

HTMLオブジェクト

- HTMLタグをドリトル内で直接扱うことができるオブジェクトです。
- 角括弧 (<>) を使って直接記述することでHTMLオブジェクトを作成できます。
- HTMLオブジェクトを使うことで、動的な画面更新が可能な双方向コンテンツを簡潔に記述できます。
- WebAPIオブジェクトと組み合わせることで、サーバとの通信を行う双方向コンテンツを作成できます。

HTMLオブジェクトの基本

- HTMLタグは角括弧 (<>) で囲んで作成します。
 - (例) 順序なしリスト(ul)タグを作成します。

```
メッセージボード<ul></ul>
```

- (例) 入力欄(input)タグを作成します。

```
入力欄<input>
```

- (例) ボタン(button)タグを作成します。

```
送信ボタン<button>送信</button>
```

- HTMLオブジェクトには、それぞれの要素に応じた様々なメソッドが用意されています。

タグの属性設定

- HTMLタグには、属性を直接指定することができます。
 - (例) 右寄せを指定したpタグを作成します。

```
段落<p align="right">右寄せの文章</p>
```

- (例) 枠線を持つテーブルを作成します。

```
表<table border="1"></table>
```

- (例) idを指定したpタグを作成します。

```
段落<p id="id1">IDを持つ段落</p>
```

- (例) typeを指定したリストを作成します。

```
リスト<ul type="circle"></ul>
```

- (例) styleを指定して直接スタイルを設定します。

```
掲示板<ul style="border:solid;"></ul>
```

- (例) 入力フィールドのタイプを指定します。

```
数値入力>
```

- (例) 日付入力フィールドを作成します。

```
予約日>
```

- (例) チェックボックスを作成します。

```
チェックボックス>
```

- (例) スライダーを作成します。

```
スライダー
```

- (例) プレースホルダーを指定します。

```
検索欄
```

- (例) 画像に代替テキスト(alt)を指定します。

```
画像![きつねのアイコン](画像ファイルのURL)>
```

- (例) クラス名を指定します。

```
投票数

</div>


```

- (例) 複数のクラス名を指定します。

```
ピクセル

</div>


```

共通のメソッドと属性

- **テキスト**: タグの内容のテキストを設定します。
 - (例) リスト項目(li)タグにテキストを設定します。

```
メッセージ<li></li><<"こんにちは") テキスト。
```

- **いれる**: 子要素としてHTMLタグを追加します。
 - (例) ulタグにliタグを追加します。

```
メッセージボード<ul></ul>
```

```
メッセージ<li></li><<"こんにちは") テキスト。
```

```
メッセージボード! (メッセージ) いれる。
```

- **クリア**: 内容をクリアします。
 - (例) ulタグの内容をクリアします。

```
メッセージボード! クリア。
```

- **innerHTML**: HTMLタグの内部コンテンツを設定 取得します。テキストだけでなくHTMLタグを含む内容も設定できます。

- (例) `innerHTML`属性に文字列を設定します。

```
商品名表示<<h2></h2>
商品名表示:innerHTML<<"コーヒー">
```

- (例) `innerHTML`属性を空にして内容をクリアします。

```
メッセージ表示<<div></div>
メッセージ表示:innerHTML<<">
```

- **appendTo**: 自分自身を引数で指定したHTMLタグの子要素として追加します。
 - (例) `li`タグを `ul`タグに追加します。

```
掲示板<<ul></ul>
ポスト<<li></li>
ポスト:テキスト = "新しい投稿"。
ポスト! (掲示板.appendTo)
```

- **append**: 引数で指定したHTMLタグを子要素として追加します。複数の要素を一度に追加することもできます。
 - (例) `div`タグに複数の要素を追加します。

```
コンテナ<<div></div>
画像<<img>
タイトル<<h2>タイトル</h2>
説明<<p>説明文</p>
区切り = <hr>
コンテナ! (画像) (タイトル) (説明) (区切り) append
```

- **style**: CSSスタイルを設定します。
 - (例) テキストの色を設定します。

```
メッセージ<<span></span>
メッセージ:style<<"color:orange;">
```

- (例) 要素の表示 非表示を切り替えます。

```
ページ1<<div>ページ1の内容</div>
ページ2<<div style="display: none;">ページ2の内容</div>
ボタン:動作 = 「
ページ1:style:display = "none"
ページ2:style:display = "block"

```

- **element**: HTMLタグの実際の要素にアクセスするためのプロパティです。特にスタイルの詳細な設定に使用します。
 - (例) 背景色を設定します。

```
ピクセル<<div class="pixel"></div>
ピクセル:element:style:backgroundColor<<"red">
```

- **remove**: 要素を削除します。
 - (例) 要素を削除します。

要素 `<div>`削除される要素 `</div>`

要素 `remove`

- **src** : `img`要素や`iframe`要素のURLを設定します。
 - (例) `img`要素に画像URLを設定します。

画像 ``

画像: `src="https://example.com/image.jpg"`

- (例) `iframe`要素にURLを設定します。

フレーム `<iframe></iframe>`

フレーム: `src="https://php.dolittle.cc/"`

- **checked** : チェックボックスの状態を取得します。
 - (例) チェックボックスの状態を取得します。

チェックボックス `<input type="checkbox">`

結果 `<p></p>`

チェックボックス: 動作 = 「

結果: テキスト = (チェックボックス `checked != undefined`)

〇〇

idセクタによる要素の取得

- `#id`形式を使ってHTML要素を取得できます。
 - (例) `id`を指定して要素を取得します。

```
<span id="output" style="background-color: lightGreen"></span>
```

出力欄 `#output`

出力欄: テキスト = "ドリトルを使ってテキストを設定"。

フォーム要素のメソッドと属性

- **値?** : フォーム要素の値を取得します (`input`, `textarea`など)。
 - (例) 入力欄の値を取得します。

入力欄 `<input>`

ラベル! (入力欄! 値?) 作る。

- **値** : フォーム要素の値を設定します。
 - (例) 入力欄に値を設定します。

入力欄 `<input>`

入力欄! ("こんにちは") 値。

- **value** : フォーム要素の値を取得 設定するための属性です。
 - (例) 入力欄の値を取得します。

入力欄 `<input>`

```
メッセージ = 入力欄: value
```

- (例) 入力欄の値を設定します。

```
入力欄 <input>  
入力欄: value "こんにちは".
```

- (例) 日付入力欄の値を取得します。

```
予約日 <input type="date">  
api: date = 予約日: value
```

- (例) スライダーの値を取得します。

```
スライダー <input type="range">  
結果 <p></p>  
結果: テキスト = スライダー: value
```

イベント処理

- HTMLオブジェクトはイベント処理も行えます。例えば、ボタntagには動作メソッドを定義することでクリック時の処理を記述できます。
 - (例) ボタンがクリックされたときの処理を定義します。

```
送信ボタン <button>送信</button>  
送信ボタン: 動作 = 「  
ラベル! "ボタンが押されました" 作る。  
」
```

- (例) divタグがクリックされたときの処理を定義します。

```
ピクセル <div class="pixel"></div>  
ピクセル: 動作 = 「  
ピクセル: element: style: backgroundColor="red"  
」
```

- (例) スライダーの値が変更されたときの処理を定義します。

```
スライダー <input type="range">  
結果 <p></p>  
スライダー: 動作 = 「  
結果: テキスト = スライダー: value  
」
```

条件分岐による表示切り替え

- 条件分岐を使って、表示するHTMLタグを動的に切り替えることができます。
 - (例) 貸出状況に応じてボタンの表示を切り替えます。

```
貸出状況表示: 作る = <span id title author status>  
sp=<span></span>
```

```
sp<テキスト<title+"-"+author+">>
<status == "available">> なら「
  Button=<button>借りる</button>
  Button:動作=「
// ボタンを押したときの処理
<>
」そうでなければ「
  <span>貸出済み</span>
」実行。
<br>
自分。
<>
```

- (例) 予約状況に応じてボタンの表示を切り替えます。

```
空き状況表示: 作る = <><date time room status>
sp=<span></span>
sp<テキスト<room+"-"+time+">>
<status == "空き">>! なら「
  Button=<button>予約</button>
  Button:動作=「
// 予約処理
<>
」そうでなければ「
  <span>予約済み</span>
」実行。
<br>
自分。
<>
```

アラートを表示する

- alert メソッドを使用して、アラートダイアログを表示できます。
 - (例) ボタンをクリックしてアラートを表示します。

```
ボタン<<button>クリック</button>
ボタン:動作=「
!("あいうえお") alert
<>
```

WebAPIとの連携

- HTMLオブジェクトはWebAPIオブジェクトと組み合わせることで、サーバとの通信を行う双方向コンテンツを作成できます。
- 以下は、簡単なチャットプログラムの例です。

```
メッセージボード<<ul></ul>
入力欄<<input>
送信ボタン<<button>送信</button>
```

```
受信_wa [] webapi!作る。  
受信_wa.url [] "read.php"[]
```

```
メッセージ表示 = 「 | メッセージ配列 |  
メッセージボード!クリア。  
メッセージ配列! 「 | 内容 |  
メッセージ<li></li>[] [] 内容) テキスト。  
メッセージボード! (メッセージ) 入れる。  
」それぞれ実行。  
[]
```

メッセージ表示! (受信_wa[]読む) 実行。

```
送信_wa [] webapi!作る。  
送信_wa.url [] "write.php"[]  
名前 = "ユーザー"。
```

```
送信ボタン: 動作 = 「  
送信_wa.内容 = 名前 + " : " + (入力欄!値?)。  
送信_wa[]読む。  
メッセージ表示! (受信_wa[]読む) 実行。  
[]
```

- このプログラムでは、サーバ側で動作するPHPプログラム[]read.phpとwrite.php[]とクライアント側で動作するドリトルプログラムを組み合わせています。
- サーバ側のPHPプログラムは以下のようになります。

snippet.php

```
//掲示板にメッセージを書き込むwrite.php  
<?php  
  $message = $_GET['message'] . "\n";  
  file_put_contents("chat.txt", $message, FILE_APPEND);  
?>  
  
//掲示板の書き込み一覧を取得するread.php  
<?php  
  $message = file("chat.txt");  
  echo json_encode($message);  
?>
```

よく使われるHTMLタグ

- **ul, li**: 順序なしリストとリスト項目を表示します。
 - (例) 順序なしリストを作成します。

```
リスト<ul></ul>[]  
項目1<li></li>[] [] "項目1") テキスト。  
項目2<li></li>[] [] "項目2") テキスト。
```

リスト！（項目1）いれる（項目2）いれる。

- **ol, li**: 順序ありリストとリスト項目を表示します。
 - （例）順序ありリストを作成します。

```
リスト<ol></ol>
項目1<li></li> "項目1" テキスト。
項目2<li></li> "項目2" テキスト。
リスト！（項目1）いれる（項目2）いれる。
```

- **input**: 入力欄を表示します。
 - （例）入力欄を作成します。

入力欄<input>

- （例）プレースホルダーを指定した入力欄を作成します。

検索入力欄<input placeholder="検索ワード" id="search-input" />

- （例）数値入力欄を作成します。

数量<input type="number" value="1"/>

- （例）日付入力欄を作成します。

予約日<input type="date">

- **datalist, option**: 入力欄の選択肢リストを作成します。
 - （例）スライダーと選択肢リストを作成します。

```
スライダー<input type="range" list="数値" />。
<datalist id="数値">
  <option value="0"></option>
  <option value="10"></option>
  <option value="20"></option>
  <option value="30"></option>
  <option value="40"></option>
  <option value="50"></option>
</datalist>
```

- **textarea**: 複数行の入力欄を表示します。
 - （例）テキストエリアを作成します。

内容<textarea></textarea>

- **button**: ボタンを表示します。
 - （例）ボタンを作成します。

ボタン<button>クリック</button>

ボタン：動作 = 「ラベル！"ボタンがクリックされました" 作る」。

- **iframe**: 外部ページをフレーム内に表示します。
 - （例）iframeタグでWebページを表示します。


```
<style>
iframe { width: 100%; height: 95%; }
</style>
フレーム<iframe></iframe>
フレーム:src<"https://php.dolittle.cc/">
削除ボタン<button>消す</button>
削除ボタン:動作 = 「
フレームremove

```

- **div**: 汎用的なブロック要素を作成します。
 - (例) divタグを作成します。

```
コンテナ<div></div>
テキスト<p></p> "こんにちは" テキスト。
コンテナ! (テキスト) 入れる。
```

- **span**: 汎用的なインライン要素を作成します。
 - (例) spanタグを作成します。

```
テキスト<span></span> "強調" テキスト。
```

- **p**: 段落を表示します。
 - (例) 段落を作成します。

```
段落<p></p> "これは段落です" テキスト。
```

- **h1, h2, h3, ...**: 見出しを表示します。
 - (例) 見出しを作成します。

```
タイトル<h1>大阪電気通信大学</h1>
見出し = <h2>学部</h2>
```

- **img**: 画像を表示します。
 - (例) 画像を表示します。

```
画像
```

- (例) 代替テキストとクラスを指定した画像を表示します。

```
画像
```

- **figure**: 画像や図表などを囲むコンテナとして使用します。
 - (例) 画像をfigureタグで囲みます。

```
図<figure>

</figure>
```

- **table, tr, th, td**: 表を作成します。
 - (例) 2行2列の表を作成します。

```
表<<table border="1">
<tr>
<th>工学部</th>
<th>情報工学部</th>
</tr>
<tr>
<td>電気電子工学科</td>
<td>情報工学科</td>
</tr>
</table>
```

- **a**: リンクを作成します。
 - (例) リンクを作成します。

```
リンク<<a href="https://www.city.neyagawa.osaka.jp/index.html">寝屋川市</a>
```

- **br**: 改行を作成します。
 - (例) 改行を挿入します。

```
<span>あいことば</span>
あいことば<input>
<br>
<span>名前</span>
```

- **hr**: 水平線を作成します。
 - (例) 水平線を挿入します。

```
<h1>ショッピングサイト</h1>
<hr/>
```

- **label**: フォーム要素のラベルを作成します。
 - (例) ラベルを作成します。

```
<label>ユーザ名:</label>
ユーザ名=<input type="text"/>
```

スタイル設定

- **style**要素を使うことでHTMLタグのスタイルをまとめて設定できます。
 - (例) スタイルを設定します。

```
<style>
body{margin: 2%;}
.card-image{width:100px;}
.card-name{font-size: 30px;font-weight: bold;}
.card-price{font-size: 25px;}
.card-desc{color: grey;}
</style>
```

- (例) ドット絵用のスタイルを設定します。

```
<style>
.pixel{width:6px;height:6px;}
.line{display:flex;}
.canvas{border:solid;}
</style>
```

- (例) ポップアップウィンドウ用のスタイルを設定します。

```
<style>
.popup-container {
  position: absolute;
  top: 50%;
  left: 50%;
  border: 1px solid;
  transform: translate(-50%, -50%);
}
</style>
```

HTMLオブジェクトの利点

- HTMLタグをプログラム内に直接記述可能なため、動的な画面更新を簡潔に記述できます。
- WebAPIによるサーバ側とクライアント側で処理を分離することで、各処理の責任範囲が明確になり、可読性が高くなります。
- HTMLとプログラムが混在しないため、従来のCGI方式と比較して処理の流れが理解しやすくなります。

実践的な例：メッセージボード

- 以下は、ボタンをクリックするとメッセージを追加表示するシンプルなメッセージボードの例です。

```
メッセージボード<ul></ul>
入力欄<input>
送信ボタン<button>追加</button>

送信ボタン：動作 = 「
メッセージ<li></li><<input>入力欄！値？）テキスト。
メッセージボード！（メッセージ）いれる。
入力欄！（”）値。
<<
```

実践的な例：ショッピングサイト

- 以下は、商品を表示するシンプルなショッピングサイトの例です。

```
<style> .card-image{width:100px;} </style>
```

```
<h1>ショッピングサイト</h1>[]
<hr>[]
container = <div></div>[]
商品表示 = 「 |商品名 価格 画像URL |
商品画像=<img class="card-image">[]
商品画像:src=画像URL[]
商品名表示=<h2></h2>[]
商品名表示:innerHTML=商品名。
価格表示=<p></p>[]
価格表示:innerHTML="" + 価格。
container!(商品画像)(商品名表示)(価格表示)(<hr>)append[]
[]
商品表示!"コーヒ" 130 "https://example.com/coffee.jpg" 実行。
```

実践的な例：チャットアプリケーション

- 以下は、合言葉を使ったグループチャットの例です。

```
<span>あいことば</span>[]
あいことば<input>[]
<br>[]
<span>名前</span>[]
名前<input>[]
<br>[]
<span>書き込み内容</span>[]
書き込み内容<input>[]
<br>[]
送信ボタン<button>送信</button>[]
掲示板<ul style="border:solid;"></ul>[]

送信_wa [] webapi!作る。
送信_wa:url = "chat_send.php"[]
送信ボタン:動作 = 「
送信_wa:message["+(あいことば:value)+"][]
送信_wa:message[]送信_wa:message+(名前:value)[]"[]
送信_wa:message[]送信_wa:message+(書き込み内容:value)[]
送信_wa!読む。
[]

受信_wa [] webapi!作る。
受信_wa:url = "chat_read.php"[]
タイマー!作る 3 間隔 600 時間「
受信_wa:あいことば = あいことば:value[]
掲示板:innerHTML[]"[]
書き込み一覧 = 受信_wa[]読む。
書き込み一覧:data![]メッセージ |
  ポスト<li></li>[]
  ポスト:テキスト = メッセージ。
  ポスト!(掲示板[]appendTo[]
  」それぞれ実行。
```

」実行。

実践的な例：投票サイト

- 以下は、投票機能を持つシンプルな投票サイトの例です。

```
<style>
  .pet-icon{width:100px;}
  .vote-count{font-weight:bold;}
</style>
<h1>あなたはどっち派？</h1>


<h2>きつね派</h2>
きつねButton=<button>投票する</button>
きつね票数<div class="vote-count" id="fox-votes"></div>


<h2>たぬき派</h2>
たぬきButton=<button>投票する</button>
たぬき票数<div class="vote-count" id="raccoondog-votes"></div>

api = webapi!作る。
api:url = "vote.phpのURL"

結果表示 = 「」。
結果表示：作る = fox raccoondog
:きつね票数：テキスト fox+「票」。
:たぬき票数：テキスト raccoondog+「票」。

```

```
きつねButton:動作=「
  :api:vote="fox"
  r = :api!読む。
  r:results | 結果|
結果表示！（結果!1 読む）（結果!2 読む）作る。
」それぞれ実行。

```

```
たぬきButton:動作=「
  :api:vote="raccoondog"
  r = :api!読む。
  r:results | 結果|
結果表示！（結果!1 読む）（結果!2 読む）作る。
」それぞれ実行。

```

```
// 初期状態の表示
r = api!読む。
r:results | 結果|
結果表示！（結果!1 読む）（結果!2 読む）作る。
```

」それぞれ実行。

実践的な例：ドット絵エディタ

- 以下は、シンプルなドット絵エディタの例です。

```
<style>
.pixel{width:6px;height:6px;}
.line{display:flex;}
.canvas{border:solid;}
</style>□

// WebAPIの設定
write_wa = webapi!作る。
write_wa:url = "pixelArt_set.phpのURL"□

// 色変更とサーバー更新処理
write = □|x y color|
  write_wa:x=x□
  write_wa:y=y□
  write_wa:c=color□
  write_wa!読む。
□□

// キャンバスの作成
canvas = <div class=canvas></div>□
canvas_ary = 配列!作る。
□|y|
  line = <div class=line></div>□
  line_ary = 配列!作る。
  □|x;pixel|
    pixel = <div class=pixel></div>□
    pixel□動作 = 「
      pixel:element:style:backgroundColor="red"□
      write!(x)(y)("red")実行。
    □□
    pixel!(line)appendTo□
    line_ary!(pixel)書く。
  」!100 繰り返す。
  line!(canvas)appendTo.
  canvas_ary!(line_ary)書く。
」!100 繰り返す。

// サーバーからの状態取得処理
read_wa = webapi!作る。
read_wa:url = "pixelArt_get.phpのURL"□
タイマー!作る 3 間隔 600 時間「
  canvas_sv = read_wa!読む。
  canvas_sv□□□line y□
  line□□□color x; pixel□
  pixel = canvas_ary!(y)読む (x)読む。
```

```

pixel:element:style:backgroundColor=color[]
」それぞれ実行。
」それぞれ実行。
」実行。

```

実践的な例：予約サイト

- 以下は、日付を指定して予約できるシンプルな予約サイトの例です。

```

<h1>予約サイト</h1>[]
result = <div></div>[]

reserve = webapi!作る。
reserve:url = "reserve.phpのURL"[]
api = webapi!作る。
api:url = "check_availability.phpのURL"[]

予約日[]<input type="date">[]
空き検索[]<button>検索</button>[]

空き検索:動作=「
  api:date = 予約日:value[]
  r = api!読む。
  r:results[]|結果|
  空き状況表示!(結果!1 読む) (結果!2 読む) (結果!3 読む) (結果!4 読む) 作る。
  」それぞれ実行。
  []
  <br>[]

空き状況表示 = 「」。
空き状況表示:作る = []date time room status[]
  sp=<span></span>[]
  sp[]テキスト[]room+"-"+time+"[]"[]
  []status == "空き"! なら「
  Button=<button>予約</button>[]
  Button:動作=「
    reserve:date = date[]
    reserve:time = time[]
    reserve:room = room[]
    r = reserve!読む。
    []r == "予約が完了しました"! なら「
      result:テキスト[]date+" "+time+" "+room+"の予約が完了しました"。
    」実行。
    []
  」そうでなければ「
    <span>予約済み</span>[]
  」実行。
  <br>[]
自分。
[]

```

実践的な例：ページ切り替え

- 以下は、ボタンをクリックして画面を切り替えるシンプルなページ切り替えの例です。

```
page1 = <div>
<h1>page1</h1>
<p>こんにちは。これはページ 1 です</p>
<button id="toPage2">ページ2へ</button>
</div>

page2 = <div style="display: none;">
<h1>page2</h1>
<p>こんばんは。これはページ 2 です</p>
<button id="toPage1">ページ1へ</button>
</div>

#toPage2:動作 = 「
  page1:style:display = "none"
  page2:style:display = "block"
」

#toPage1:動作 = 「
  page2:style:display = "none"
  page1:style:display = "block"
」
```

実践的な例：ポップアップウィンドウ

- 以下は、ボタンをクリックしてポップアップウィンドウを表示する例です。

```
<style>
.popup-container {
  position: absolute;
  top: 50%;
  left: 50%;
  border: 1px solid;
  transform: translate(-50%, -50%);
}
</style>

popupButton = <button>popup</button>
popupButton:動作 = |container|
  container = <div class="popup-container"></div>
  title = <h3>ポップアップウィンドウ</h3>
  closeButton = <button>閉じる</button>
  closeButton:動作 = 「
    container! remove
  」
  container!(title) (closeButton) append
```




From:

<https://dolittle.eplang.jp/> - プログラミング言語「ドリトル」

Permanent link:

https://dolittle.eplang.jp/ref_html_js?rev=1741330426

Last update: **2025/03/07 15:53**

