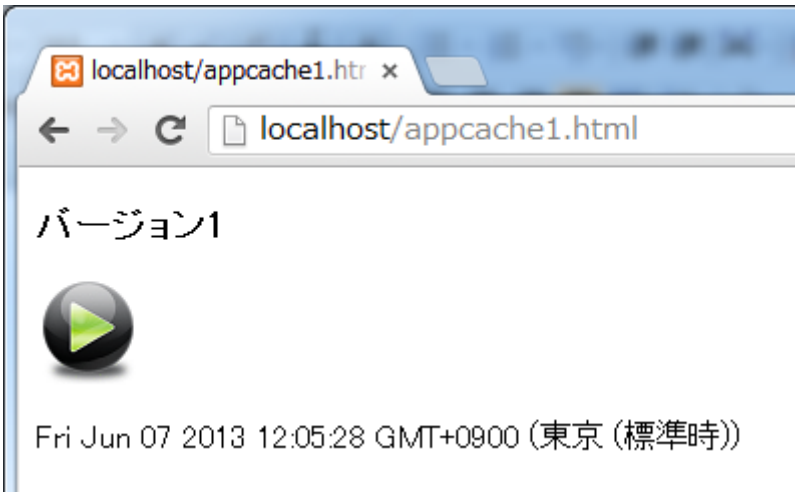
・.htaccess

```
AddType text/cache-manifest .appcache
```

「appcache1.html」、「manifest1.appcache」、「.htaccess」ファイルは「http://localhost」に置きます。「appcache1.html」は「http://localhost/appcache1.html」で実行します。また、Apache サーバーを稼働状態にしておきます。これらの内容はこの章の他の例でも同様です。
「appcache1.html」の全文は以下です。

・appcache1.html

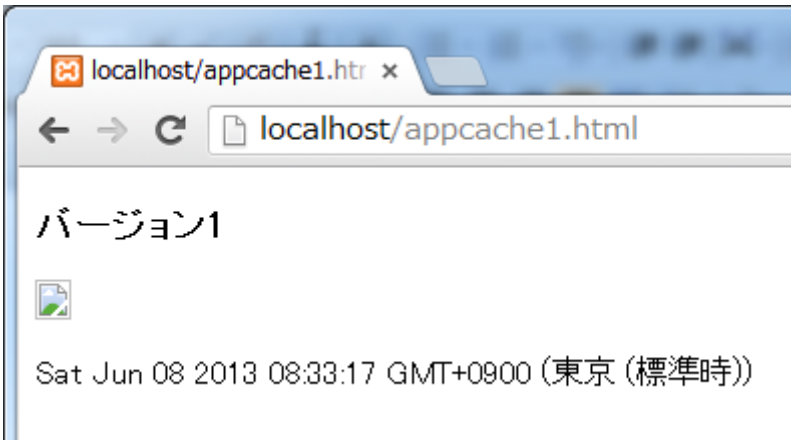
```
<!DOCTYPE html>
<html lang="ja" manifest="manifest1.appcache">
<head>
<script type="text/javascript">
    function initial(){
        document.getElementById("time").innerHTML = new Date();
    }
</script>
</head>
<body onload="initial();">
<h3>バージョン 1</h3>
<img src="play.png" /><br />
<p id="time"></p>
</body>
</html>
```



2.　キャッシュされたページの更新

以上により、「appcache1.html」をロードする際に「manifest1.appcache」を参照して、指定されているファイルがキャッシュされることになります。そして 2 回目以降のアクセスでは、キャッシュされた内容が表示されるようになります。たとえば、「appcache1.html」ファイルの play.png を play1.png のように存在しないファイル名に変更し、ブラウザの更新を行ってもキャッシュが使われるため、play.png は変更が反映されずそのまま表示されることになります。これに対し時間は更新されます。これは、ページを読み込んだ後で JavaScript によって動的に設定されているものなので、キャッシュとは無関係に動作するためです。

キャッシュされたページの更新を行うためにはマニフェストファイルの内容を変更します。マニフェストファイルの内容が更新されると、それによってキャッシュされているファイルも自動的に更新されます。マニフェストファイルの内容を変更するといっても、キャッシュするファイルが同じならそこを変更するわけにはいきませんので、コメントを利用します。現在「manifest1.appcache」マニフェストファイルには「#VERSION 1」というコメントが入っているので、これを「#VERSION 2」のように変更してからブラウザの更新を行うと今度は「play1.png」が反映され、このファイルはないので、イメージは×印で表示されます。



3.　キャッシュのクリア

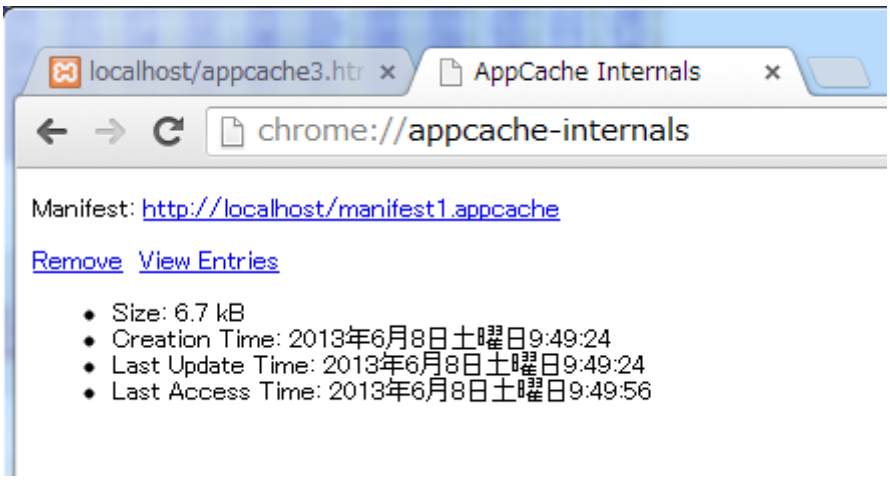
　ブラウザに保存されているキャッシュをクリアするにはブラウザごとに以下の方法で行います。

・Chrome の場合

ブラウザのツールバーにある Chrome メニュー　をクリックし、「ツール」－「閲覧履歴を消去」を選択します。

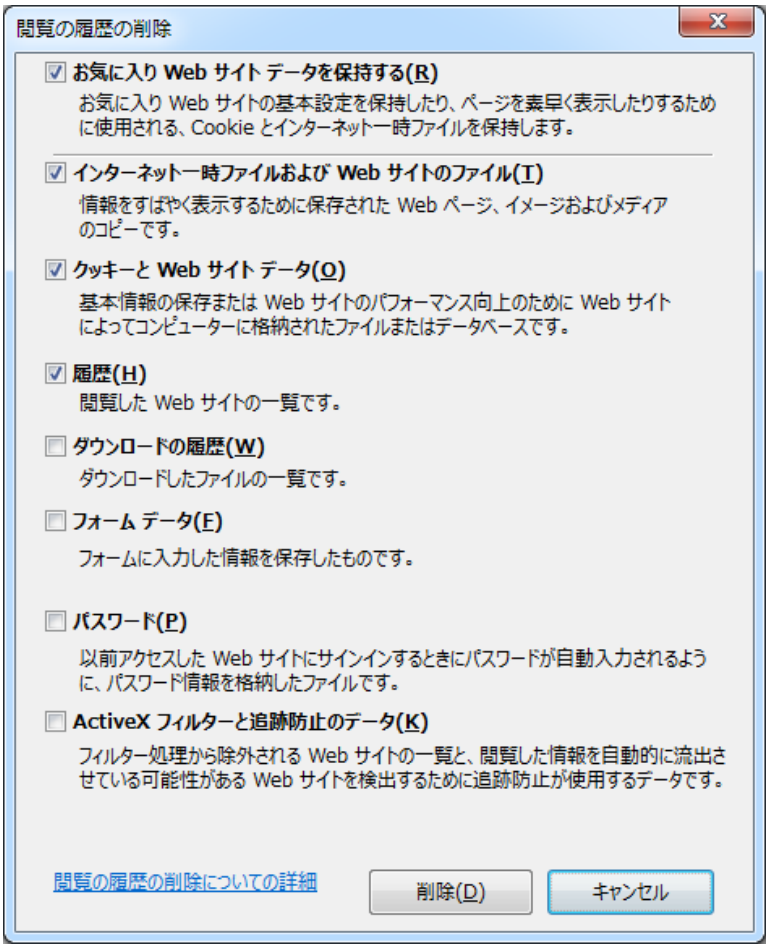
　Chrome の場合、キャッシュマニフェストを読み込ませると、強力にキャッシュを保持してしまい、上の方法でもキャッシュが消えない場合があります。その場合はアドレスバー

から「chrome://appcache-internals/」と入力します。すると現在保持しているキャッシュマニフェストの一覧が表示されるので「Remove」で削除できます。



・Internet Exploler の場合

メニューバーから「ツール」－「閲覧履歴の削除」を選択します。



17 章　WebWorkers によるバックグラウンド処理

基本的に JavaScript で実行される処理は、シングルスレッドで動作するため、処理が終了するまで、ブラウザ上で別の操作をすることができません。「WebWorkers」を使うとマルチスレッドでの動作が可能になりますので、時間のかかる処理をバックグラウンドで実行させ、その間に Web ページをユーザーが操作できるようにすることが可能です。

バックグラウンドで処理する内容を JavaScript の別ファイル（.js）で作成し、Worker オブジェクトに指定することでバックグラウンド処理を行います。

この章の例は Apatche サーバーを稼働状態にして、「localhost」が使用できる状態で実行します。ファイルは「http://localhost」に置き、ブラウザのアドレスバーにたとえば「http://localhost/worker1.html」などを指定して実行します。

17-1　Worker オブジェクト

Worker オブジェクトを使ったバックグラウンド処理の概要を呼び出し側とバックグラウンド処理側についてそれぞれ説明します。

1.　呼び出し側の処理

バックグラウンドで処理する JavaScript ファイルが「jsworker1.js」であったとき、これをバックグラウンドで処理するための Worker オブジェクトを生成するには以下のようにします。

```
worker = new Worker("jsworker1.js");
```

Worker オブジェクトに onmessage イベントハンドラを設定します。引数の event.data でバックグラウンドからのデータを取得することができます。

```
worker.onmessage = function(event){
    // event.data でバックグラウンドからのデータを取得
}
```

バックグラウンド処理側にデータを送るには postMessage メソッドを使います。

```
worker.postMessage("データ");
```

2.　バックグラウンド処理側の処理

onmessage イベントハンドラ内にバックグラウンドで処理する内容を記述します。呼び出し側にデータを送信するには postMessage メソッドを使います。

```
onmessage = function(event){
    // event.data で呼び出し側からのデータを取得
    // postMessage で呼び出し側にデータを送信
}
```

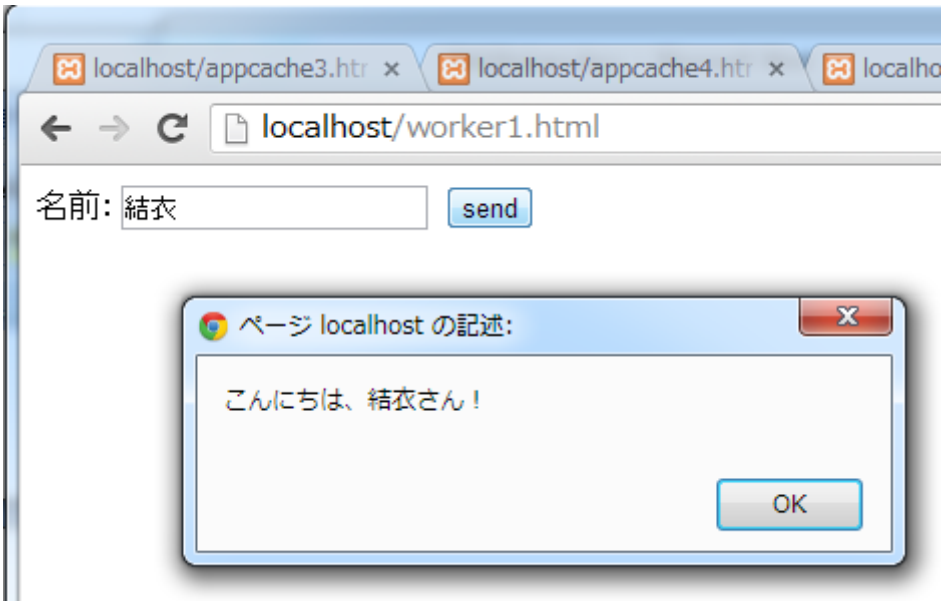
以下の例は呼び出し側からデータをバックグラウンド側に送り、バックグラウンド側では受信したデータを呼び出し側に返すという単純なものです。

・worker1.html

```
<!DOCTYPE html>
<head>
<script type="text/javascript">
	var woker;
	function init(){
		worker = new Worker("jsworker1.js");
		worker.onmessage = function(event){
			alert(event.data);
		}
	}
	function action(){
		var msg= document.getElementById("text1").value;
		worker.postMessage(msg);
	}
</script>
</head>
<body onload="init();">
名前：<input type="text" id="text1">
<input type="button" onclick="action();" value="send">
</body>
</html>
```

・jsworker1.js

```
onmessage = function(event){
	postMessage("こんにちは、" + event.data + "さん！");
}
```



「注」バックグラウンド側のソースを修正してもバックグラウンド側の処理がキャッシュされていて、更新が行われない場合があります。その場合はブラウザの「閲覧履歴を消去する」を選択してください。詳細は「16-1　アプリケーションキャッシュの使い方」－「3 キャッシュのクリア」を参照してください。

18 章　応用アプリ

Web アプリの応用として「リバーシー」と「キー入力練習」を紹介します。
HTML5 の技術を使っているわけではありませんが、パソコンのブラウザでもスマホのブラウザでもどちらのサイズでも丁度よい表示になるように<meta>タグで以下のように viewport を指定しています。

<meta name="viewport" content="width=device-width, initial-scale=1.0,
maximum-scale=1.0, user-scalable=no" />

「注」<meta>タグと viewport
viewport は、ブラウザの仮想的なウィンドウサイズを表す概念のことで、スマートフォンなどの Web ページでよく使われます。最初に iOS の Safari で実装されましたが、現在では Android でも実装されています。viewport を指定することで、画面の縦横長、解像度を気にせず、表示の大きさをコントロールすることができるようになります。viewport は<meta>タグで指定します。

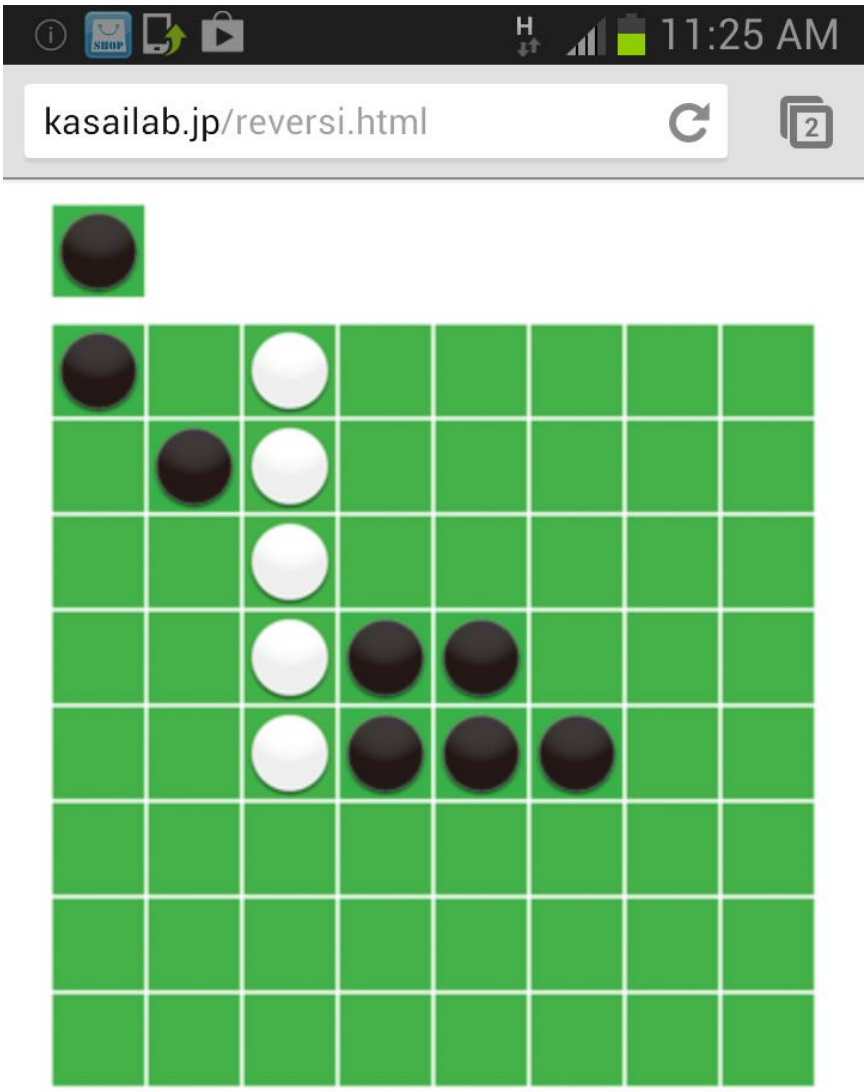
<meta name="viewport" content="プロパティ=値,プロパティ=値,・・・">

content 属性に指定できるプロパティとして以下があります。

content 属性に指定できるプロパティ	機能
width	viewport の横幅（デフォルト:980px)。 device-width を指定するとそのデバイスの横幅が設定される。
height	viewport の縦幅（デフォルト:width とアスペクト比から自動計算）
initial-scale	ページ読み込み時の拡大率で minimum-scale～maximum-scale（デフォルト:1.0)。
user-scalable	ユーザーによる拡大操作の許可（デフォルト:yes)。
minimum-scale	拡大率の最小値で 0.0～10.0（デフォルト:0.25)。
maximum-scale	拡大率の最大値で 0.0～10.0（デフォルト:1.6)。

18-1　リバーシー

黒がクライアント、白がコンピュータで対戦するリバーシーゲームです。黒が先手で、盤面の左上に次の手番の石の色が表示されます。石が置ける位置か確認します。石は自動反転されます。コンピュータの手は黒を打ってから 3 秒後に置かれます。



1.　盤面を作る（盤面初期化関数 init）

8×8 のマスに 3 種類のイメージを配置します。配置するイメージには img0～img63 の通し番号を割り振ります。イメージのサイズは 40×40 ピクセルとします。

green.jpg　　緑（石を置いていない状態）のイメージファイル
white.jpg　　白を置いた状態のイメージファイル
black.jpg　　黒を置いた状態のイメージファイル

18-2　キー入力練習

表示されている日本語テキストをソフトキーからローマ字入力します。正しく入力されると紫色で確定文字が表示されます。ジャンル選択ボタンで、「一般」,「歴史」,「文学」,「四文字熟語」の 4 ジャンルが選択できます。各ジャンルの問題は 10 問固定です。ジャンル名の横の「1/65」のような表示は「現在の正解数/経過秒」です。「⇒」ボタンで次の問題に進みます。

1.　キー入力練習用テーブル

キー入力練習で使用するテーブル（配列）は以下の 3 つです。text[]は日本語のテキストです。read[]は日本語の読みを示す英大文字です。flag[]はキーから入力された文字を元に確定文字を表示するために使用するフラグです。

```
var text=new Array("定期総会",・・・
var read=new Array("TEIKISOUKAI",・・・
var flag=new Array("00.0.00.00.", ・・・
```

2.　読みの規則

ローマ字入力する英字が 1 通りだけの標準的な例として「定期総会」の read と flag の値は以下のようになります。たとえば「定」は「TEI」と入力し、「I」の入力で文字を確定しますので、通常の英字のフラグは「0」とし、確定位置は「.」とします。