# Webページ

## MVC

Model, View, Control に分けた Web ページの開発手法。

View ユーザーが目にする出力の部分

Control クライアントからのリクエストに対応する。htmlやJavaScriptなどが対応。

Mode データベースのテーブルなどに該当。ロジックなどを処理。

## 基本

### フォルダ構成

┬ css/

│ ├ style.css # MVCのV。文字や背景色を担う。

│ # style.cssは最上位ディレクトリ配置される事も多い。

├ images/

│ ├ favicon.ico # ページタイトルの左に表示されるアイコン

│ │ # 　必須ではないが，集客などの理由で使用を強く推奨されている。

│ ├ main.jpg # webページで使いたい画像

├ js/

│ ├ script.js #

├ page3/ # 規模の大きなページだと，サブページ用ディレクトリがある事も。

│ └ index.html #

├ index.html # 最初にアクセスされるページ。慣例的にindexという名前が多い。

├ page2.html # ２つ目（以降）のページ。名前はなんでもよい。

### Pathの書き方

|  |  |  |
| --- | --- | --- |
| 対象 | 書き方 | 例／備考 |
| カレントディレクトリ | ./ | ./page2.html  この場合 page2.htmlだけでも良い。 |
| 一階層上 | ..( ドット  ２つ) | ../css/styles.css” |
|  |  |  |

### html［基本］

<!doctype html> <!-- 大文字（DOCTYPE）でも構わない -->

<html lang="ja"> <!-- ページの言語を指定 “en” なら英語-->

<head>

<meta charset="UTF-8">

<title>タイトル</title>

</head>

<body>

<!-- ここにページ本体を記述 -->

</body>

</html>

### ヘッダ［基本］

<head>

<title>ページのタイトル</title>

<!-- スタイルシートの指定 -->

<link rel="stylesheet" href="assets/css/style.css" />

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/.../font-awesome.min.css" />

</head>

*※ヘッダ関連の情報は別ページにまとめてある：*[*ヘッダ*](#_ヘッダ)

### CSS［基本］

この方法が一番多い。

index.html：

<head>

<link rel="stylesheet" href="./styles/styles.css">

<title>ページタイトル</title>

</head>

<body>

　　　<!-- ページ内容 -->

./styles/styles.css：

p {

color: #212121; /\* 最低１つはルールを記述しなければならない \*/

}

（参考）[CSS/基本](#_基本)

### スクリプト［基本］

<body>

<!-- JavaScript-->

<script>

/\* 直接JavaScriptのコードを記述する場合 \*/

</script>

<!-- 別ファイルを指定する場合 -->

<script type='text/javascript' src='assets/js/script.js' defer></script>

<script src="js/scripts.js"></script> <!-- html5ではtype= を省略できる -->

<?php

/\* php ここにPHPの処理を記述 \*/

$text = "hoge";

echo "<p> $text </p>";

?>

</body>

### 用語：基本

|  |  |  |
| --- | --- | --- |
| 単語 | 内容 | 例 |
| タグ | <>で区切られた範囲の文字列。htmlの核となる部分。 | **<div>** とか **<p>** など |
| 要素 | htmlの開始タグから終了タグまでの１つの塊を示す。 | **<p>段落１</p>**  ※これが１つの要素。 |
| 属性 | タグ内部に書かれる副次的な機能を与える文字列。 | <div **class**=”menu”>　→ class属性  <a **href**=”./\*\*\*”>　→ href属性など |
| プロパティ | CSSで使われる : の後に記述される属性の値。  属性の事をプロパティと呼ぶこともある |  |
| セレクタ | CSSで使われる，デザインを適用するタグやクラスを表す部分。 | **h1** { font-size: 20px; } |

## レンダリング

Parse→Style → Layout → Paint → Composite

Parse DOMツリーとStyleルールズの解析

Style DOMツリーとStyleルールズの紐付け

Layout

Paint

Composite

## 配色

|  |  |  |
| --- | --- | --- |
|  | 割合 |  |
| ベース | 70% | 全体の70%にする |
| メイン | 25% |  |
| アクセント | 5% | ユーザーの目を引く |

色による印象の違い

エネルギッシュ、強いインパクト、興奮、怒り、心拍数を高める色、情熱

信頼と安全の色、安心感、知的、誠実、理性、清潔感

親しみ、元気、楽しい、活発、活動

楽観的、明るい、快活、希望、陽気、若々しさ

健康、リラックス、安らぎ、穏やか

神秘的、高貴、優雅、不安、癒し、宇宙

落ち着き、穏やか、信頼、上品、安定、迷い

神秘、威厳、高級、極限、自信

（参考）<https://www.m-hand.co.jp/design/4002/>

配色選択ツール

<https://www.palettelist.com/>

## 静的ページと動的ページ

|  |  |  |
| --- | --- | --- |
|  | **静的ページ** | **動的ページ** |
| **メリット** | サーバー負荷が少ない  セキュリティ対策が比較的容易 | 最新の情報を表示しやすい。  PHPなど，学習コストがかかる |
| **デメリット** | 情報のリアルタイム性が弱い  データベースと連動したページなどは無理  （ユーザー毎に異なるページを表示、など） | サーバーに負荷をかける。 |

## url

### プロトコル

http:// があまりに有名だが，ここはプロトコルを指定しているだけ，当然他のプロトコルも使用できる。

有名なのはsmb://192.168.1.30，とかftp://192.168.1.30とか。

### ID

http://sample.com/#id\_name でページ内のid=id\_name のところまでジャンプできる。

ページ内のid は開発ツールなどで調べる事ができる。

### クエリ文字列(query string)

url内に含まれるパラーメータの事。例）?lang=eng&user=0401

当然ながら値が丸見えになるため、セキュリティには考慮が必要。

# 記述法まとめ

## コメントアウト

|  |  |  |  |
| --- | --- | --- | --- |
|  | 1行 | 複数行 | 備考 |
| html | <!-- コメント --> | |  |
| CSS | /\* コメント\*/ | |  |
| Javascript | //コメント | /\*コメント\*/ |  |
| PHP | //コメント  #コメント | /\*コメント\*/ |  |

## 動的レスポンス系

### ホバーによるtool-tip表示

html - onmouseover属性

<span onmouseover = "show('menu');" onmouseout = "show('menu');">

jQuery – mouseover

$('h1').mouseover(function() {

$(this).text('マウスオーバーを実行しました');

});

bootstrap - Tooltips

<https://getbootstrap.jp/docs/4.4/components/tooltips/>

html - data-tooltip属性 ⇒動作しなかったが？

# html

## タグ

### 共通の属性：全てのタグ

属性は「プロパティ」とも呼ばれている

name 対象オブジェクトの名前。例えば，JavaScriptなどが要素へアクセスする際に使用する。

id id\_nameでジャンプできるようになる。

### 基本中の基本

#### 目次：基本

|  |  |  |
| --- | --- | --- |
| タグ | 概要 | 詳細／備考 |
| <head> | ヘッダー | １つのhtmlの中で１回だけ使用。  html5以降はbody内に<header> タグを使う事が多い。 |
| <body> | ページ本文 |  |
| <p> | 段落 | ブロック要素。 |
| <h1> | 見出し | ブロック要素。h1～h5まで |
| <a> | リンク。anchor を指す。 | ブロック要素。 |
| <div> | ブロック要素の範囲 | 自動的に改行される。 |
| <span> | インライン要素の範囲 | 改行されず，同じ行内で処理される。 |
|  |  | html5かららしい |
| <br> | 改行 |  |
| <img> |  | インライン要素。 |

（基本タグのみ）<https://gray-code.com/html_css/a-element/>

インライン要素 ：自動的に改行される。

ブロック要素 ：改行されず，同じ行内で処理される。

#### <a>：アンカーリンク

属性

href

具体例

<a href="./about.html">店舗情報</a> <!--別ページへのリンク。つまりページ移動。 -->

<a href="#message">メッセージ</a> <!--自ページ内リンク 但し id=”message”の要素が必要 -->

<a href="target\_example.html" target="\_blank">

download

ブラウザでダウンロード用のウインドウ画面を表示してダウンロード。

注意点として，hrefで指定した元ファイルの名前と，ダウンロード後の名前は別にできる。

具体例

<a href="pdf/original.pdf" download="new\_fn.pdf">ダウンロード</a>

<a href="download/acme-doc-2.0.1.txt" target="\_blank">ダウンロード</a> <!-- targetとの併用 -->

download属性が無い場合

pdfや画像 ：そのまま開く。（他にも.docなど）

それ以外 ：hrefに指定した，元ファイル名のままダウンロード。

Chrome，最新Opera，Chromium/Blink　では属性値を無視する，Firefoxでは同一ドメインのファイルダウンロードしかできない。など，ブラウザによって挙動が異なる。

Content-Type: application/force-download　詳細不明：[情報元](https://mamewaza.com/support/blog/force-download.html)

onclick

クリックしたときの処理を、基本的にはjavaScriptで記述する。

target

\_self

\_blank タブブラウザであれば、新しいタブを開いて展開。新しいブラウジングコンテキスト。

\_parent

\_top

rel 現在のページとの関係を指定する。

stylesheet CSSの指定。最もよく使われるrelの用途。

icon アイコンを指定。いわゆるファビコンの事。

next ページに分かれた記事の際，リンク先が次のページである。 (syn) prev

canonical 正規のurlを示す。SEO対策で，urlを分散させない為に使う。

#### <div>：ブロック範囲

属性

align “center”

#### <link>：ハイパーリンク

アンカー（aタグ）との違い

ハイパーリンク： ホームページに限らず、別の場所にジャンプするためのリンク

アンカーリンク： ホームページに埋め込まれたリンク

属性

### フォーム系

#### 概要

<form>基本的に一連の<input>タグなどは<form>タグ内に収めるべき。

また、table内部で<form>

required 必須入力

<form action="" method="get">

<label>数値:<input type="number"　name="ival" required></label> <!--必須になる-->

<input type="submit" value="送信">

</form>

action htmlメソッド処理先URLを指定。（例）<form action="https://example.com">

method 実行するhtmlメソッド（後述）を指定する

novalidate ブラウザによるvalidationを無効化する

selected optionなどで，最初に選択されている項目

autofocus ページの読み込みが完了した時に，この要素にフォーカスをする。

fieldset フォームや入力コントロールの内容をまとめる。あまり使われない？

#### HTMLメソッド

|  |  |  |
| --- | --- | --- |
| メソッド | 概要 | 備考 |
| Get | リソースの取得のみ。 |  |
| Post | 新規作成 |  |
| Put | リソースを更新 |  |
| Delete | 削除 |  |
| Send | ファイルなどのバイナリファイルを送る |  |

dispatch

#### タグ目次：フォーム

|  |  |  |
| --- | --- | --- |
| タグ | 概要 | 詳細／備考 |
| form |  |  |
| label |  |  |
| input | ユーザー入力 | ボタンなどもこれで作る。 |
|  |  |  |
|  |  |  |
| select | セレクトボックス | C#で言うコンボボックス。ドロップダウン式。 |
|  |  |  |
|  |  |  |

#### <input>：ユーザー入力

プロパティ

type 入力方式を選択する

<input name="my\_title" type="text">

post, メソッドではnameで各入力値を識別する。

|  |  |  |
| --- | --- | --- |
| type | 概要 | 備考 |
| text |  |  |
| number | 数値入力 | min：最小値 |
| checkbox |  |  |
| submit |  | <buttn type=”submit”>と同等 |
| date |  |  |
| month | 年、月のみ入力 |  |

value

（補足）html5なら<button></button> を使う事もできる。

<button type="submit">ボタン１</button>

autocomplete

具体例：

複数選択

参考：https://developer.mozilla.org/ja/docs/Web/HTML/Element/input

##### type="number"

<http://www.htmq.com/html5/input_type_number.shtml>

##### type="checkbox"

値変化イベント

jQuery

$('#chk\_mode').change(()=> {

if($('#chk\_mode').prop('checked')){

console.log('checked');

}

else{

console.log('not checked');

}

})

##### プロパティ（属性）

value 画面表示時点での初期値を指定する

pattern正規表現で入力文字を指定できるらしい

例：

<!--　数値 -->

<input type="text" name="sample" pattern="^[0-9]+$">

<!-- ひらがな -->

<input type="text" name="sample" pattern="(?=.\*?[\u30A1-\u30FC])[\u30A1-\u30FC\s]\*">

情報元：<https://arts-factory.net/pattern/>

#### <select>：セレクトボックス

属性

multiple 複数選択可能になる

<select id="contact" name="contact" multiple>

具体例：

単一選択

<label for="contact">ご希望の連絡方法：</label>

<select id="contact" name="contact" size="5">

<option value="email" selected>電子メール</option>

<option value="tel" selected>電話</option>

<option value="fax" selected>ファックス</option>

<option value="postal" selected>郵便</option>

<option value="office">ご来店時に係員より説明</option>

</select>

#### <button>

##### 概要

<button type="button">ボタンテキスト</button>

##### type

"submit" inputタグの入力内容を適用してPOSTで投げる？（**デフォルト**）

"button"

"reset" おせっかい機能になる事が多いらしい。あまり使われない。

注意：**初期値はsubmit**なので、不正な値や空白の場合意図しないpostなどが発生する点に注意。

##### onclick

type= "button"の時だけ有効

location.href

<button onclick="location.href='https://example.net/test'">リンク</button>

#### <form>

##### onsubmit

概要

submitボタン（button type=”submit”）のクリックイベントを処理するjavascriptのメソッドを指定する。

onsubmitに指定するメソッドはtrue/falseを返す必要がある。

例：

[ sample.html ]

<script type="text/javascript">

function create\_confirm(){

return window.confirm('よろしいですか？ ');

}

</script>

<form class="form" method="post" onsubmit="return create\_confirm()">

<button type="submit">適用</button>

</form>

編集中のEnter押下によるpostを止めたい場合

ポイント

・formのonsubmit属性の値を "return false;" にする。

・適用ボタンのtypeを "button" にしてonclick属性の値を "submit();" にする。

例：

[ sample.html ]

<form class="form" method="post" onsubmit="return false;">

<button type="button" onclick="submit();">適用</button>

</form>

補足：submit()という関数を自作で用意する必要は無い。

補足：エクセル操作に慣れた人は、Enterで次の入力項目へ移動する事を期待するらしい。

jQueryで書く場合

<script type="text/javascript">

$(document).ready(function() {

$('form').on('keydown', (event)=>{

if(event.key == 'Enter'){

event.preventDefault();

};

})

})

</script>

### その他基本タグ

どこの区分におくかまよっているもの

<pre> 半角スペースや改行をそのまま表示する

<del> 削除されたテキスト

<ins> 追加されたテキスト

<abbr> 略語（全般）

<acronym> 略語（頭字語）

<dfn> 定義する用語

<blockquote> 長文の引用（ブロックレベル）

<q> 短文の引用（インライン）

<cite> 引用元（出典・参照先）

<pre> 整形済みテキスト

<code> ソースコード

<var> 変数・引数

<kbd> 入力する文字

<samp> 出力サンプル

<bdo> 書字方向

draggable

<p draggable="true">TRUE</p>

<p draggable="false">FLASE</p>

### アトリビュート

readonly属性

disabled属性

### Html標準視覚効果

#### 目次：視覚効果

|  |  |  |
| --- | --- | --- |
| タグ | 概要 | 詳細／備考 |
| <em> | 強調 |  |
| <strong> | より強い強調 |  |
| <sup> | 上付き文字 |  |
| <sub> | 下付き文字 |  |
| <table> |  |  |
| <thead> | テーブルヘッダ |  |
| <tbody> | テーブルボディ |  |
| <td> | テーブルデータ |  |
| <th> | テーブルヘッダ | <thead>内で使う<td>と思えばよい。 |
|  |  |  |
|  |  |  |

#### <font>：フォント

属性：size 値：1~7　+1, -3 などの相対サイズも指定可能。

属性：color

属性：face カンマ区切りで複数指定可能。（例）face="メイリオ ,'ＭＳ Ｐゴシック

font size="サイズ" color="文字色" face="フォント種類">

※但し昨今ではcssのfont-size属性などで指定する場合の方が多い。

#### <table>：テーブル

属性：table

border 1px

solid

cellpadding

cellspacing

属性：td

colspan 列幅。列方向の結合数。 関連：rowspan

<td colspan="2"></td>

具体例：

テーブル：パターン１

<table>

<thead>

<tr>

<th>号機</th> <!—**T**able **H**eader の意 -->

</tr>

</thead>

<tbody>

<tr>

<td>１号機</td> <!—**T**able **D**ata の意 -->

</tr>

</tbody>

<tfoot>

<tr>

<td>備考</td><td"></td>

</tr>

</tfoot>

</table>

補足：tfootはあまり使われない

テーブル：パターン２

<table>

<tr>

<th>表題１</th><th>表題２</th><th>表題３</th>

</tr>

<tr>

<td>項目１</td><td>項目２</td><td>項目３</td>

</tr>

</table>

#### <ul>／<ol> リスト

具体例

##### ol番号付きの箇条書き

Oredered List

<ol>

<li>項目1</li>

<li>項目2</li>

<li>項目3</li>

</ol>

##### ul黒ポチの箇条書き

Unordered List

メニューでよく使われる

<ul>

<li>項目1</li>

<li>項目2</li>

<li>項目3</li>

</ul>

li（リストアイテム）の幅はheightで決める

### HTML5

#### HTML5とは

2014年に勧告されたHTMLの大幅な仕様変更で，人間にも読めるコード，メディア再生が可能など様々な機能があり，4.01以前とは別物として扱われる事が多い。

特徴の1つとして構造化タグと呼ばれるものがある。これは <article>などの事で，今までは<div>にidやclass名を付け各々が独自にやっていた構造タグを仕様で統一した。

#### 目次：html5

構造化タグ：

構造化（アウトライン）の分かりやすい例：<https://qiita.com/minami-actindi/items/176ffeeb6d660badab56>

|  |  |  |
| --- | --- | --- |
| タグ | 概要 | 詳細／備考 |
| article | 独立した記事 | それだけで完結する，独立した記事である事を示す。  コメントなど。 |
| section | １つのまとまり | 章・節・項などの１つのまとまりであることを示す  内部に<h1>を持つ事を推奨 |
| aside | 補足情報 | メインコンテンツとは関連性が低い補足  関連リンク，広告など |
| main | メインコンテンツ |  |
| nav | ナビゲーション | メニューやサイトマップのようなものが多い。 |
| header | ヘッダ | つまりhtml5 では<html><body> と，いきなりbodyが書けることになる。 |
| footer | フッタ |  |
|  |  |  |

※IE9より前のバージョンに対応させる場合には、html5shiv.jsを使う

media パソコン以外のメディアである場合に適用される？

**all** 全てのデバイス

#### タグ

##### <section>：１つのまとまり

（比較）article

内部に<h1>を持つ事を推奨

属性

具体例：

<section>

<h1>コンテンツ内容見出し</h1>

<p>コンテンツ内容1</p>

</section>

##### <artcle>：独立した記事

（比較）section

具体例：

<article> <!-- よくsectionと併用して入れ子にする -->

<section>

<h2>見出し</h2>

<p>本文</p>

</section>

<section>

<h2>見出し</h2>

<p>本文</p>

</section>

</article>

##### <nav>：主要なナビゲーション

サイト内を大きく移動するリンクなど。特にメニューによく使われる。

属性

具体例

<nav>

<h1>メインメニュー</h1>

<ul>

<li><a href="/">ブログ</a>

<li><a href="/menu">メニュー</a>

<li><a href="/about/"> 店舗情報</a>

<li><a href="/contact/">お問い合わせ</a>

</ul>

</nav>

##### <header>：ヘッダ

属性

具体例

<html>

<body>

<header>

<!-- いままでヘッダータグにかいていた内容をここに書く（らしい） -->

</header>

</body>

</html>

##### <button>：ボタン

type="" ボタンのタイプ

submit 送信ボタン （初期値）

reset リセットボタン

button 汎用ボタン ※on-clickはこれじゃないと動かない

menu ［5.1］メニュー表示ボタン

#### 属性

##### input type=”date”

<input type="date" name="birtdate" placeholder="select Birth date" >

##### placeholder

コントロールの内部に表示される，薄い灰色のヒント文字列の事。

<input placeholder="プレースホルダ">

<textarea placeholder="プレースホルダ"></textarea>

・複数行の入力はできない。

### 便利タグ

#### datalist

テキストボックスで入力中に表示される候補リストを指定する。

<label for="like-language">好きなプログラミング言語は？</label>

<input list="language" id="like-language" name="like-language" />

<datalist id="language">

<option value="HTML/CSS">

<option value="JavaScript">

...

</datalist>



label 補助メッセージを表示する（右図参照）

公式：<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/datalist>

表示文字と値

<option value="dog">犬</option>

ブラウザによって

#### details

<details>

<summary>詳細</summary> <!-- summaryを省略すると"詳細"　と表示 -->

<p>内容</p>

</details>

<details open> 開いた状態をデフォルトにする

#### iframe

概要

htmlの埋め込み。（htmlを入れ子で表示）

<body>

<iframe src="index2.html" width="500" height="500"></iframe>

</body>

## ヘッダ

### 基本

<meta charset="utf-8" /> <!-- 文字コードの指定 -->

<!-- スタイルシートの指定 -->

<link rel="stylesheet" href="css/style.css" />

<link　rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css" />

<script type="text/javascript" async="" src="js/sample.js">

script id="facebook-jssdk" とかいうのを見た 2022-04-23

### 一般的なタグ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| rel | type | href | 概要 | 備考 |
| icon |  |  | ファビコンを指定 | W3Cで推奨される |
| shortcut icon |  |  | Internet Explorerの初期バージョンでの指定方法 | あまり使われていないっぽい |

### ヘッダ専用タグ<meta>

#### 目次：メタタグの要素

|  |  |  |
| --- | --- | --- |
| 要素 | 概要 | 詳細／備考 |
| application-name | Webアプリの指定 | １つの文書に１つだけ |
| author | ページ制作者の名前 |  |
| name |  | contentと併用して使う？ |
| content | データ | 他の要素と併用して使う |
| charset | 文字エンコードの指定 | UTF-8を使うべき。 |
| http-equiv | プラグマ指示子 | contentと併用する。 |

※特筆する必用の無い要素は，この表の説明だけて終わらせる。

#### content

このcontent要素はname要素との組み合わせで様々な事ができる。

keywords：ページの内容を示すキーワード

<meta name="keywords" content="HTML,CSS,JavaScript,特殊文字,リファレンス">

description：ページを説明する短文。

<meta name="description" content="HTML・CSS等の早見表的リファレンス">

※この**descriptionは検索エンジンで利用される**ので，言葉は慎重に選ぶ。

robots ：検索エンジンのクローラ（ロボット）

noindex：URLをインデックスしないように

nofollow：文書内のリンクをたどらないように

<meta name="robots" content="noindex,nofollow">

20.09.23よく意味が分からない。

generator：ページ作成に使用したツール

<meta name="generator" content="Frontweaver 8.2">

##### viewpoint

レスポンシブデザインには欠かせない。

「width=device-width」「user-scalable」「initial-scale」などの属性値が使えるようになる（？）

<meta name="viewport" content="width=device-width, initial-scale=1, minimum-scale=1, maximum-scale=1, user-scalable=no">

　この設定値は：<https://tree-file.com/web-design/understanding-viewport-for-responsive-webdesign/>より

•width=device-width

表示領域の幅で、端末画面の幅に合わせる指定ができる。必ず記述がある。

width=320 # 数値での指定はpixel単位での指定となる。様々な理由で非推奨らしい。

•initial-scale

初期のズーム倍率指定。（minimum-scale～maximum-scale の範囲）必ず記述がある。

但し，省略した場合はinitial-scale=1が適用されるらしいので，意味がないかも。

「maximum-scale」（0～10 の範囲、初期値 1.6）。ほぼ間違いなく記述がある。

「minimum-scale」（0～10 の範囲、初期値 0.25）。記述がない事もある。

•user-scalable

ユーザーのズーム許可。値はyes/no　デフォルトでyesなので，記述が無い例もよくみられる。

併せてCSSで

body {

-webkit-text-size-adjust: 100%;

}

などの設定が必要

•height

widthと同じで，device-heightやピクセル単位での設定が可能。但しほとんど使われない。

•target-densitydpi

古いAndroid端末のために指定するもの。Android4.2以降では廃止。

～viewport 続き

悪い例：

width=device-width, initial-scale=0.5

縮小表示する必要がないにも関わらず縮小表示を強要。この場合、viewportの幅が予期せず大きくなってしまうことが有る。何故なら縮小するためにはviewportの幅が液晶画面よりも大きく無ければならないから。

この例だとviewportの幅はdevice-widthの2倍になってしまう。

width=320, initial-scale=0.5

320は大抵の端末のdevice-widthより小さい。やはり縮小の必要がない。

なのに縮小を強要するとviewportの幅を無理やり大きくして対応しようとしてしまう。

width=240, initial-scale=1.0

上の2つの逆パターン。viewportの幅は液晶画面より小さいのでそのまま表示したのでは液晶画面に余白ができてしまうため、デフォルトでviewportをズームアップするようになる。

つまり、暗黙のうちにinitial-scaleが1より大きな値となる。そこに無理やり1.0というズーム率を指定すると、液晶画面に余白が出来るのではなく、ブラウザはそれを防ぐために勝手にviewportの幅を大きくしてしまう。

width=1280,initial-scale=0.5

一見問題無さそうに見えるが、機種やブラウザによってはviewportが予期せぬ値に書き換えられる。viewportは大きくし、それを縮小して初期状態で液晶画面に収まるようにしたいという狙いで使いたくなるが非常に挙動が不安定。リロードする度に$(window).width()の結果が0になったり異常に大きな値になったりと訳の分からないことが起こった。initial-scaleに1より小さな値は使わないのが無難。

参考サイト：<https://qiita.com/ryounagaoka/items/045b2808a5ed43f96607>

#### charset

この用に使う。

<meta charset="utf-8"/>

htmlでは**常にUTF-8**を用いるべき。

このような方法もあるらしい

<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>

#### http-equiv

content-language： コンテンツ言語の指定

<meta http-equiv="content-language" content="ja">

content-type：ファイルタイプや文字エンコーディングの指定

<meta http-equiv="content-type" content="text/html" charset="UTF-8">

default-style：デフォルトスタイルの指定

<meta http-equiv="default-style" content="default.css">

refresh：リダイレクト・再読み込みの指定

<meta http-equiv="refresh" content="5; url=newpage.html">　<!-- 5秒後にリダイレクト -->

<meta http-equiv="refresh" content="300"> 　<!-- 5分ごとに再読み込み -->

### スクリプト系

<script type='text/javascript' src='assets/js/script.js' defer></script>

<script src="js/scripts.js"></script> <!-- html5ではこのように書くことも可能 -->

### その他

<meta http-equiv="X-UA-Compatible" content="IE=edge" />

<meta name="format-detection" content="telephone=no" /> <!-- 電話番号が自動的にリンク扱いにされるのを防ぐ。 -->

<meta name="viewport" content="width=device-width, initial-scale=1" />

<meta name="robots" content="index">

<meta name="author" content="SitePoint">

<meta property='og:locale' content="ja\_JP" />

<meta property="fb:app\_id" content="Facebook App ID" />

<meta property='og:site\_name' content="サイトのタイトル" />

<meta property='og:title' content="ページのタイトル" />

<meta name="twitter:card" content="summary\_large\_image" />

<meta name="twitter:site" content="ページのツイッターアカウント" />

<meta name="twitter:creator" content="ページコンテンツ製作者のツイッターアカウント" />

<meta property="og:description" content="ページの解説" />

<meta property="og:url" content="ページのURL" />

<meta property="og:type" content="website" />

<meta property="og:image" content="シェアされたときに表示する画像" />

<meta property="og:image:width" content="画像の横幅（数値指定）" />

<meta property="og:image:height" content="画像の高さ（数値指定）" />

<link rel="apple-touch-icon" sizes="180x180" href="icon.png" />

<link rel="canonical" href="ここに正規化するURL" />

## 具体例

### メニュー

<nav>

<h1>メインメニュー</h1>

<ul>

<li><a href="/">ブログ</a>

<li><a href="/menu">メニュー</a>

<li><a href="/about/"> 店舗情報</a>

<li><a href="/contact/">お問い合わせ</a>

</ul>

</nav>

### インデント

Bootstrapの場合 px-5 py-3など

Cssの場合 style="text-indent:200px;"

Htmlだけでやるばあい ulを使う？

## 資料

### htmlステータスコード

100番台は情報

200番台は成功

300番台はリダイレクト

400番台はリクエストエラー

500番台はサーバーエラー

|  |  |  |  |
| --- | --- | --- | --- |
| コード | 英語 | 日本語 |  |
| 102 | Processing | 処理を継続中 |  |
| 200 | OK |  |  |
| 301 | Permanent Redirect |  |  |
| 302 | Found |  |  |
| 303 |  |  | 稀 |
| 304 | Not Modified | リソースが未更新 |  |
| 307 |  |  | 稀 |
| 308 |  |  | 稀 |
| 400 | Bad Request | リクエストが不正 |  |
| 401 | Unauthorized | 認証が必要 |  |
| 403 | Forbidden | アクセス禁止 |  |
| 404 | Not Found | ページが見つからない |  |
| 500 | Internal Server Error | サーバー内エラー |  |
| 503 | Service Unavailable | サーバーが一時的に利用不可 |  |

### 公式レファレンス

MDN：html　 <https://developer.mozilla.org/ja/docs/Web/HTML>

# css

## 基本

### 基本

後に定義した（下の行で書いた）ものが優先される。

### 書き方

CSSの書き方は以下の３通り

方法１：別途.css ファイルを用意する。この方法が一番多い。

<head>

**<link rel="stylesheet" href="styles/styles.css">**

</head>

<body>

　　　<!-- 内容 -->

</body>

・ページ全体で統一感のあるデザインを作る事ができる

方法２：headタグ内に<style>タグを作る

<head>

**<style>**

menu { background-color: #1D566E;　}

**</style>**

</head>

<body>

　　　<!-- 内容 -->

</body>

・基本的にhtmlが読みにくくなる。

・外部CSS（上の方法）と併用するのが基本。

・特定のページに特化したデザインを適用したい場合は有効。

### セレクタ

セレクタ：CSSの適用範囲の書き方の事。

#### 基本

.class1{ /\* クラス指定 ：すべてのclass1に適用。**一番よく使う**セレクタ \*/

color: #F80206; /\* プロパティ：値 \*/

}

p{} /\* タグ指定 ：全てのpに適用 \*/

#headline{} /\* ID指定 ：tableのclass1にのみ適用 \*/

table.class1{} /\* タグ＋クラス指定 ：tableのclass1にのみ適用。あまり使用されない\*/

p␣strong　{} /\* <p> 内にある <strong> のみ適用。同プロパティがある場合，外側（この場合p）が優先 \*/

p span.class1　{} /\* <p> 内にある <spanの内，class1を指定されたものだけ。\*/

.main p　{} /\* class1を指定されたタグ内にある <p>タグ にのみ適用　\*/

a[target]{} /\* 属性セレクタ ：target属性を含む <a> タグのみ適用。あまり知られていない？ \*/

a[target="\_blank"] /\* 属性＋値 ：上の例＋値が　=”\_blank”　の要素のみ適用 \*/

#### 様々な指定方法

全要素：\*

\*{} /\* すべての要素に適用 \*/

p \* {} /\* 全てのp に適用 \*/

複数指定

h1, h2 {} /\* すべてのh1とh2 両方に適用される ※この使い方はtag, class 混合でも可能らしい \*/

.class1.class2 {} /\* 複数のクラスに適用。

<div class="class1 class2">Hello, world!</div>

子孫

div.class1 **>** a　{} /\* 直接の子孫　<div class=”class1”><a> など、直下にある要素のみ　\*/

.main **+** p　{} /\* 隣接　<div class=”main”><p> など，同階層の１つ目だけ適用。次の<p>は無効。\*/

.main **~** p　{} /\* 同階層　class=”main” と同階層のすべての<p>に適用。 \*/

補足

html側では複数のクラスの指定が可能

<p class=”main light”></p> <!-- mainとlightの２つのクラスを指定 -->

#### 特殊な指定方法

擬似クラス

<https://developer.mozilla.org/ja/docs/Web/CSS/Reference> > Pseudo-class

特定の要素

p:first-letter　{} /\* 全ての<p> の最初の文字だけに適用 \*/

:root ドキュメントツリーのルート要素

:first-child 親要素の最初の子要素

.menu li:first-child 　{} /\* <ul class=”menu”> 内の最初の <li>要素にのみ適用 \*/

.menu li:last-child {} /\* <ul class=”menu”> 内の最後の <li>要素にのみ適用 \*/

https://developer.mozilla.org/ja/docs/Web/CSS/:first-child

:last-child 親要素の最後の子要素

:only-child

:nth-child()

ul li:nth-child(odd) 　{} /\* ul内のliの奇数番目にのみ適用。当然evenもある \*/

.class1 tr:nth-child(3n) {} /\* .class1の適用されたtableのtrの3の倍数行。 \*/

<https://developer.mozilla.org/ja/docs/Web/CSS/:nth-child>

:nth-last-child()

:first-of-type

:last-of-type

:only-of-type

:nth-of-type()

:nth-last-of-type()

p:not(.main) {} /\*　全ての<p>タグの内，class=”main” でない要素のみ適用 \*/

ul li:not(:last-child):not(:first-child) /\*　notの組み合わせ。最初と最後のliは除く \*/

### テクニック

基本的に、cssによるデザインにはclassを多用する。

<h1> など最初からタグを使っているところは<h1 class=”midasi”> などとする。

タグが無いが，クラスを適用したいところには<div>や<span>を使う。（html4.01以前の場合）

CSSの指定方法

html側

<p class=”class1 class2”>段落１</p>

id

・id はページ内で一意(unique) にするべき。個別idととらえる。

css：

.className {　<!- 最初にドットをつける ->

}

html：

<div class=”className”></div>　<!- 必ずしもdivでなくても良い e.g <h1 class=”hoge”> ->

<div class=”class1 class2”>　　　　<!-- 2つ以上のクラスを適用させる場合 -->

アスタリスク

\*{

color:redl

}

p \* { /\* pタグの全ての要素に適用 \*/ }

### 良くあるクラス名

クラス名は単語1つだけの場合が非常に多く，２つ使う場合はハイフンで区切られることが多い。

menu メニュー表示部分

about 説明

news お知らせ

post 投稿や記事

large 大きい文字やサイズ

medium 中サイズの文字やサイズ

small 小サイズの文字やサイズ

### 数値

|  |  |  |  |
| --- | --- | --- | --- |
| 単位 | 種類 | 説明 | ピクセル換算 |
| px | 絶対単位 | ピクセル | 1px |
| em | 相対単位 | 親要素の「1」を１とした倍率。何文字分。 | - |
| ex | 相対単位 | 親要素の「x」の高さを１とした倍率。何文字分。 | - |
|  |  |  |  |
| in | 絶対単位 | インチ。2.54センチメートル | 96px |
| cm | 絶対単位 | センチメートル。10ミリメートル | 約37.80px |
| mm | 絶対単位 | ミリメートル | 約3.78px |
| pt | 絶対単位 | ポイント。1/72インチ相当。 | 約1.33px |
| pc | 絶対単位 | パイカ。12ポイント相当 | 16px |
|  |  |  |  |
| rem | 相対単位 | 親要素の文字サイズを１とした倍率 |  |
| vw | 相対単位 | viewportの幅に対する割合 |  |
| vh | 相対単位 | viewportの高さに対する割合 |  |

viewport簡単に言うとブラウザの幅，高さの事。

詳細

<meta name="viewport" content="width=480"> とすると幅480pxの仮想的なウィンドウが作られる。

これをviewportと呼ぶ。

但し実際にはWebページを表示する機器のpixel幅は分からない為、実際のviewportでは

width=device-widthと言う動的な挙動を指定する値が選ばれる。

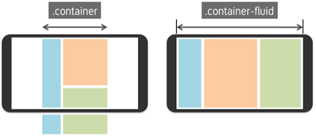
但し上の例ははみ出しなどが起こる反面，端末ごとの表示サイズを統一できる。

### コンテナ

#### 概要

テーブルよりも使い勝手が良いかも

#### container と container-fluidの違い



### 色

#### 指定方法

|  |  |
| --- | --- |
| 指定形式 | 指定例 |
| キーワード | black　red　blue　など |
| 16進数 | #ffcc00」「#c00」など、16進数を使ったカラーコード |
| RGB | rgb( 255, 42, 53)　　rgba( 255, 42, 53, 0.5)　透過度も指定できる |
| HSL | hsl( 0, 100%, 100%)　hsla( 0, 100%, 100%, 0.5)　透過度も指定できる |
| initial | デフォルト値を使用。CSSレベル3で追加（一部ブラウザではサポートされない） |

#### 属性

color フロントカラー。文字の色など。

background-color 背景色

.status table th{

border: 1px solid #E7E7E7;

color : #FFFFFF;

background-color: #0066CC;

}

参考<https://www.htmq.com/color/colorname.shtml>

### 入力関連

#### IME

'ime-mode: active'

active: IMEをオンにする。ユーザはオフにする操作ができる。

inactive: IMEをオフにする。ユーザはオンにする操作ができる。

disabled: IMEを無効にする。ユーザが状態を変えることはできない。これは使用してはならない。

具体例（.html）

<input type="text" value="" style="ime-mode: active">

### CSS3

便利な機能を増やした新しいCSSの規格。

## 具体的な問題対応

### テーブルの列幅対応

th(Table Header)タグで指定する事がおおいらしい。

<thead>

<tr>

<th> <!-- ここ -->

<tr>

</thead>

方法１：col-nを使う

<th class="col-1">内容</th>

方法３：直接styleで指定

<th style="width:200px;">

<th scope="col" style="width: 10%">10パーセント列</th>

⇒ こちらは列全部で100%になるように指定する必要があるらしい

⇒ 他に方法がない、またはどうしても固定にしたい場合の最終手段感がぬぐえない。

番外編：Bootstrap3までは

col-md-4

Bootstrap4.1以降：flex

flex-(\*\*-)fill:flex: 1 1 auto ショートハンド対応クラス

flex-(\*\*-)grow: flex-grow CSS プロパティ のコントロール

flex-(\*\*-)shrink: flex-shrink CSS プロパティ のコントロール

## レファレンス（抜粋）

<http://www.htmq.com/style/>

### 共通

GUI（グラフィックを使ったアプリケーション）の共通項目

#### margin／border／padding共通

各プロパティの違いは右図の通り

基本的に一括指定で，値の数によって挙動が異なる

値を1つ指定した場合：上下左右に適用

値を2つ指定した場合： 順番に[上下][左右]に適用

値を3つ指定した場合：順番に[上][左右][下]に適用

値を4つ指定した場合： 順番に[上][右][下][左]に適用（時計回り）

-radius

#### margin

マージン関連の一括指定

特徴：一括指定

### 表示・配置

配置に関した参考資料：

<https://creive.me/archives/13552/>

#### display　要素の表示形式を指定

特徴：多用する

inline インライン形式（改行なし）のボックス。width，height指定ができない。

block ブロック形式（改行あり）のボックス。要素が横いっぱいに表示。

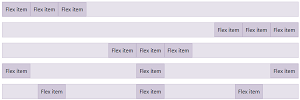
inline-block インライン形式。１行に詰められるだけ表示。大きさによって表示方法が動的に変化。

table-cell <table>のtdのように扱う。

flex 外側の構造タグに指定する。便利で，よく使われる。

list-item <li>のように扱う

run-in 文脈に応じてブロックまたはインラインボックスのいずれかを作成する

justify-content

寄せ、均等などの処理ができる（右図参照）

<div class="container d-flex justify-content-end">

<div class="d-flex justify-content-start">...</div>

<div class="d-flex justify-content-end">...</div>

<div class="d-flex justify-content-center">...</div>

<div class="d-flex justify-content-between">...</div>

<div class="d-flex justify-content-around">...</div>

公式：<https://developer.mozilla.org/ja/docs/Web/CSS/justify-content>

#### flex

display: flex; の時に使われる

flex-direction

row-reverse 要素が逆順に配置される

flex-wrap

wrap 画面の大きさに応じて，下に回り込む。floatのような感じ。

nowrap 回り込みの解除。floatで言うなら，こちらはclear

#### position

基本的にこのプロパティで全体のレイアウトを構成するのは得策でない。

スポット的に使う事が多くなる

static

特に指定なし。top、bottom、left、rightは適用されない。初期値です。

relative

positionプロパティでstaticを指定した場合に表示される位置が基準位置となる相対配置。

absolute

親ボックスにpositionプロパティのstatic以外の値が指定されている場合には、親ボックスの左上が基準位置となる。親ボックスにpositionプロパティのstatic以外の値が指定されてない場合には、ウィンドウ全体の左上が基準位置となる。

fixed

絶対位置への配置となるのはabsoluteと同じですが、スクロールしても位置が固定されたままとなります。

float 回り込み

指定された要素の配置を右寄せ、左寄せにする。画像の横に文章を回り込ませたい時などに使う。

ブロック要素を横に並べたい時はdisplay: inline-blockの方が適切。

値：**none**， left，right

（関連）clear

clear 回り込み解除

shadow

項目用rectの外側に影のような描写をする。

引数：水平(±)、垂直(±)、ぼかし(±)、広がり(±)、色、(insert)

※新しい手法として、drop-shadow と言うものがある IE11以前では未対応。

### 背景とボーダー

border

特徴：一括指定

（関連）border -top

border-radius ブロック範囲の角を丸くする

特徴：CSS3，一括指定

top-left　 top-right　bottom-right　bottom-left　の順で指定。

border-radius: 100px 25px 50px 50px

（関連）border-top-left-radius，border-top-right-radius

border-bottom-left-radius，border-bottom-right-radius

### Html標準視覚効果

#### テーブル

border-collapse ：隣接するセルのボーダーの描写方法を指定

collapse：重ねて表示 separate ：間をあけて表示

#### リスト

list-style

リストの先頭に表示するマーカーに関する指定をまとめて行う

特徴：継承あり，一括指定

通常，ol（番号付きリスト）やul（黒丸付きリスト）に指定する。

厳密に言うとliに指定する事になるが，子へと継承されるのでulやolで指定するのが合理的。

list-style-type

none ：中点（・）なし square ：

disc ：黒 circle ：白丸

decimal ：数字 decimal-leading-zero：

upper-alpha ：アルファベット大文字 lower-alpha

list-style-image

マーカーに画像ファイルを指定

list-style-image: url('../images/ico32.gif')

※list-style-typeと同時に使用された場合，list-style-imageが優先される

### background

background 背景プロパティの一括指定

特徴：一括指定

-color背景色，-image背景画像，並び方，表示位置，固定表示を一度に指定する事ができる。

順不同（らしい）

background-color 背景色

url 画像のファイルパスを選択

### テキスト

#### 表示位置

text-align 文横位置の指定

left：左寄せ center：中央揃え right：右寄せ

vertical-align テキストや画像などの縦方向の揃え位置

改行

overflow-wrap 表示範囲に収まらない長い連続する文字列を、途中で改行させるかどうか

word-wrap overflow-wrapプロパティの古い名称。

normal

break-word 単語の途中で自動改行

word-break 表示範囲最後まで来た時、単語が長かろうが短かろうが途中で改行させるかどうか

normal

break-all

keep-all 句読点などの決められた場所でしか改行しない。

#### 装飾

text-decoration ：太字、下線などのテキスト装飾の指定

text-decoration -line

**none** ：

underline ：下線

overline ：上線

line-through ：中央を貫く線

blink ：

text-decoration -style

**solid**

double

dotted

dashed

wavy

text-decoration -color ：

letter-spacing： 文字の間隔を指定する

word-spacing： 単語の間隔を指定する

#### フォント

font-family：

指定win, 指定 mac、

#### 疑似要素

:before：

選択した要素の直前に「疑似要素」を作成する。

例えばすべてのh4要素の前に ・や 『 をつけたりするのに使う。

:after：

:beroreの要素の直後版。

### イベント系

#### リンク

a:hover {} カーソルを上にのせたときのスタイル

a:visited {} 訪問済みリンクのスタイル

a:link {} 未訪問のリンクのスタイル

a:active {} クリックした瞬間（ボタンを押し込んだとき）のスタイル

### フォーム系

line-height ＝　textbox の高さ。文字列を扱う際の「行」の高さ。

/\* グローバル値 \*/

line-height: inherit;

line-height: initial;

line-height: unset;

### カテゴリ不明

margin-left: auto

## Webツール

|  |  |  |
| --- | --- | --- |
|  | アプリ／サービス名　とurl | 概要 |
| フォント比較： | <https://wordmark.it/> |  |
| テンプレート　全体イメージ | <https://template-party.com/> |  |
| テンプレート　カテゴリ別 | <https://blog.maromaro.co.jp/stylerecipe> |  |
| css id 最適化 | <https://css.miugle.info/> |  |
| ボタン最適化 | <http://css3button.net/> |  |
|  |  |  |
| ページデザイン | Figma  https://www.figma.com/ja/ |  |
| Html系の実行環境 | codepen  <https://codepen.io/> | ブラウザ上でHTML,CSS,JavaScriptのコードを確認 |
| Code Sandbox | https://codesandbox.io/ | ブラウザ上でHTML,CSS,JavaScriptのコード、特にReactの確認？ |

## css具体例ライブラリ

### 見出し



border-left: 6px solid #5ca3c5;

border-bottom: 1px solid #5ca3c5;

### テーブル

### ボタン

ボタンアイディア集）<https://saruwakakun.com/html-css/reference/buttons>

### メニュ－

（参考１）<https://copypet.jp/671/>

# cssフレームワーク

## フレームワーク比較

|  |  |
| --- | --- |
| 名前 | 概要 |
| Bootstrap | 最も有名なCSSフレームワーク。  jQuery及び独自のjsが組み込み済み。MITライセンス。 |
| Foundation | Bootstrapに次いで有名とのこと。  シリコンバレーにあるZURBという企業が管理。  独自のjsが組み込み済み。MITライセンス。 |
| Materialize | Googleが提唱した「マテリアルデザイン」を志向したFW。アメリカのカーネギーメロン大学の学生チームによって管理されている。  独自のjsが組み込み済み。MITライセンス。 |
| Semantic UI | HTTP、HTMLの設計者ティム・バーナーズ＝リーが提唱した「セマンティック・ウェブ」を志向したフレームワーク。  独自のjsが組み込み済み。MITライセンス。 |
| Pure.CSS | アメリカのYahoo!が提供。シンプルさ志向。  組み込みのJavaScriptは無し。Yahoo BSD License。 |
| BULMA | CSSのみでのフレキシブルデザイン(Flexbox)に対応。  組み込みのJavaScript無し。MITライセンス。 |

（情報元）<https://qiita.com/belq/items/10ec41f656e47ee2b540>

補足

・reactはUIに焦点を当ててこそはいるがJavaScriptのライブラリ

## Bootstrap

### 概要

公式ページのレファレンス：<https://getbootstrap.jp/docs/4.4/>

と具体的なCSSファイル（など）の公開サイトのサンプルを参照し合いながらデザインすることになる。

### チュートリアル

#### htmlのhead

JavascriptはjQuery、Popper.js、bootstrap の順番に記述する。

<!-- よくわからないが必要らしい 2022-02-28 -->

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<!-- Bootstrap CSSの読み込み -->

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">

<!-- Bootstrap 関連のJavascript読み込み -->

<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" crossorigin="anonymous"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"></script>

（情報元；公式）<https://getbootstrap.jp/docs/4.3/getting-started/introduction/>

#### components

ComponentsのSearchウィンドウで検索をかけると、関連するコンポーネントの情報が得られる。

（例）navbar→ナビゲーションバーの情報が出てくる

資料

### Web上の実行環境

Bootply

比較

|  |  |  |
| --- | --- | --- |
|  | 概要 | 備考 |
| Bootply | Bootstrap専用の実行環境 |  |
| CodePen |  |  |
| JSFiddle |  |  |
| StackBlitz | Angular, React, TypeScriptなどをサポート |  |
| Repl.it |  |  |

### グリッドシステム

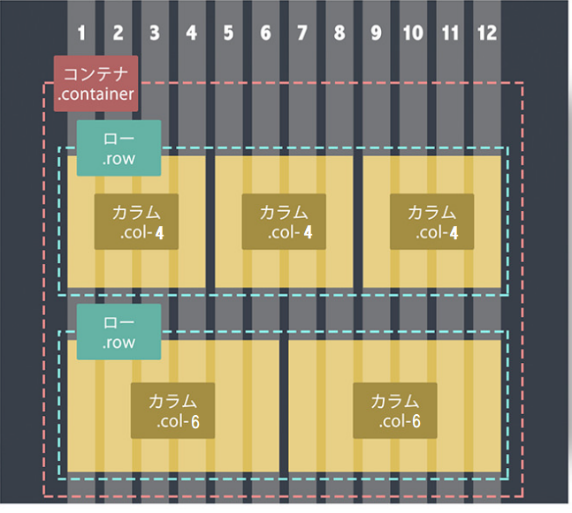
#### 概要

横幅を12分割し以下のルール下で使用する。

class = “row”の中に書く

rowはclass = “container”（固定枠）または”container-fluid”（流動枠）の中に書く

class = col-{columns} の形で書き、columnsの合計が12になるようにする



こんな感じで使う

<div class="container">

<div class="row">

<div class="col-4">col-4</div>

<div class="col-4">col-4</div>

<div class="col-4">col-4</div>

</div>

<div class="row">

<div class="col-6 col-sm-6">col-6</div>

<div class=" col-6 col-sm-6">col-6</div>

</div>

</div>

sm-8などのprefixはレスポンシブデザイン（省略可能）

書式：class ="col-{prefix}-{columns}"

|  |  |  |
| --- | --- | --- |
| prefix | 主な利用用途 | 画面幅 |
| xl | デスクトップ PC などの大画面モニタ | 1200px 以上 |
| lg | ノート PC などの省サイズモニタ | 992px 以上 |
| md | タブレット端末など | 768px 以上 |
| sm | スマートフォン（横持ち）など | 576px 以上 |
| - | スマートフォン（縦持ち）、IoT 端末など | 576px 未満 |

BootStrap3まで？

参考：<https://qiita.com/akatsuki174/items/53b7367b04ed0b066bbf>

#### 行／列

##### 行

<div class="row">行の要素</div>

補足：rowクラスはdisplay: flex; flex-wrap: wrap;として定義してある。

##### 幅を指定

<div class="col-6">2 of 3 (wider)</div>

補足：行の内、１つだけ幅を指定することもできる。

#### 幅／高さ

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | 備考 |
| タグ（幅） | .w-25 | .w-50 | .w-75 | .w-100 | .w-auto |  |
| タグ（高さ） | .h-25 | .h-50 | .h-75 | .h-100 | .h-auto |  |
| 意味 | 25% | 50% | 75% | 100% | auto |  |

#### 整列（上揃え、中央揃えなど）

##### 基本

align-items-start

align-items-center

float ブロック要素（h, a, p, divなど）に対しての指定

※厳密には文字列の回り込みらしい

float-left float-righ

float-sm-left float-md-left float-lg-left float-xl-left

以前はpull-rightだった？

pull-left ブロック要素（h, a, p, divなど）に対しての指定

pull-left，pull-center，pull-right

text-align テキストなどのインライン要素（）に対しての指定

text-align-left，text-align-center，text-align-right

##### rowの時

colと併用する必要がある。

<div class="row">

<div class="col text-center">

　row col sample

　</div>

</div>

d-flex

##### flex

Bootstrap4.1以降で使用可能

基本

flex-(\*\*-)fill:flex: 1 1 auto ショートハンド対応クラス

flex-(\*\*-)grow: flex-grow CSS プロパティ のコントロール

flex-(\*\*-)shrink: flex-shrink CSS プロパティ のコントロール

d-flex 子要素をフレックスアイテムとして配置

<div class="d-flex">

<div class="my-box w-25">Flex 1</div>

<div class="my-box w-25">Flex 2</div>

<div class="my-box w-25">Flex 3</div>

</div>

d-flex ブラウザの幅が変わっても１行

d-flexアイテムを並べる方向

flex-row 左から右方向

flex-row-revers 右から左方向

flex-column 上から下方向

flex-column-reverse 下から縦並び

d-[breakpoint-]flex

d-inline-flex ブラウザの幅が狭くなると２行になったりする。

d-sm-flex

d-xl-inline-flex

表示順序は標準CSSのjustify-contentを使ったりする

例：幅3の列にボタンを横幅全体に表示にして、その列を右から順番に表示していく場合

<div class="row d-flex justify-content-end">

<div class="col-3">

<button type="submit" class="btn btn-primary btn-block">検索</button>

</div>

</div>

align-items-start

align-items-center

文字列の右左寄せ

float-left float-righ

float-sm-left float-md-left float-lg-left float-xl-left

##### align関連

文字列の配置位置

<span class="align-baseline">ベースライン</span>

<span class="align-top">上端揃え</span>

<span class="align-middle">中央揃え</span>

<span class="align-bottom">下端揃え</span>

<span class="align-text-top">テキストの上端揃え</span>

<span class="align-text-bottom">テキストの下端揃え</span>

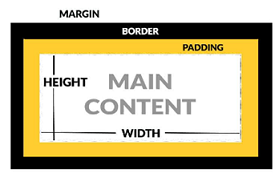
#### マージン

margin-top

margin-bottom

margin-left

margin-right



参考：<https://qiita.com/nakanishi03/items/c80a16b9b9796c25f890>

補足

bootstrapではmt, mbなどのクラスが定義されている。mt=margin-top, mb=margin-bottomの意

#### パディング

P

<nav class="navbar navbar-expand p-4">

#### 余白

###### 真ん中に余白

右図のような感じ。

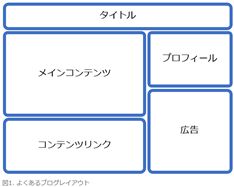
<div class="row">

<div class="col-md-4">.col-md-4</div>

<div class="col-md-4 offset-md-4">.col-md-4 .offset-md-4</div>

</div>

#### 資料



右図のようなレイアウトも可能らしい？

<https://www.hibiya-cs.co.jp/blog/今からでも遅くないhtml5-grid-layoutで今風デザインを簡潔に書く>

⇒ bootstrapではなく、スタイルシートで

grid-column: span 2やgrid-column: row2

などを使ったセルの結合らしい。が、アイディアとしては良い。

### 属性(class)

#### container

##### 概要

コンテンツの一番外側の枠の事で、bootstrap問わず、CSSの基本となる。

こんな感じで使う（らしい）

<div class="container　container-m">

<div class="row">

<div class="col　col-4">コンテンツ</div>

<div class="col　col-4">コンテンツ</div>

<div class="col　col-4">コンテンツ</div>

</div>

<div class="row">

<div class="col">コンテンツ</div><div class="col">コンテンツ</div>

</div>

<div>

補足：グリッドシステムに則って、合計が12になるようにcolの値を調整する

##### clearfix

##### 幅

container-fluid すべてのブレークポイントでwidth:100%

#### カード

右図のように、１つの情報の塊を枠内に表示させたりする。

<div class="card">

<div class="card-header h5">タイトル</div>

<div class="card-body">

<div class="card-text">

内容

</div>

</div>

</div>

</div>

（公式情報）<https://getbootstrap.jp/docs/5.3/components/card/>

#### タブ

tab-content

#### 文字列

<p class="text-primary">.text-primary</p>

<p class="text-secondary">.text-secondary</p>

<p class="text-success">.text-success</p>

<p class="text-danger">.text-danger</p>

<p class="text-warning">.text-warning</p>

<p class="text-info">.text-info</p>

<p class="text-light bg-dark">.text-light</p>

<p class="text-dark">.text-dark</p>

<p class="text-muted">.text-muted</p>

<p class="text-white bg-dark">.text-white</p>

文字の大きさ

<div class="container-fluid h4"> <!-- 大体どのタグにも使えるらしい -->

<div class="display-1"> <!—かなり大きめな見出し。1~4まで存在する -->

<div class="lead "> <!- 小見出しに使うリード文フォント。1.25rem 太さ300 -->

<div class="small"> <!- leadの逆 -->

#### リスト系

##### List Group

一連のコンテンツを表示する

・画面横に表示するメニュー項目

<ul class="list-group">

<li class="list-group-item">An item</li>

<li class="list-group-item">A second item</li>

<li class="list-group-item">A third item</li>

<li class="list-group-item">A fourth item</li>

<li class="list-group-item">And a fifth one</li>

</ul>

公式：<https://getbootstrap.jp/docs/5.3/components/list-group/>

#### 色

背景色

<div class="p-3 bg-primary text-white">.bg-primary</div>

<div class="p-3 bg-secondary text-white">.bg-secondary</div>

<div class="p-3 bg-success text-white">.bg-success</div>

<div class="p-3 bg-danger text-white">.bg-danger</div>

<div class="p-3 bg-warning text-white">.bg-warning</div>

<div class="p-3 bg-info text-white">.bg-info</div>

<div class="p-3 bg-light text-dark">.bg-light</div>

<div class="p-3 bg-dark text-white">.bg-dark</div>

<div class="p-3 bg-white text-dark">.bg-white</div>

bg-primary

text-white

Bootstrap4に用意されているクラス【color編】

<https://webnetamemo.com/coding/bootstrap4/201711026071>

#### form系

##### label



class="label label-default"

class="label label-primary"

class="label label-success"

class="label label-info"

class="label label-warning"

class="label label-danger""

<https://bootstrap.hana87.club/badge.html>

参考：<https://www.hts-web.biz/bootstrap-label/>

#### その他

##### 表示位置（ポジション）

基本

position-{static|relative|absolute|fixed|sticky}

|  |  |  |
| --- | --- | --- |
|  |  |  |
| fixed-top |  |  |
|  |  |  |
| sticky-top | ビューポートの上部に要素を端から端まで配置。  スクロールして通過した後でなければ、その要素は配置されない。 |  |

<https://getbootstrap.jp/docs/5.3/helpers/position>

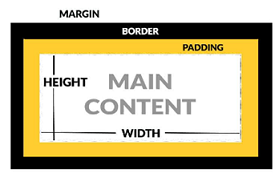
### 表示

#### html系

##### マージン

|  |  |  |
| --- | --- | --- |
| タグ | 概要 | 備考 |
| mt | cssのmargin-topに対応 |  |
| mb | cssのmargin-bottomに対応 |  |
| mr | cssのmargin-rightに対応 |  |
| ml | cssのmargin-leftに対応 |  |
| me | margin-endの意 | LTR言語では右、RTL言語では左。bootstrap5で対応 |
| ms | margin-startの意 | LTR言語では左、RTL言語では右。bootstrap5で対応 |

補足：RTL；Right To Left　アラビア語など右から左へ進行する言語の事

補足：marginはコンテンツの外側に設ける隙間の事（右図参照）

関連

・CSS標準のmargin-inline-start：<https://developer.mozilla.org/ja/docs/Web/CSS/margin-inline-start>

#### collapse

セルの非表示

<div class="collapse" id="collapseExample">非表示データ</div>

補足：htmlの属性にもcollapseというものがあるが、tableタグ専用。

#### ナビゲーション

##### 概要

<ul class="nav">

<li class="nav-item">

<a class="nav-link active" href="#">Active</a>

</li>

<li class="nav-item">

<a class="nav-link" href="#">Link</a>

</li>

<li class="nav-item">

<a class="nav-link disabled" href="#">Disabled</a>

</li>

</ul>

navbar-expand-sm

navbar-expand-lg

背景色

bg-light

nav-tab タブになる

data-toggle

<li class="nav-item">

<a class="nav-link" data-toggle="dropdown"

="dropdown"

="dropdown"

#### パンくず



<nav aria-label="breadcrumb">

<ol class="breadcrumb">

<li class="breadcrumb-item"><a href="#">Home</a></li>

<li class="breadcrumb-item"><a href="#">Library</a></li>

<li class="breadcrumb-item active" aria-current="page">Data</li>

</ol>

</nav>

公式：<https://getbootstrap.jp/docs/5.3/components/breadcrumb/>

#### ボタン

基本

<button type="button" class="btn">ボタン</button>

抜粋

|  |  |  |
| --- | --- | --- |
|  | 概要 | 備考 |
| btn-sm | 小さいボタン | -lg 大きいボタン |
| btn-primary | 色 | primary, secondary, success… など |
| btn-outline-primary | 線だけで表現されたボタン | primary, secondary, success… など |
| btn-link |  |  |
| btn-block | 属しているコンテナの幅いっぱいに広げる | Bootstrap v5で廃止  代案：.d-gridと.gap-\*ユーティリティで囲み、必要に応じて間隔を広げる |
| Disabled | 無効化 |  |

ボタン種類



アウトラインボタン



公式：<https://getbootstrap.jp/docs/5.3/components/buttons/>

#### フォーム

##### 概要

フォームのレイアウト関連。まず一読。

<https://qiita.com/zaburo/items/8983993d173c51cb3827>

##### インラインフォーム

内部のinputが横並びになる？

<form class="form-inline">

例：searchインプットボックスがインラインフォームになっている



以下の例のように、一部をインラインにする事もできる

<form>

<div class="form-group mr-3">

<label>お名前</label>

<div class="**form-inline**">

<label for="InputName1">性:</label>

<input type="email" class="form-control" id="InputName1">

<label for="InputName2">名:</label>

<input type="email" class="form-control" id="InputName2">

</div>

</div>

<div class="form-group mr-3">

<label for="InputAsk">お問い合わせ内容</label>

<textarea rows="5" class="form-control" id="InputAsk"></textarea>

</div>

</form>

補足：form-groupはbootstrap5.3で廃止

##### ラベル

代わりにbadgeというものがある

<span class="badge badge-primary">Primary</span>

##### フォームグループ（form-group）

概要

フォーム内の項目（≒input）毎に適用させるクラス

≒<div class="row"> と捉えると理解しやすい。その為、大きさを変えるときはcol-\* となる。

<form>

<div class="form-group">

<label for="InputName" class="control-label **col-2**">お名前</label>

<input type="email" class="form-control" id="InputName">

</div>

<div class="form-group">

<label for="InputAsk">お問い合わせ内容</label>

<textarea rows="5" class="form-control" id="InputAsk"></textarea>

</div>

<button type="submit" class="btn btn-primary">送信</button>

</form>

##### フォームロウ（form-row）

<form class="form">

<div class="form-row">

<div class="form-group **col-6**">

<label for="text4a">First name</label>

<input type="text" class="form-control" id="text4a" placeholder="First name">

</div>

<div class="form-group **col-6**">

<label for="text4b">Last name</label>

<input type="text" class="form-control" id="text4b" placeholder="Last name">

</div>

</div>

</form >

##### インプットグループ

概要

右図のようにビジュアル的にくっついたinputboxができあがる　

・インプットグループは <input>, <select>, <textarea> に適用できる。

・.input-group-prepend は前方部品、.input-group-append は後方部品

・サイズinput-group-sm、サイズ指定無、input-group-lg

<div class="input-group mb-3">

<div class="input-group-prepend">

<span class="input-group-text" id="basic-addon1">@</span>

</div>

<input type="text" class="form-control" placeholder="Username" aria-label="Username" aria-describedby="basic-addon1">

</div>

##### その他の関連属性

###### form-check-input

#### テーブル

|  |  |  |
| --- | --- | --- |
|  | 概要 | 備考 |
| table-bordered | テーブルに罫線が引かれる。 | 「普通のテーブル」というイメージ |
| table-borderless | 罫線なしのテーブル | コンテンツを整頓して表示したい場合などに使用 |
| table-striped | テーブル列の背景色が交互に | 少しおしゃれ。データ間違い防止用途などに |

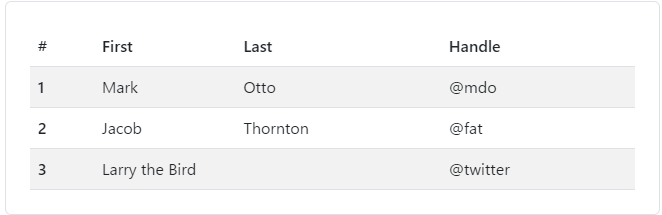


table-stripedの例（右図）

公式：<https://getbootstrap.jp/docs/5.3/content/tables>

幅の指定方法 以下のどちらか

<table>

<thead>

<tr style="width: 40%"></tr>

<tr style="width: 60%"></tr>

</thead>

</table>

<table>

<thead>

<tr class="col-4"></tr>

<tr class="col-8"></tr>

</thead>

</table>

補足：上記のように、<thead>内のすべての<tr>タグ内が空白の場合、ヘッダは描画されない。

#### リストグループ

一連のコンテンツを表示する

・画面横に表示するメニュー項目

<ul class="list-group">

<li class="list-group-item">An item</li>

<li class="list-group-item">A second item</li>

<li class="list-group-item">A third item</li>

<li class="list-group-item">A fourth item</li>

<li class="list-group-item">And a fifth one</li>

</ul>

抜粋：

|  |  |  |
| --- | --- | --- |
|  | 概要 | 備考 |
| list-group-numbered | 番号付きリストになる |  |
| list-group-item-action | ホバー時に背景色が変わる。 |  |
| list-group-item-primary | 色 | primary, secondary, success… など |
| disabled | 無効化された描画になる |  |

公式：<https://getbootstrap.jp/docs/5.3/components/list-group/>

#### リンク

##### 概要

##### nav-link

※パンくずでこれを入れると、見た目が崩れる事案が発生。原因不明。

⇒ nav（ナビゲーションバー）ではnav-linkを使わないのが普通かも

##### パンくずの基本

<nav aria-label="breadcrumb">

<ol class="breadcrumb">

<li class="breadcrumb-item"><a href="#">Home</a></li>

<li class="breadcrumb-item"><a href="#">Library</a></li>

<li class="breadcrumb-item active" aria-current="page">Data</li>

</ol>

</nav>

#### 表示位置

fixed-top

fixed-bottom

sticky-top

公式：<https://getbootstrap.jp/docs/5.0/helpers/position/>

#### メモ

d-inline-block

#### バッジ

badge

<ul class="list-group">

<li class="list-group-item d-flex justify-content-between

align-items-center">A list item

<span class="badge bg-primary rounded-pill">14</span>

</li>

<li class="list-group-item d-flex justify-content-between

align-items-center">A second list item

<span class="badge bg-primary rounded-pill">2</span>

</li>

<li class="list-group-item d-flex justify-content-between align-items-center">

A third list item

<span class="badge bg-primary rounded-pill">1</span>

</li>

</ul>

### 機能

#### ドロップダウン

<https://getbootstrap.com/docs/4.0/components/dropdowns/>

#### サブダイアログの表示

<https://getbootstrap.jp/docs/4.3/components/modal/>

最もシンプルな例

[ html]

<div class="modal" id="delete-confirm" tabindex="-1" role="dialog">

<div class="modal-dialog modal-dialog-centered">

<div class="modal-content">

<div class="modal-body">

<div>メッセージ</div>

<button type="button" class="btn" data-dismiss="modal">とじる</button>

</div>

</div>

</div>

</div>

modal-dialog-centered 縦方向の中心に表示

<div class="modal-dialog modal-dialog-centered">

header, body, fotter

<div class="modal-content">

<div class="modal-header"></div>

<div class="modal-body"></div>

<div class="modal-footer"></div>

</div>

### bootstrapライブラリ

#### 公式サンプル

<https://getbootstrap.jp/docs/4.3/examples/>

#### その他

明るめ

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">

<https://bootswatch.com/>

Flatly，Lumen，Lux，Slate，Yetiあたりが良い

### bootstrap icons

Install → CDNでlinkする為のhtml文字列がコピーできる。

<https://icons.getbootstrap.com/>

### フリー素材

Bootswatch

<https://bootswatch.com/>

Bootstrap 5 TEMPLATE

<https://mobirise.com/bootstrap-4-theme/?__c=1>

<https://getbootstrap.com/docs/4.1/examples/>

### 資料

5.1のチュートリアル（公式）

<https://getbootstrap.com/docs/5.1/getting-started/introduction/>

## Bulma

### 概要

CSSだけで成立しているCSSフレームワーク。BootstrapはJavaScriptまでセットになっている点が異なる。

つまり、vue.js、react、jQuery自由に組み合わせられるわけです。

## Markdown

シンタックスハイライトを使えるのは以下の言語

Bash, C#(cs), C++(cpp), CSS, Diff, HTML, XML, Ini, Java, Javascript, PHP, Perl, Python, Ruby, SQL, 1C, AVR Assembler(avrasm), Apache, Axapta, CMake, DOS .bat(dos), Delphi, Django, Erlang, Erlang, REPL, Go, Haskell, Lisp, Lua, MEL, Nginx, Objective C(objectivec), Parser3, Python, profile, Scala, Smalltalk, TeX, VBScript, VHDL, Vala

# スクリプト

## 基本

### PHPとJavaScriptの違い

|  |  |  |
| --- | --- | --- |
|  | JavaScript | PHP |
| 動作 | クライアント側  フロントエンドとかも呼ばれる。 | サーバー側  サーバーサイドとも言われる |
| 特徴 |  | フレームワークが多い |
| 主な用途 | ページのデザインに動きをつける | パスワードなどを扱う  データベースとの連携 |
| デメリット | ブラウザに依存しており，動作が一様でない。最悪動作しない。 | Webサイトに特化。他の環境で使えない。  セキュリティが弱い |
|  |  |  |

# JavaScript

## 概要

## タグ識別子

Identifier

例文

<form name="my\_form" id="my\_id" action="">

</form>

|  |  |  |
| --- | --- | --- |
| JavaScript | jQuery | 概要 |
| my\_form | ない？ | nameで識別 |
| getElementById("my\_id") | $("#my\_id") | 識別子にidを用いる |
| getElementsByClassName | $(".myclass") | classで識別 |

補足

var form = document.getElementById('main\_form')

その他

$('.q2:checked').length

メンバにアクセス？jQueryでのみ使える？

$('input[type="checkbox"]')

type=checkboxとなっているすべてのinputタグ。jQueryでのみ使える？

## プログラム情報

### 基本

function my\_method (){}

### 条件

#### if

|  |  |  |
| --- | --- | --- |
| 演算子 | 名称 | 意味／概要 |
| == | 等値演算子 | 左右の値が同じ場合にtrueを返す。  型変換はJavaScript側で自動的に行ってくれる。 |
| === | 同値演算子 | 値だけでなく、**型も同じ場合**にtrueを返す。 |
| !== | 非同値演算子 | 同値演算子の逆となります。 |

nullか空文字判定

if (!str) {

// do something

}

### Arrow function（匿名メソッド）

通常なら以下のように書くところ

const btn = document.getElementById("btn");

btn.addEventListener("click", btnEventfunction);

function btnEventfunction() {

console.log('クリック');

}

このように書く事ができる

btn.addEventListener('click', () => {

console.log('クリック');

});

パラメータ（引数）がある場合

btn.addEventListener('click', arg => {

console.log('クリック');

});

参考までに、こういった書き方もある

btn.addEventListener('click', function(){

console.log('クリック')

});

### fetch then

基本

fetch('http://...',{オプション}) // 非同期処理のターゲット？

.then((response)=>{ // 応答が合った場合

if(!response.ok){

throw new Error();

}

return response.blob(); // Binary Large Object?　あるいは response.json()

})

.then((blob)=>{ // 更に処理を続ける（メソッドリンク）

})

.catch((reason)=>{

//エラー

});

### 文字列操作

Formatっぽい事

var arg = "hoge";

// 変数を使う時は '（シングルクォーテーション）ではなく、　` である事に注意

console.log(`message is ${chk\_len}`)

文字のきりだし

year = "2020-2-22".substr(0, 4);

### 非同期処理

#### Promise

参考：<https://qiita.com/suin/items/97041d3e0691c12f4974>

## レファレンス

### ActiveXObject

MSのアクセス（db）になら、JavaScrip単体で接続できるらしい

function openConnection() {

// 「Connection」オブジェクト生成

var con = new ActiveXObject("ADODB.Connection");

// 接続文字列設定

con.ConnectionString = "Provider=MSDASQL" // プロバイダ

### button

onclick

注意点として、idやnameが"call"，"cancel"など

参考：https://stackoverflow.com/questions/9158238/why-js-function-name-conflicts-with-element-id

### Date

他プログラムのDateTimeの値の取得方法

var now = new Date();

var Year = now.getFullYear();

var Month = now.getMonth()+1;

var Date = now.getDate();

var Hour = now.getHours();

var Min = now.getMinutes();

var Sec = now.getSeconds();

### input

#### onkeyup

押下したキーを離した時のイベント

### window

#### open

構文：var newSubWindow = window.open(url, window\_name [, options]);

string url 新規ウィンドウで表示するURL

string window\_name ウィンドウ名（何でも良い？）

string options 幅やスクロールの有無など各種オプション。（カンマ区切り）

例

<a href="javascript:void(0);" onclick="window.open('popup.html')','subwin','width=500,height=500');">ポップアップテスト</a>

options

|  |  |  |  |
| --- | --- | --- | --- |
| オプション | 値 | def. | 説明 |
| width | 数値 |  | 横幅 |
| height | 数値 |  | 縦幅 |
| left | 数値 |  | 左からの位置 |
| top | 数値 |  | 上からの位置 |
| titlebar | yes / no | yes | タイトルバーの有無 |
| menubar | yes / no | yes | メニューバーの有無 |
| toolbar | yes / no | yes | ツールバーの有無 |
| location | yes / no | yes | アドレスバーの有無 |
| status | yes / no | yes | ステータスバーの有無 |
| resizable | yes / no | yes | リサイズの可否 |
| scrollbars | yes / no | yes | スクロールバーの有無 |
| directories | yes / no | yes | ディレクトリバーの有無 |
| fullscreen | yes / no | yes | フルスクリーンの有無 |

（公式）<https://developer.mozilla.org/ja/docs/Web/API/Window/open>

window.open

現在のウィンドウを開いたウィンドウへの参照を返す。

##### 親ウィンドウの値を変更する例

ディレクトリ構成

/

├ index.html

├ subwin.html

├ subwin.js

[ index.html ]

<!DOCTYPE html>

<html lang="jp">

<head>

<meta charset="UTF-8">

<title>subwin test</title>

</head>

<body>

<button type="button" onclick="window.open('subwin.html')','subwin');"> 開く</a>

</body>

</html>

<!DOCTYPE html>

<html lang="jp">

<head>

<title>sub win</title>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script src="subwin.js"></script>

</head>

<body>

<h2>subwin</h2>

<button type="button" id='btn-subwin'>テスト</button>

</body>

</html>

[ ./subwin.js ]

$(document).ready(function() {

$('#btn-subwin').click(() => {

//親ウインドウが開かれているか

if((window.opener) && (!window.opener.closed) ){

//親ウインドウのhtmの内容を編集

window.opener.document.getElementById("from\_subwin").innerHTML = 'hoge';

}

});

});

（公式）https://developer.mozilla.org/ja/docs/Web/API/Window/opener

（参考）http://web-designer.cman.jp/javascript\_ref/window/close/

補足：Internet Explorer以外のブラウザでは、window.openerとwindow.parentに違いはない。

##### 親ウィンドウで子ウィンドウが閉じた事を検知する

function closing\_test () {

win = window.open('example.com','subwin','width=400,height=300');

$(win).on('load', function () {

$(win).on('unload', function () {

console.log('閉じたよ');

});

});

}

イベントを利用する方法

function closing\_test () {

win = window.open('{% url 'gyoumu:popup\_select\_work\_day' %}','subwin','width=400,height=300');

$(win).on('load', function () {

win.dialog.cancel.addEventListener("click", () => {

console.log("「キャンセル」がクリックされました");

date = win.document.getElementById("id\_workdate").value;

win.close();

});

});

}

資料

<https://developer.mozilla.org/ja/docs/Web/API/Window/open>

#### alert

#### confirm

具体例：submitの確認ダイアログ

$('#form').submit(()=> window.confirm('送信してよろしいですか？')); // falseを返すと中止される

#### opener

そのウィンドウを開いた親ウィンドウを取得する

#### prompt

メッセージと入力フィールドを持った ダイアログボックスを表示します。

self

top

parent

### window.location

シンプルにこれでも良い

<input type="button" value="更新" onclick="window.opener.location.reload()">

href

リンクを提供する

<button onClick="jump ()">遷移する</button>

<script>

function jump () {

window.location.href = https://example.com /

}

</script>

### window.performance

#### navigation.type

ページへの移動方法を説明する

|  |  |  |
| --- | --- | --- |
| 値 | 定数名 | 意味 |
| 0 | TYPE\_NAVIGATE | リンク、ブックマーク、フォーム送信、スクリプト、またはアドレスバーに URL を入力してページにアクセスしました。 |
| 1 | TYPE\_RELOAD | このページには、[再読み込み] ボタンをクリックするか、Location.reload() メソッドを介してアクセスしました。 |
| 2 | TYPE\_BACK\_FORWARD | このページには履歴をたどってアクセスしました。 |
| 255 | TYPE\_RESERVED | その他の方法。 |

補足

ブラウザの「戻る」ボタンには対応していない。

公式：<https://developer.mozilla.org/ja/docs/Web/API/PerformanceNavigation/type>

### console.log

### element関連

#### createElementメソッド

post実行時などにformの要素を動的に追加する方法

function funcBtn() {

// エレメントを作成

var elem = document.createElement('input');

// データを設定

elem.setAttribute('type', 'hidden');

elem.setAttribute('name', 'hoge');

elem.setAttribute('value', 'テスト');

// 要素を追加

document.form\_id.appendChild(elem);

}

#### name プロパティ

#### styleプロパティ

#### InnerHTMLプロパティ

elementの全子孫のシリアライズされたHTMLコードを取得

（比較）textContentプロパティ（HTMLタグ文字列を含まない場合）

#### textContentプロパティ

console.log(document.getElementById('result').textContent);

#### valueプロパティ

（inputタグ限定）コントロールの現在の値を返却または設定

#### dataset プロパティ

別名：カスタムデータ

要素上に設定された全てのカスタムdata属性(data-\*)のアクセス

JavaScriptからだと.dataset[]で参照。

jQueryからだと.data()で参照。

// JavaScript

<script>

window.addEventListener('DOMContentLoaded', function(e){

console.log(document.querySelector("#hoge").dataset["test"]);

console.log(document.querySelector("#hoge").dataset["myTestData"]);

});

</script>

// jQuery

<script>

$(function(){

console.log($("#hoge").data("test"));

console.log($("#hoge").data("my-test-data"));

});

</script>

<div id="hoge" data-test="fuga" data-my-test-data="piyo">aaa</div>

（公式）<https://developer.mozilla.org/ja/docs/Web/API/HTMLElement/dataset>

（参考）<https://js.studio-kingdom.com/javascript/element/dataset>

良くみられるもの

data-toggle BootstrapのJavaScriptプラグインで使われる。

modal:モーダルウィンドウ

dropdown:ドロップダウン

tooltip:ツールチップ

tab:タブ

pill:ピルナビゲーション

### document関連

#### 概要

表示している文書自体を示す。大体の場合は≒表示中のhtmlドキュメントの意。

XML、

#### cookie　プロパティ

文字通り、クッキーを扱う

document.cookie

#### getElementById()　メソッド

htmlの中から、指定されたidの属性を持つ要素を返す。

構文：

element = document.getElementById(id);

例文：

<!DOCTYPE html>

<html>

<head>

<title>getElementById example</title>

<script>

function changeColor(newColor) {

var elem = document.getElementById("para1");

elem.style.color = newColor;

}

</script>

</head>

<body>

<p id="para1">Some text here</p>

<button onclick="changeColor('blue');">blue</button>

<button onclick="changeColor('red');">red</button>

</body>

</html>

（参考）https://js.studio-kingdom.com/javascript/document/get\_element\_by\_id

#### getElementsByClassName

与えられたクラス名を持つ全ての子要素を格納した、配列に似たオブジェクトを返す。

構文

var elements = document.getElementsByClassName(names);

var elements = rootElement.getElementsByClassName(names);

#### createElement

#### createDocumentFragment

#### querySelector

#### document.location.href;

### イベント関連

#### .addEventListener()

構文：３つのオーバーロード

event\_target.addEventListener(type, listener);

event\_target.addEventListener(type, listener, options);

event\_target.addEventListener(type, listener, useCapture);

例文

var element = document.getElementById("outside");

element.addEventListener("click", modifyText, false);

document.addEventListener('keyup', e => {});

次ページに続く

#### setInterval

setInterval(function(){ some\_method (); }, 1000);

func

実行する関数です。または関数名の文字列。

最初の実行は delay ミリ秒後に行われます。

delay

タイマー時間をミリ秒 (1/1000 秒) 単位で指定します。

引数が 10 より小さい場合は、10 を使用します。

arg0, ..., argN 省略可

タイマーが満了したときに、 func で指定した関数に渡す追加の引数です。

関連：setTimeoutメソッド

補足：jQueryなどと組合す事も可能

公式：<https://developer.mozilla.org/ja/docs/Web/API/setInterval>

〜

addEventListener()　つづき

type

|  |  |
| --- | --- |
| イベント種類 | 内容 |
| click | マウスがクリックされた場合 |
| keydown | キーボードが押された場合 |
| keyup | キーボードが離された場合 |
| mousedown | マウスボタンが押された場合 |
| mouseup | マウスボタンが離された場合 |
| mousemove | マウスカーソルが移動した場合 |
| mouseover | マウスカーソルがオブジェクトに重なった場合 |
| mouseout | マウスカーソルがオブジェクトから離れた場合 |
| onLoad | ページが読み込まれた場合 |
| onUnload | 他のページに移動する場合 |
| focus | 選択された状態になった場合 |
| blur | フォーカスを失った場合 |
| submit | フォームがサブミットされた場合 |
| reset | フォームがリセットされた場合 |
| change | フォームの内容が変更された場合 |
| resize | ウインドウがリサイズされた場合 |
| abort | イメージの読み込みが中断された場合 |
| error | 読み込みが失敗した場合 |
| change | 値が変更された場合（？） |
| load | Webページの読み込みが完了した時に発動（画像などのリソースすべて含む） |
| DOMContentLoaded | HTML ページの読込み、解析、DOMツリー構築が完了した時 |

onchange?

DOMContentLoaded

jQueryの$(document).readyに似ている。

readyは割と昔からの実装で、DOMContentLoadedは比較的新しい点が大きな違い（らしい）

資料

（公式）https://developer.mozilla.org/ja/docs/Web/API/EventTarget/addEventListener

（参考）https://js.studio-kingdom.com/javascript/event\_target/add\_event\_listener

## JavaScriptライブラリ

### numeral.js

### AutoNumeric.js

### popper.js

簡単にツールチップのような吹き出しのようなパーツを作れる

Bootstrap内でも使用される

公式：<https://popper.js.org/>

## JSX

別名：JavaScript XML

XML のような構文を使用して DOM ツリーを作成できるようにする JavaScript 拡張機能。

Reactで用いられる

詳細は別ドキュメント「開発系」

## トラブルシューティング

### Uncaught DOMException:

ウェブサーバ上のコンテンツを実行していない場合に発生

## 資料

MDN：JavaScript　<https://developer.mozilla.org/ja/docs/Learn/JavaScript>

## Web Bluetooth

Java Script でBluetoothセンサなどの情報を直接扱う。

（参考資料）<https://houwa-js.co.jp/2021/03/20210316/>

例えばGoogle Chartsと合わせるとWebブラウザだけでセンサの見える化ができる。

# PHP

## 概要

Hypertext Preprocessorの略でサーバーサイドのスクリプト。オープンソースの言語。

・サーバー側で処理されるので，ブラウザでhtmlのコードを見ても，<? PHP の表記は見れない。

・index.htmlの代わりにindex.phpを使うのが基本だが、設定を変更 .htmlでも実行可能。

グレード

CGI 版

モジュール版

LiteSpeed 版

## 設定ファイル：

これらのファイルのパスはphpinfo() で表示される。

/etc/php/[ver]/cli/php.ini

/etc/php/[ver]/apache2/php.ini apache向きphpの設定

### php.ini

extension\_dir 拡張モジュールを設置するディレクトリを指定。phpinfo()で確認できる。

### .htmlでPHPが実行できるようにする

AddTypeで対応する方法

<FilesMatch "\.html$">

AddType application/x-httpd-php .html

</FilesMatch>

AddHandlerで対応する方法

<FilesMatch "\.html$">

AddHandler php5-script .html

</FilesMatch>

AddHandler php5-script .php

## 基本

### 基本コマンド

php -v # バージョンの表示

php -m # モジュールの表示

php -i # 情報を表示

php -r "echo date('Y-m-d');" # 簡単なコードを試す　-r　オプションでコマンドの直接実行

### モジュールの読み込み

他言語のimportとかusingにあたる。php.iniの下の方にextention から始まる行が大量に存在する。

そこでコメントアウト(；)を外す事で使用可能になる（らしい）

;extension=pgsql

↓

extension=pgsql

## 関数リファレンス

phpinfo() phpのバージョンなどの情報が表示される。

※ちなみにここで表示される、php.iniなどの情報は結構重要。

（参考）<https://www.php.net/manual/ja/>

## 疑問

# Ruby

MVCフレームワークの元祖

# ライブラリ

## 概要

### JavaScript系ライブラリ

ブラウザ間の差異を吸収する目的で用いられる

|  |  |  |
| --- | --- | --- |
| 名前 | 種類 | 概要 |
| jQuery | ライブラリ | JavaScriptのライブラリ 少し古いものになりかけている（2022年現在） |
| React.js | ライブラリ | UIパーツのライブラリ  フレームワークとも言われる（らしい） |
| Node.js | ライブラリ | サーバーサイドのJavaScriptの実行環境。  (Webブラウザでも可） |

Webアプリフレームワーク

|  |  |  |
| --- | --- | --- |
| 名前 | 種類 | 概要 |
| Angular.js | フレームワーク | フロントエンド用フレームワーク。  Googleが開発。（旧）AngularJS |
| Vue.js | フレームワーク | フロントエンド用フレームワーク。  MVVM(Model-View-ViewModel)を採用。 |

### 静的サイトジェネレーター

|  |  |  |
| --- | --- | --- |
| 名前 | 種類 |  |
| Gatsby.js | Reactベース |  |
| Next.js | Reactベース | Webアプリフレームワーク |
| Nuxt | Vue.jsベース |  |
| Hugo |  | Linux機にはインストールできるらしい |

静的サイトジェネレーターの詳細は別ドキュメント「クラウド系」を参照

## jQuery

### 概要

$("セレクタ") で始まり、メソッドをつないでいく。なお、$()と「jQuery()」は同じ。

一番わかり易い例は下記。

例

$(document).ready(function() {

//処理を実行する

});

documentはJavaScriptで文書、特に現在のhtml文書を扱う。

つまり、このソースはhtmlが表示されたら、functionを実行する、という形。

なお、以下のソースと全く同じ結果になる。（省略型）

$(function(){

// 処理を実行する

});

なお、使用するにはヘッダーでjqueryの読み込みが必要

<head>

<title>jQuery</title>

<script src="http://code.jquery.com/jquery-2.0.0.js"></script>

</head>

次ページに続く

$("#myid") 識別子にidを用いる場合；単一の要素を対象にする。

<html>

<head>

<title>サンプル</title>

<script type="text/javascript" src="jquery.js"></script>

<script type="text/javascript">

$(document).ready(function(){

$("#**btn**").click(function(){

alert("Click");

});

});

</script>

</head>

<body>

<form>

<input type="button" value="Click" id="**btn**">

</form>

</body>

</html>

$(".myclass") 識別子にclassを用いる場合；複数の要素を対象にする。

2022-04-24こんなこともできる様子

$(function(){

var list\_str = j$( "#name\_list" ).val();

var list = list\_str.split(',');

$( "#select\_name" ).autocomplete({

minLength: 0,

source: list

}).focus(function() {

$(this).autocomplete("search", "");

});;

});

### 基本

#### 基本ファイル構成

ディレクトリ構成

├ index.html

├ js\

│ ├ script.js

[ index.html ]

<!DOCTYPE html>

<html lang="jp">

<head>

<meta charset="UTF-8">

<title>Document</title>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

<script type="text/javascript">

// 直接記述する場合（この辺りは普通のjavaScript と同じ）

</script>

<script type="text/javascript" src="js/script.js"></script> <!-- 別ファイルを読み込む場合 -->

<script src="js/scripts.js"></script> <!-- html5（以降？）ではこのように書くことも可能 -->

</head>

<body>

<button type="button" onclick="some\_method();">実行1</button>

<button type="button" id="btn\_excute">実行2</button> <!-- この方式の方がいろいろと都合が良い -->

</body>

</html>

[ js/script.js ]

$(document).ready(function() { // htmlドキュメントが開かれた時に実行されるイベント

console.log('Hello World'); // つまり、画面読み込み時にこの行の処理が実行される。

$('#btn-excute').click(() => { //　イベントリスナーの追加。この場合、id=btn-excuteクリックイベント

console.log('btn-excute clicked!'); //　ボタンクリック時の処理。

})

});

function some\_method() {

console.log('some\_method invoked!'); // こちらは普通のjavascript

}

#### オプション

|  |  |  |
| --- | --- | --- |
| 種別 | 名前 | 説明 |
| 基本 | url | リソースの URL を指定します。  (例: url: "../common/getdata.php") (1.0) |
| type | HTTP メソッドの種別を "GET" か "POST" で指定します。jQuery 1.9 以降では、type の代わりに method を使用します。 |
| method | "GET", "POST", "PUT" などのリクエストメソッドを指定します。  (例: method: "POST") (1.9) |
| data | サーバに送信するフォームデータを指定します。  (例1: data: { id:"tanaka", pw:"qWeRtY" })  (例2: data: "id=tanaka&pw=qWeRtY") (1.0) |
| timeout | はタイムアウト値(ミリ秒)を指定します。  (例: timeout: 10000) (1.0) |
| 結果取得 | success | 通信が成功した時に呼ばれるコールバック関数を指定 |
| error | 通信が失敗した時に呼ばれるコールバック関数を指定 |
| complete | 成功・失敗に関わらず通信が完了した時に呼ばれるコールバック関数を指定します。詳細は後述。 (1.0) |
| 送信データ処理 | mimeType | リクエストの MIMEタイプを指定します。  (例: mimeType: "application/json") (1.5.1) |
| contentType | リクエストの Content-Type を指定します。規定値は、送信データが存在する場合は "application/x-www-form-urlencoded; charset=UTF-8" です。送信データが存在しない場合、通常は Content-Type は付加されませんが、contentType を指定した場合は強制的に付与されます。 |
|  | (例: contentType: "application/json; charset=UTF-8") (1.0) |
| processData | data で指定した送信データをパーサで処理するか否かを true / false で指定します。規定値は true で、自動的に "application/x-www-form-urlencoded" 形式に変換されますが、false を指定することで、DOMDocument や 非変換データを無変換で送信することができます。 |
|  | (例: processData: false) (1.0) |
| traditional | true を指定すると、1階層分の変換のみを行う古い形式のパラメータ変換を行います。 |
|  | (例: traditional: true) (1.0) |
| headers | 追加の HTTPヘッダ情報をヘッダ名と値のマップで記述します。 |
|  | (例: headers: { "X-Oroginal-Header" : "data..." }) (1.5) |
| beforeSend | リクエスト送信前に呼び出されるコールバック関数を指定します。コールバック関数が false を返却した場合、リクエストは中断されます。 |
|  | (例: beforeSend: function(xhr, options) { ... }) (1.0) |

|  |  |  |
| --- | --- | --- |
| 認証 | username | Basic認証を用いる際のユーザ名を指定します。 |
|  | (例: username: "tanaka") (1.0) |
| password | Basic認証を用いる際のパスワードを指定します。 |
|  | (例: password: "qWeR+tYu") (1.0) |
| 受信データ処理 | dataType | "text", "html", "xml", "script", "json", "jsonp" のいずれかで指定  省略時はレスポンスの MIMEタイプなどから jQuery が自動的に判断します。  (例: dataType: "json") (1.0) |
| accepts | dataType で指定したタイプに対して、HTTP の Accept: ヘッダで送信する MIMEタイプを指定します。  (例: accepts: { text: "text/plain", json: "application/json" }) (1.0) |
| contents | データタイプと、レスポンスの Content-Type の正規表現とのマッピングを記述します。例えば例の場合、レスポンスの Content-Type が text/x-yaml の場合、正規表現 /yaml/ にマッチし、そのデータタイプは yaml であるとみなされます。  (例: contents: { "yaml": /yaml/, "csv":/csv/ }) (1.5) |
| converters | contents で判別したデータタイプに対するパーサ関数を指定します。これにより、未知の MIME タイプのデータでも、独自パーサで敵セスに変換した後に done() メソッドに渡すことが可能となります。  (例: converters: { "\* yaml": function(data) { ...; return convertedData; } }) (1.5) |
| dataFilter | json や xml などのパーサで解釈される前の生のレスポンスデータを参照・更新するコールバック関数を指定します。 |
|  | (例: dataFilter: function(data, type) { ... }) (1.0) |
| cache | キャッシュを利用するか否かを true / false で指定します。規定値は通常 true ですが、dataType が json や jsonp の場合は false となります。cache が false の場合、If-Modified-Since や If-None-Match などのヘッダ送信が抑制され、また、URL の末尾に ?\_=タイムスタンプ を付加することで毎回異なる URL がリクエストされるようになります。 |
|  | (例: cache: false) (1.0) |
| ifModified | true を指定すると、リクエストヘッダに If-Modified-Since ヘッダが含まれ、レスポンスが 304 Not Modified の場合に、success: や done() とならず、error: や fail() にエラー番号 304が返却されます。これにより、リソースが、すでに受信炭のデータか否かを見分けることができます。ただし、レスポンスヘッダに Last-Modified ヘッダが含まれない場合はうまく機能しないようです。 |
|  | (例: ifModified: true) (1.0) |
| statusCode | レスポンスの HTTP ステータスに応じたコールバック関数を指定します。 |
|  | (例: statusCode: { 404: function() { alert("Not Found!!"); } }) (1.5) |
| スクリプト呼び出し | scriptCharset | type が "GET" で、dataType が "script" または "jsonp" の場合に、読み込んだスクリプトの文字コードを指定します。 |
|  | (例: scriptCharset: "UTF-8") (1.0) |
| scriptAttrs | スクリプトに引き渡す属性リストを指定します。(3.4) |
| jsonp | (未稿) (1.0) |
| jsonCallback | (未稿) (1.0) |

|  |  |  |
| --- | --- | --- |
| その他 | async | false を指定すると Ajax が同期的に実行され、Ajax 処理が完了するまで次の処理を実行しません。 |
|  |  | (例: async: false) (1.0) |
|  | context | done() などのコールバック関数で $(this) として参照されるオブジェクトを指定します。 |
|  |  | (例: context: document.body) (1.0) |
|  | global | .ajaxStart() などのグローバル Ajaxイベントを実行するか否かを true / false で指定します。規定値は true です。 |
|  |  | (例: global: false) (1.0) |
|  | isLocal | 処理をローカルのものとして扱うか否かを true / false で指定します。通常、プロトコルが file, \*-extension, widget の時のみローカルと認識されますが、isLocal を true にすることでそれ以外の場合でもローカルと見なして扱います。設定する場合は、$.ajaxSetup() で設定します。 |
|  |  | (例: isLocal: true) (1.5.1) |
|  | crossDomain | 同じドメイン上で JSONP の様なクロスドメイン要求を強制的に使用したい場合に true を指定します。例えば、他ドメインへのサーバサイドリダイレクションなどで利用されます。規定値は、同ドメインの場合は false、クロスドメインの場合は true です。 |
|  |  | (例: crossDomain: true) (1.5) |
|  | xhr | XMLHttpRequest オブジェクトを生成するためのコールバック関数を指定します。規定値では、IE の場合は ActiveXObject()、その他のブラウザでは XMLHttpRequest() が使用されます。 |
|  |  | (例: xhr: MyXMLHttpRequest) (1.0) |
|  | xhrFields | XHR オブジェクトに追加で設定する key-value マップを指定します。 |
|  |  | (例: xhrFields: { withCredentials: true }) (1.5.1) |

#### 他メソッドの実行

function somemethod(){

}

// 方法1

$('#btn').click(somemethod)

// 方法2

$('#btn').click(() => click())

// 方法3

$('#btn').click(() => { click(); })

#### jsonの取得

function reflesh\_image() {

$.ajax({

type: "GET",

url: "/myapi",

dataType: "json", // jsonデータとして取得 【ここが重要！】

}).done(response => {

$("#result ").text(`${response.result}`); // Base64エンコードして表示

})

}

#### append

要素を追加する

$('#elem').append('<p>Hello</p>'); # 要素の末尾（内側）に追加

（公式）<https://developer.mozilla.org/ja/docs/Web/API/Element/append>

#### attr

属性を変更する

$('#elem').attr('title')

#### html

$("#elem ").html("なし");

#### on

対象要素.on( イベント名, セレクタ, データ ,関数 )

subwin = window.open(

"subwin.html",

"\_blank",

"menubar=0,width=600,height=400"

);

$(subwin).on('unload', function(){

console.log('subwin closed!');

})

イベント

blur, focus, load, resize, scroll, unload, beforeunload, click, dblclick, mousedown, mouseup, mousemove, mouseover, mouseout, mouseenter, mouseleave, change, select, submit, keydown, keypress, keyup, input

公式：<http://api.jquery.com/Types/#Event>

#### show/hide系

$('#id\_hoge').show();

$('#id\_hoge').hide();

$('#id\_hoge').fadeOut();

#### click系

click() クリック操作

dblclick ダブルクリックされた際に、登録したイベントを発火

mousedown マウスが押されたときにイベントが発火

mouseup マウスが押された状態から離されたときにイベントが発火

contextmenu 右クリックでイベントを発火

### form系

#### checkbox

if($('#checkbox1').prop('checked')){ // チェック状況の取得

}

$('#checkbox1').prop("checked", true); // チェック状況の変更

#### textbox

keyup

$('#tb\_sample').keyup(()=> my\_method());

#### option

##### 概要

<input id="id\_datalist\_input" type="text">

<datalist id="id\_datalist ">

<option value="val1"\>

<option value="val2"\>

</datalist>

$('#id\_datalist\_input').change(function() {

}

##### 表示文字と別の値をpostする場合

id = $("#id\_datalist option[value='" + $(this).val() + "']").data('id');

補足：datasetプロパティ

data-hoge とする事で一時的に独自のタグを作成できる

例：

<body>

<p data-name="taro" data-age="30">私は太郎です</p>

</body>

<script>

const name = $('p').data('name');

console.log( result );

</script>

#### tooltip



[ .html]

<script type="text/javascript">

$(function () {

$('[data-toggle="tooltip"]').tooltip();

})

</script>

<a href="#" data-toggle="tooltip" data-original-title="説明">リンク</a>

JavaScriptの部分は以下でも良い

// 同等

$(document).ready(function() {

$('[data-toggle="tooltip"]').tooltip();

});

関連プロパティ（属性）

|  |  |  |  |
| --- | --- | --- | --- |
|  | 値 | 概要 | 初期値 |
| data-placement | left, bottom, right, top | ツールチップの表示場所 | top |

資料

公式：<https://jqueryui.com/tooltip/>

### プログラム系

ループ

for (var i = 0; i < response.name\_list.length; i++) {

console.log(response.name\_list[i]);

}

$.each(response.name\_list, function(idx, value) {

console.log(value);

});

### メモ

https://notepad-blog.com/content/150/

### ライブラリ

#### jQuery-contextMenu

<https://swisnl.github.io/jQuery-contextMenu/docs.html>

// コンテキストメニュー（右クリックメニュー）の処理

// https://swisnl.github.io/jQuery-contextMenu/docs.html

$.contextMenu({

selector: 'input.context-menu',

callback: function(key, options) {

var id = options.$trigger.attr("id");

if (key == 1){

$('#' + id).val(Number($("#ajax\_result\_charge\_1").text()));

} else if (key ==2){

$('#' + id).val(Number($("#ajax\_result\_charge\_2").text()));

} else {

$('#' + id).val(Number($("#ajax\_result\_charge\_3").text()));

}

},

items: {

{% with ''|center:3 as range %}

{% for \_ in range %}

"{{ forloop.counter }}": {name: "チャージ{{ forloop.counter }}"},

{% endfor %}

{% endwith %}

}

});

#### テーブル系

##### Tabulator

HTMLテーブルの id を指定することでテーブルのデータ を読み込み、表示する

Jsonからテーブルを作成できたり、各列でソートかけたりと色々とできる。

###### チュートリアル

ほとんど公式のチュートリアルに従えばよい

<http://tabulator.info/docs/5.2/quickstart>

[ index.html （抜粋）]

<!DOCTYPE html>

<html lang="ja">

<head>

<meta charset="UTF-8">

<title>Tabulator</title>

<script src="https://code.jquery.com/jquery-3.6.0.js" integrity="sha256-H+K7U5CnXl1h5ywQfKtSj8PCmoN9aaq30gDh27Xc0jk=" crossorigin="anonymous"></script>

<link href="https://unpkg.com/tabulator-tables@4.2.7/dist/css/tabulator.min.css" rel="stylesheet">

<script type="text/javascript" src="https://unpkg.com/tabulator-tables@4.2.7/dist/js/tabulator.min.js"></script>

</head>

<body>

<script>

$(document).ready(function () {

var table = new Tabulator("#example-table", {

// 列の設定などが必要

});

<div id="example-table"></div> <!-- テーブルはここに表示される -->

</body>

</html>

直接記述する場合

var tableData = [

{id:1, name:"Billy Bob", age:"12", gender:"male"},

{id:2, name:"Mary May", age:"1", gender:"female"},

]

var table = new Tabulator("#example-table", {

data:tableData,

}

Jsonファイルを読む場合

var table = new Tabulator("#example-table", {

ajaxURL:"http://www.getmydata.com/now", // URLを指定する場合

ajaxURL:"static/json/data.json", //　サーバー内ファイルを指定する場合（例）

});

（補足；ファイル読み込み方式だと、http.severやapacheなどのhtmlサーバーが必要）

（公式；データの読み込み方）<http://tabulator.info/docs/5.2/data>

###### コンストラクタ

概要

var table = new Tabulator("#example-table", {

data:tabledata,

autoColumns:true,

});

概要

|  |  |  |
| --- | --- | --- |
| パラメータ | データ型 | 概要 |
| autoColumns | bool | columns の設定なしで、自動的に列を生成してくれる。 |

公式：<https://tabulator.info/docs/5.5/options#table>

columns

formatter

よく使いそうなもののみ抜粋

star スターの表示

関連するformatterParams：stars 星の最大数(def.=5)

tickCross バツまたはチェックの表示／入力

money

progress プログレスバー

関連するformatterParams：min:0, max:10,

例：

columns:[{

title:"進捗", field:"seconds", sorter:"seconds", formatter:"progress",

formatterParams:{

min:0, max:1000,

},

}],

datetimediff ２つのdatetimeの差を表示

その他のformatter

topCalc 最上行を追加して、集計値を表示する

bottomCalc 最下行を追加して、集計値を表示する

例

{title:"Example", field:"example", topCalc:"count"}

値：avg、max、min、sum、concat、count

公式（formatter関連）： <https://tabulator.info/docs/5.5/format#format-builtin>

hozAlign

横方向の表示位置？

sorter

 ボタンによる並替えの時の基準となる値（フィールド）を指定する。

補足：通常特に指定の必要は無い。別の列の値をソートフィールドとして使いたい場合などに指定する。

###### リファレンス

table

setFilter

table.setFilter("name", "=", "Steve");

参考：http://tabulator.info/docs/5.2/filter

###### リファレンス（イベント）

<https://tabulator.info/docs/5.5/events>

rowClick

var tabledata = [

{id:1, name:"Oli Bob", progress:12, gender:"male", rating:1, col:"red", dob:"19/02/1984", car:1},

...

];

rowClick:function(e, row){ // 行がクリックされた時のコールバック

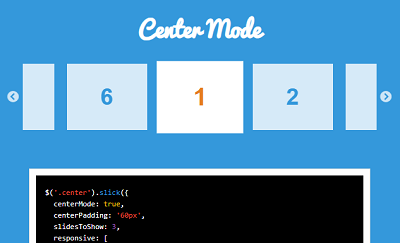
alert("Row " + row.getData().id + " Clicked!!!!"); // getData()で対象の行インスタンスが取得できる

},

#### スライドショー

##### slick

<https://kenwheeler.github.io/slick/>

複雑なスライドショーが実現可能。

##### bxSlider

<https://bxslider.com/>

例

<!DOCTYPE html>

<html lang="ja">

<head>

<title>スライドショーテスト</title>

<meta charset="UTF-8">

<link rel="stylesheet" href="https://cdn.jsdelivr.net/bxslider/4.2.12/jquery.bxslider.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>

<script src="https://cdn.jsdelivr.net/bxslider/4.2.12/jquery.bxslider.min.js"></script>

<script type="text/javascript">

$(document).ready(function(){

$('.slider').bxSlider({

auto: true,

pause: 5000,

});

});

</script>

</head>

<body>

<div class="slider">

<img src="media/images/img1.jpg" width="500" height="400" alt="">

<img src="media/images/img2.jpg" width="500" height="400" alt="">

<img src="media/images/img3.jpg" width="500" height="400" alt="">

<img src="media/images/img4.jpg" width="500" height="400" alt="">

</div>

</body>

</html>

##### FlexSlider2

<http://flexslider.woothemes.com/>

ちょっとポップすぎるかも

#### ガントチャート

##### jQuery.Gantt

デモサイト：<http://taitems.github.io/jQuery.Gantt/>

補足

moment.jsで日本語化

メモ

月、曜日の日本語化

$("#gantt").gantt({

dow: ["日","月","火","水","木","金","土",],

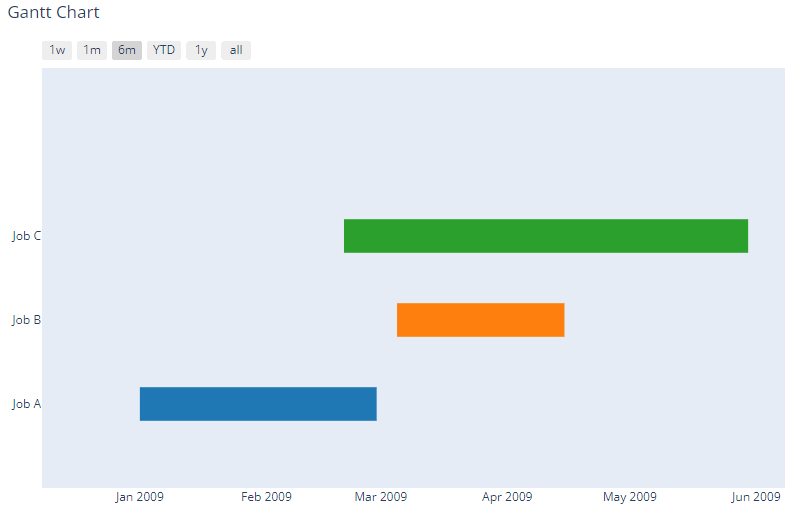
months : ["1月", "2月", "3月", "4月", "5月", "6月", "7月", "8月", "9月", "10月", "11月", "12月"],

});

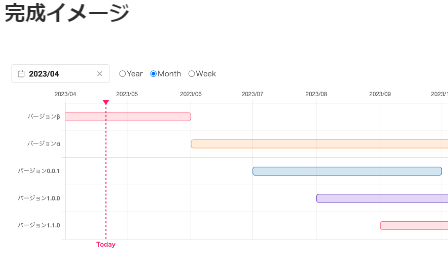
資料

時間の書式：<https://dygraphs.com/date-formats.html>

##### plotlyのpx.timeline

デモサイト：<https://plotly.com/python/gantt/#deprecated-figure-factory>

##### Chart.js

これでも出来るっぽいが資料が少ない

参考：<https://zenn.dev/tatausuru/articles/3650c5fb0d9110>

デモサイト：

<https://www.chartjs.org/docs/latest/samples/information.html>

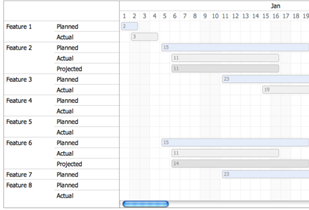
ガントチャートの作り方（動画）：<https://www.youtube.com/watch?v=InMnGwP7iX4>

##### Frappe Gantt

gantt.change\_view\_mode('Week')

// Quarter Day, Half Day, Day, Week, Month

マウスで伸び縮み出来る。⇒タスク管理に良い



##### jQuery.ganttView

マウスで伸び縮み出来る。⇒タスク管理に良い

タイムスケールの変更が直感的ではない

デモサイト：<https://github.com/namespace-ee/react-calendar-timeline/tree/master/examples>

##### React timeline calendar

ピンチイン・ピンチアウトでカレンダーの拡大縮小・横移動

##### Angular Gantt

マウスで伸び縮み出来る。⇒タスク管理に良い

タスク同士の依存関係も直感的に行える

<https://www.htmlelements.com/angular/ganttchart/>

##### rc-gantt

ドキュメントが中国語らしいのが難点

UIも抜群に良いというほどでもない為、保留。

<https://codesandbox.io/examples/package/rc-gantt>

##### Timeline View

[FullCalendar](https://fullcalendar.io/)のプラグイン（Timeline View）

かなり良さそうだが、有料（Premium Plugins；$480）

資料

<https://zenn.dev/2timesbottle/articles/4ca75466772711>

### 資料

#### 公式APIリファレンス

<https://api.jquery.com/>

## ajax

### 概要

Asynchronous JavaScript + XML

非同期処理を行う。通常、inputへの入力を受けて、少し遅れて画面を部分的に更新（レンダリングし直す）といった事を行う。

下の形が基本

$(function(){

$.ajax({

//実行する処理。

});

});

$(document).ready(function() {

$('#keyword').keyup(() => {

$.ajax({

type: 'GET',

url: '/gyoumu/api/modelcode/get/',

data:{

'item\_name' : $('#id\_item\_name').val(),

},

dataType: 'json'

}).done(function(data, textStatus, jqXHR) {

// 成功時の処理

console.log('done');

}).fail(function(jqXHR, textStatus, errorThrown) {

// 失敗時の処理

console.log('fail');

});

});

});

補足：success, errorを使う例もみられるが、比較的古い例らしい2022-04-28

Vanilla JavaScriptでの実行も可能ではあるらしい

var req = new XMLHttpRequest();

公式：<https://developer.mozilla.org/ja/docs/Web/API/XMLHttpRequest>

### 具体例

formを使う場合１

<form name="example\_form" method="post" action="">

<input type="text" name="name" value="" />

<button type="submit">適用</button>

</form>

$('#example\_form').submit(function(event) {

});

formを使う場合２

<form name="example\_form" method="post" action="">

<input type="text" name="name" value="" />

<button type="submit">適用</button>

</form>

$('#example\_form').on('submit', function(event) {

});

.on buttonイベントに連動？

<button class="onSend">ajaxで送信</button>

<script>

$(function () {

$(".onSend").on('click', function () {

}

### 書き方比較

$.ajax({

method: 'GET',

url: '/api/myapi',

}).done(response => { // 成功した時の処理

})).fail({ // 失敗した時の処理

})

こちらは古い書き方

$.ajax({

type: 'GET',

url: '/image',

success: function(response) { // response => {} と意味は同じ

},

})

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| バージョン | 成功 | 失敗 | 完了 |  |
| ～jQuery1.7 (非推奨) | success | error | complete |  |
| jQuery1.8～ | done | fail | always |  |

methodとtype：typeはjQuery1.8まで。1.9以降はmethodを使う。

## Node.js

### 概要

JavaScriptとほぼ同じ文法でサーバーを動かすためのしくみ。

偶数バージョンはLTS

JavaScriptの実行

node sample.js

nvmとnpmの違い

|  |  |
| --- | --- |
| nvm(Node Version Manager) | Node.js 自体をバージョン管理するツール |
| npm(Node Package Manager) | Node.js で作られたパッケージモジュールを管理するツール |

バージョン（抜粋）

|  |  |  |  |
| --- | --- | --- | --- |
| node | LTS | npm |  |
| 18.12.1 | Hydrogen | 8.19.2 |  |
| 10.24.1 | Dubnium | 6.14.12 |  |

公式のバージョン一覧：<https://nodejs.org/ja/download/releases/>

package.json

パッケージ管理をするテキストファイル。下記のようにするとパッケージをまとめてインストールできる。

# package.json の内容に基づきインストール

npm install

補足：pythonにおけるrequirements.txtと同等。

### インストール

#### 概要

大きく以下の方法がある

|  |  |  |
| --- | --- | --- |
| 方法 | 備考 | 備考／感想 |
| 公式スクリプト | 公式のダウンロードスクリプト | 何やってるか分からない怖さは多少ある |
| バイナリをダウンロード | 公式がコンパイルしたバイナリ | CPUが特殊（ARMなど）の場合 |
| nvm | 公式のバージョン管理ツール | 却ってややこしいような？ |
| apt | debian系パッケージ管理ツール | 一番手軽に見えて、実は一番厄介。**非推奨**。 |

以下のようなディレクトリ構成が一般的らしい

/usr/local/bin/

├ node

├ npm

インストール確認

node -v

#### 方法１　インストールスクリプト

curl -sL https://deb.nodesource.com/setup\_18.x | bash -

$ sudo apt install -y nodejs

公式：https://github.com/nodesource/distributions/blob/master/README.md

<https://github.com/nodesource/distributions/blob/master/README.md#debinstall>

参考：[https://www.ultra-noob.com/blog/2020/164/#](https://www.ultra-noob.com/blog/2020/164/#結論サクッとインストールするなら)結論サクッとインストールするなら

補足： 対応CPUはARMv7 以降なので、RaspberryPi2はこの方法ではインストールできない。

#### 方法２　nvm経由

公式Gitのインストールスクリプトを利用：<https://github.com/nvm-sh/nvm#installing-and-updating>

nvmのインストール

wget -qO- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.2/install.sh | bash

→terminalを再起動

# 利用可能なバージョンを確認

nvm ls-remote

nvm list available # Windowsの場合

# インストール

nvm install 18.12.1 # バージョン指定してインストール

nvm install 18.12.1 32 # 32bit版の場合

# 使用するバージョンの選択（バージョンが１つでもこれを行う必要がある）

nvm list # インストールされているバージョンの確認

nvm use 18.12.1 # バージョンの選択

# バージョン確認（インストール確認）

node -version # nodejsのバージョン -vでも可

npm --version # npm のバージョン -vでも可

参考：<https://zenn.dev/antez/articles/a9d9d12178b7b2>

補足：この方法だとnodeは~/.nvm/versions/node/{ver}/bin/node に配置される

#### 方法３　バイナリをダウンロード

以下公式サイトからバイナリを入手する

https://nodejs.org/ja/download/

ARMの場合はv7かv8かを確認；以下のコマンドどちらでも良い

cat /proc/version

uname -a

[ terminal ]（ホームディレクトリで実行しているものとする）

# ダウンロード

wget https://nodejs.org/dist/v18.12.1/node-v18.12.1-linux-arm64.tar.xz # 2022.12月現在最新

wget https://nodejs.org/download/release/v10.24.1/node-v10.24.1-linux-armv6l.tar.xz #armv6用

# 展開

tar Jxvf node-v18.12.1-linux-armv7l.tar.xz

# ファイルをコピー

sudo cp -r node-v18.12.1-linux-armv7l/\* /usr/local/

# 下記コマンドが成功したら成功

node -v

npm -v

補足：このままではsudo npm が実行できないため、対応が必要 2023-01-07

# /usr/local以外に配置した場合（非推奨）はパスを通す必要がある

echo 'export PATH=$HOME/node/bin:$PATH' >> ~/.profile

nodeディレクトリの配置先

|  |  |  |
| --- | --- | --- |
| nodeの配置先例 | 参考元 | 備考 |
| /usr/local/bin/ | 標準 |  |
| ~/.nvm/versions/node/{ver}/bin/ | nvmでのインストール | 要確認 |
| /usr/bin/ | aptでのインストール | 要確認 |

参考：https://zenn.dev/tmitsuoka0423/articles/nodejs-raspberry-pi-setup

参考：https://qiita.com/mininobu/items/c07f59a8210e8c71280e

補足： RaspberryPi2(ARMv6)用はv11が最終バージョンとなる <https://nodejs.org/dist/>

#### 方法４　apt 経由

一番手軽に見えて、実は一番微妙。2022-12-23時点での感想。

# まずはapt でインストールするが、バージョンが古い事が多い

sudo apt install -y nodejs npm

# npm のパッケージを管理する n というパッケージをインストール

sudo npm install n -g

# npm のインストール

sudo n lts # Long Term Service をインストール

# バージョン確認

node --version

npm --version

# apt でインストールした古いバージョンは削除

sudo apt purge -y nodejs npm # apt でインストールした古いバージョンは削除

LTSのバージョン確認

公式：<https://nodejs.org/ja/download/releases/>

補足：この方法だと/usr/bin/node にインストールされる

n を使った方法はPATHの修正が必要

参考：https://ytyaru.hatenablog.com/entry/2019/01/07/000000

参考：[https://qiita.com/gdate/items/f87af2e8491c6cc3c19e#nodejs](https://qiita.com/gdate/items/f87af2e8491c6cc3c19e#nodejsのインストール)のインストール

参考：<https://rikei-life.com/raspberry-pi-google-home-how-to-speak/>

### チュートリアル

mkdir nodejs\_test

cd nodejs\_test

npm init

package.json

### node

#### which

node のインストール先の確認

which node

#### コマンドライン引数

process.argv という配列で取得可能。

[ terminal ]

node /workspace/test.js hoge fuga

[ /workspace/test.js ]

console.log( process.argv[0] ) // node

console.log( process.argv[1] ) // /workspace/sample.js

console.log( process.argv[2] ) // hoge

console.log( process.argv[3] ) // fuga

### npm

Node Package Managerの略

インストール

Debian系Linux

sudo apt install npm

#### 基本

init package.jsonファイルが作成される

install

npm install パッケージ名@バージョン

installサブコマンドのオプション

--no-save package.jsonへの書き込みをしない

list

npmコマンドのオプション

-g グローバルインストール。コンピュータ全体で使う。マンドラインツールとして使用したいもの。

#### install

install package\_name

npm install express # パッケージ「express」をインストール

npm i express # i でも良いらしい

npm install express@4.16.3 # バージョン指定でインストール

npm install # package.json の内容に基づきインストール

オプション

-g グローバルインストール

-save package.json の dependencies に依存関係が追記

#### init

プロジェクトの初期化。package.jsonが作成される。

オプション

--yes yes/noの問いにすべてyesで回答する

公式：<https://docs.npmjs.com/cli/v8/commands/npm-init>

#### update

# npm 自身をアップデートする例

npm update -g npm

#### uninstall

またはun

パッケージの削除。

#### config

npm config list

#### audit

npm audit

npm audit fix

大体以下のメッセージが出た時に実行する

found 10 vulnerabilities (3 low, 3 moderate, 4 high)

実行後のメッセージ

fixed 0 of 10 vulnerabilities in 99 scanned packages

公式：https://docs.npmjs.com/cli/v6/commands/npm-audit

### REPL；Read-Eval-Print Loop

「node」と入力する事で開く、対話式の実行環境の事。

pythonに於いて「python」と入力して対話式実行環境が開くのと同じ

let x=10; // 行末の ; は有っても無くても良い

x+1;

.exit //終了

|  |  |  |
| --- | --- | --- |
| Ctrl | キー | 機能 |
| Ctrl+A | Home | 行前に移動 |
| Ctrl+E | End | 行末に移動 |
| Ctrl+P | ↑ | 前の実行式 |
| Ctrl+N | ↓ | 前の実行式 |
|  | TAB | コマンド補完 |

公式：<https://nodejs.org/api/readline.html#tty-keybindings>

環境変数

NODE\_REPL\_HISTORY =空白：有効、=空文字（’’）：無効 履歴機能の有効／無効

特殊メソッド

.clear コンソール表示をクリア

.exit REPLを終了

.help 特殊コマンドの簡易ヘルプの表示

.load <file\_name> ファイルからヒストリをよみだす

.save <file\_name> ファイルにヒストリを書き出す

### Node.js全体の注意点

・ApacheやNginxと異なり「ドキュメントルート」という概念は無い。

どちらかと言うと「スクリプト言語」としてとらえるべき。

#### ルーティング

##### expressのメソッドをそのまま使う

正規表現がつかえる（らしい）

app.get('/ab?cd', (req, res) => {}) // acd および abcd に一致します。

##### routes.jsを使う

#### htmlで実行

Node.js用の.jsファイルはそのままhtmlで読み込んでも実行できないらしい。

jsonwebtoken.js

### レファレンス

#### Promise

非同期処理を提供する。

### 標準パッケージ

#### fs

##### 概要

File Systemの意らしい

const fs = require('fs');

const file = fs.readFileSync( filepath + voiceFileName, 'binary');

doc：<https://nodejs.org/api/fs.html>

lib：<https://www.npmjs.com/package/fs>

##### existsSync()

書式：existsSync(path)

戻り値：boolean

同期でファイル有無を確認する。

<https://nodejs.org/api/fs.html#fsexistssyncpath>

##### mkdirSync()

### パッケージ（基本）

#### express

##### 概要

Webアプリケーション開発のフレームワーク

Node.jsで作成するサーバーサイドWebアプリケーションのデファクトスタンダード

const express = require('express')

const app = express()

app.get('/', function (req, res) {

res.send('Hello World')

})

app.listen(3000) // listen(port\_num)

:url\_parameter でurlパラメータを指定（？）

app.get('/voicetext/:voiceFileName', (req, res) => {

const voiceFileName = req.params.voiceFileName;

if (!voiceFileName) {

// ファイルが無かった時の処理

}

}

lib：<https://www.npmjs.com/package/express>

express公式：<https://expressjs.com/ja/4x/api.html>

参考：https://qiita.com/kamihork/items/ba7d86b4933fb2071a6b

##### all()

GET、POST、PUT、DELETEすべてのリクエストに対応するメソッド

req

req.ip

req.method

req.path

req.protocol

req.query.name

req.query パスパラメータから値を受け取る

req.params クエリ文字列(?arg=val)から値を受け取る

res HTTPレスポンスを返すオブジェクト

##### listen()

指定したポートをlistenし、サーバーを開始する。

var express = require('express')

var app = express()

app.listen(3000)

<http://expressjs.com/ja/api.html#app.listen_path_callback>

##### get()

基本：

var express = require('express');

var app = express();

app.get('/', func);

function func(req, res) {

res.send('Hello World')

})

#### forever

特定のスクリプトをバックグラウンドで常駐させる。

補足

あくまで簡易的なサービス化ととらえる。確実に起動して欲しい場合はsystemdなどを使った方が無難。

使い方

forever start sample.js # sample.js を常駐させる

forever list # 常駐しているスクリプトのリストを表示

forever log #

forever config #

オプション

-w .jsファイルの変更を自動検知

-l ログライルの指定

公式：<https://www.npmjs.com/package/forever>

#### repl

const repl = require('repl')

#### routes

https://www.npmjs.com/package/routes-js

#### n

インストール

sudo npm install n -g # -g はグローバルインストール；sudoが必要

基本的な使い方

sudo n stable

n version

sudo n 14.16.0

で使うらしい

#### node-gyp

C++を用いた拡張機能を作るためのビルドツール的なもの

ExpressとはNodeでのWebアプリケーションやシステムを開発するのに必要な機能が予め用意された骨組みで、

Express-GeneratorはExpressのアプリケーションを生成するツールです。

#### npx

ローカルインストールしたコマンドを実行する

例：

// nuxtのアプリの起動方法

npx create-nuxt-app <project-name>

// reactのアプリの起動方法

npx create-react-app <project-name>

// インストール済みモジュールの実行

$ npx <インストール済みモジュール名>

// インストールしていないモジュールを実行（実行後に自動削除）

$ npx <未インストールのモジュール名>

// GitHub上の特定リポジトリを明示的に指定して実行

$ npx github:<リポジトリ名>

### パッケージ（3rd party系）

#### castv2-client

公式

npmjs：<https://www.npmjs.com/package/castv2-client>

const Client = require('castv2-client').Client;

const DefaultMediaReceiver = require('castv2-client').DefaultMediaReceiver;

2022-12-24 メモ：ヒントになるかも

speechGoogleHome: function(host, url) {

const client = new Client();

client.connect(host, function () {

client.launch(DefaultMediaReceiver, function (err, player) {

if (err) {

console.log(err);

return;

}

player.on('status', function (status) {

console.log(`status broadcast playerState=${status.playerState} content=${url}`);

});

const media = {

contentId: url,

contentType: 'audio/mp3',

streamType: 'BUFFERED'

};

player.load(media, { autoplay: true }, function (err, status) {

client.close();

});

});

});

client.on('error', function (err) {

console.log(`Error: ${err.message}, host: ${host}, url: ${url}`);

client.close();

});

},

#### Prisma

Node.jsで標準となっているORM

公式：<https://www.prisma.io/>

TypeScriptでの使用も検討にいれた仕様になっているらしい。

#### swagger-node

Swagger（スワッガー）はREST APIを構築するためのオープンソースのフレームワーク

yaml形式でAPI定義を書く？

npmjs：<https://www.npmjs.com/package/swagger-node>

#### voicetext

HOYAの音声合成ソフトウェアのWebアプリ版

資料

WebAPI：https://cloud.voicetext.jp/webapi

公式Github：<https://github.com/pchw/node-voicetext>

npmjs：https://www.npmjs.com/package/voicetext

### nvm

# 利用可能なバージョンを確認

nvm ls-remote

nvm list available # Windowsの場合

# インストール

nvm install 18.12.1 # バージョン指定してインストール

nvm install 18.12.1 32 # 32bit版の場合

# 使用するバージョンの選択（バージョンが１つでもこれを行う必要がある）

nvm list # インストールされているバージョンの確認

nvm use 12.19.0 # バージョンの選択

参考：https://zenn.dev/antez/articles/a9d9d12178b7b2

公式Github <https://github.com/nvm-sh/nvm>

### 用語

グローバルインストール

ローカルインストール プロジェクトのルート配下にあるnode\_modulesにインストール

CORS

Cross-Origin Resource Sharing；オリジン間リソース共有；コーズ

http通信におけるデータの送信元としてのURIスキーム、ホスト名（ドメイン名）、ポート番号の組み合わせの事。

### 資料

nvmのGithub <https://github.com/nvm-sh/nvm>

公式リファレンス：<https://docs.npmjs.com/cli/v8>

npm コマンド（参考）：<https://www.sejuku.net/blog/75691>

React

## Vue.js

## react.js

View レイヤ(GUI) のJavaScriptのライブラリ。

SPA（Single Page Application）が作りやすい

React Nativeでスマホのネイティブで動くアプリも作成できる。

一昔前のjQueryのような感じ。

## Backbone.js

## Sails.js

MVCのWebアプリのフレームワーク

Railsライクなフレームワーク

npm install –g sails

メモ

prepend

# サーバー

## htmlサーバー

Apache

nginx

OpenLiteSpeed

lighttpd

## アプリケーションサーバー

gunicorn WSGIアプリサーバー。プロセス監視などができる

waitress WSGIアプリサーバー

## pythonンサーバー

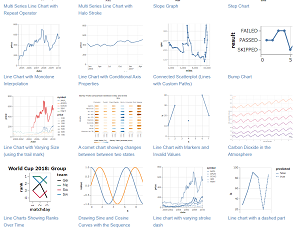
uvicorn

## データのレンダリング

# 可視化（グラフ）

## vega-lite

### 概要

Google Chartsに近いが、パーツの数がこちらの方が多い。

サンプルやチュートリアルも豊富。

フローチャートも描写できるらしい。

ライセンス：BSD

html：JavaScript、TypeScript

pythonでも使える（Pandas→Altair；アルタイル）。

⇒ 但し結局htmlサーバーで動く。

csvやjsonファイルも参照できる。

vscodeでも編集できるらしい。

Google Data Portalでも使用可能

（公式）<https://vega.github.io/vega-lite/examples/>

（参考）<https://qiita.com/torimochi/items/5523286b8e3993869d43>

（github）<https://github.com/vega/vega-lite>

⇒ vega/vaga/docsにデータセット有り

（オンラインエディタ）<https://vega.github.io/editor/#/edited>

マークダウンでも使える（らしい）

```vega-lite

{

"$schema": "https://vega.github.io/schema/vega-lite/v3.json",

...

}

```

デメリット

・右上のメニューボタンが邪魔

・英語しか資料がない。

・軸ラベルなどに日本語が使えない？

・使い方にちょっと癖がある

### 基本

#### html埋め込み

<!DOCTYPE html>

<html>

<head>

<script src="https://cdn.jsdelivr.net/npm/vega@5.22.1"></script>

<script src="https://cdn.jsdelivr.net/npm/vega-lite@5.2.0"></script>

<script src="https://cdn.jsdelivr.net/npm/vega-embed@6.20.8"></script>

</head>

<body>

<div id="vis"/> <!-- ここにグラフが表示される -->

<script>

const spec = {

"$schema": "https://vega.github.io/schema/vega-lite/v5.json",

"description": "A simple bar chart with embedded data.",

"data": {

"values": [

{"a": "A", "b": 28},

{"a": "I", "b": 52}

]

},

"mark": "bar", // グラフの種類？

"encoding": { // データの取り扱い方を指定

"x": {"field": "a", "type": "nominal", "axis": {"labelAngle": 0}},

"y": {"field": "b", "type": "quantitative"}

},

"config": {}

};

vegaEmbed("#vis", spec, {mode: "vega-lite"})

.then(console.log) <!-- JavaScript：htmlが読み込み終わったとき-->

.catch(console.warn); <!-- エラーが発生した場合 -->

</script>

</body>

</html>

#### html（csv）

const spec = {

"$schema": "https://vega.github.io/schema/vega-lite/v5.json",

"description": "A simple bar chart with embedded data.",

"data": {"url": "csv/test.csv"},

#### python（html版）

インストール

[ terminal ]

python3 -m pip install altair\_viewer # 実際には18余りのパッケージがインストールされる

[ altair.py ]

import altair as alt

import pandas as pd

source = pd.DataFrame({

'a': ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I'],

'b': [28, 55, 43, 91, 81, 53, 19, 87, 52]

})

diagram = alt.Chart(source).mark\_bar().encode(

x='a',

y='b'

).properties(

width=600,

height=500

).interactive()

diagram.show()

結局、localhostでhtmlサーバーが動く。

png化する方が使いやすい

requirements.txt（Python 3.10.4）

altair==4.2.0 altair-data-server==0.4.1 altair-viewer==0.4.0

attrs==21.4.0 entrypoints==0.4 Jinja2==3.1.2 jsonschema==4.6.0

MarkupSafe==2.1.1 numpy==1.22.4 pandas==1.4.2 portpicker==1.5.0

psutil==5.9.1 pyrsistent==0.18.1 python-dateutil==2.8.2 pytz==2022.1

six==1.16.0 toolz==0.11.2 tornado==6.1

#### python（png保存）

altar-saver というのを使うらしい

pip install altair\_saver

import altair\_saver as saver

import chromedriver\_binary

saver.save(chart, "chart.png", scale\_factor=2, method="selenium", webdriver="chrome")

# json 形式で保存

saver.save(chart, "chart.json")

### データセット

#### vega\_datasets

import altair as alt

from vega\_datasets import data

wheat = data.wheat()

chart = alt.Chart(wheat).mark\_point().encode(

y="wheat",

x="year:O"

)

chart

#### csv

#### json

import json

with open("chart.json") as f:

spec = json.load(f)

chart = alt.Chart.from\_dict(spec)

### タグ

#### $schema

読み込むvega-liteのバージョン

html上のjavascriptにおけるsrcみたいな感じ？

#### mark

円、四角など、基本的な描画要素を指定する。

|  |  |  |
| --- | --- | --- |
| 値 | グラフ | 備考 |
| bar | 棒グラフ |  |
| tick |  |  |
| point | 散布図？ |  |
| line | 線グラフ？ |  |
|  |  |  |
|  |  |  |
|  |  |  |

（公式）<https://vega.github.io/vega-lite/docs/mark.html>

#### data

データの指定

直接指定 json形式の文字列で直接記述する

"data": {

"values": [

{"a": "A", "b": 28},

{"a": "I", "b": 52}

]

},

url csvやjsonファイルなどを指定

"data": {"url": "data/csv/seattle-weather.csv"},

#### transform

"transform": [{"filter": "datum.symbol==='GOOG'"}],

#### encording

##### 概要

グラフのラベルなどの情報

##### field

dataで指定したデータの対象列の指定。

##### type

省略可能なパラメータなので

|  |  |  |
| --- | --- | --- |
| 値 | 概要 | 詳細 |
| nominal |  |  |
| quantitative | 数値的なデータ |  |
| temporal | 日時的データ |  |
|  |  |  |
|  |  |  |

（公式）<https://vega.github.io/vega-lite/docs/type.html>

##### axis

##### aggregate

集計情報

"encoding": {

"y": {"field": "a", "type": "nominal"},

"x": {

"aggregate": "average",

"field": "b",

"type": "quantitative",

"title": "Mean of b"

}

|  |  |  |
| --- | --- | --- |
| 値 | 概要 | 詳細 |
| average | 平均値 |  |
| mean |  |  |
| count | データ数 |  |
|  |  |  |
|  |  |  |

##### timeUnit

type=temporalの時の集計時間の単位？

"encoding": {

"x": {"timeUnit": "month", "field": "date"},

"y": {"aggregate": "mean", "field": "precipitation"}

}

##### title

"encoding": {

"x": {

"field": "weather",

"title" : "天気",

}

##### config

（公式）<https://vega.github.io/vega-lite/docs/config.html>

scale

"config": {

"scale": {

},

"range": {

},

}

## Google Charts

（公式チュートリアル）<https://developers.google.com/chart/interactive/docs/quick_start>

# SEO

## 概要

Search Engine Optimization 検索エンジン対策。

結局はGoogleの検索エンジン対策　と言う事になる。

## 基本

・キーワード

ユーザーが選びそうなキーワードを慎重に選び埋め込んでおく。

head内のtitle。

metaタブのdescription

metaタブのkeywords（descriptionよりは効果が薄いらしい）

hタグ；見出し

各見出しに適切なキーワードを埋める事を意識。結果的にみやすいページになるはず。

分かりやすく簡潔な文章にする。

imgのaltに使う文字列にもキーワードを埋める

・h1 タグは必須。但しページにつき1つだけ。

過去こんな事を言われていたようだが，現代の検索エンジンにはあまり効果がないらしい。

但し，ページ構造をしっかりさせるので上手に活用すれば見た目が良くなる。

現代で気にするならば：

⇒ タイトルに大きなフォントサイズを使用する

⇒ 各記事の冒頭に見出しを使う

⇒ 重要な部分は的確にハイライトする事

文字色を変えるのが一番多くみられる手法のように思う。

・ファビコンを使用する

厳密にいえばSEOには全く関係が無い（？）が，サイトのトレードマークとなるファビコンは必須。

・内部リンクの最適化

重要なページに内部リンクをあつめる

アンカーテキストにキーワードを入れる　？？

## robots.txt

どのURLにプログラムからアクセスしてよいかを定義する設定ファイル。

# SSL

Let’s Encrypt 無料でSSL証明書を発行する為のサイト

# 具体例

## TextBoxで候補を表示する方法

手法：htmlのみでdatalistを使う

補足：html5以降で有効

例：

<div class="container">

<div class="row">

<input type="text" list="items" class="text-1" />

<datalist id="items">

<option value="MacBook">

<option value="MacBook Air">

<option value="MacBook Pro">

<option value="iMac">

<option value="iMacPro">

<option value="MacPro">

<option value="Mac mini">

</datalist>

</div>

</div>

手法２：html+javaScript

event 「key up」を利用する

<div class="container">

手法３：Awesomplete

手法４：html+Ajax

<div class="container">

手法４：html+FetchAPI

memo：https://qiita.com/kanye\_\_east/items/9b419aac982c5323c429

djangoカスタムウィジェットで対応

<https://stackoverflow.com/questions/24783275/django-form-with-choices-but-also-with-freetext-option/32791625#32791625>

## ダウンロード

ディレクトリを指定 ⇒ ftpのような画面になり，ファイルをダウンロードできる。

逆に言うとディレクトリのダウンロードはできないので，zipなどのアーカイブにする必要がある。

<a href="bin/hoge.zip" download="archive.zip">hoge.zip</a>

# レンタルサーバー

マネージド型（共有サーバー）

・root権限無し

・保守は提供元が行ってくれる（逆に言うとユーザーは行えない）

・手軽だが，VPSより値段は少し高くなる。

・機能は限定的。特にデータベースは基本使えない。

仮想専用サーバー（VPS）

独自の

・各種プログラム言語（PHP、Perl、Ruby、 Python）などが使える

・サーバーの設定や運用は全てユーザーが行う

専用サーバー

・企業以外で使う事はほとんどない

・月額１万５千くらい～

各機能の説明

サブドメイン 1つのドメインを用途に応じて複数に分割する機能

独自ドメイン 端的に言うと，サイトの引っ越しの際も同じurlを引き継げる。

独自SSL 1つの証明書を１ドメインで使う事。共有SSLだと複数人で使う。

自作CGI Perlなどの独自スクリプトを書いて使用する事

データベース WordPressにはMySql (MariaDB)が必要

CMS高速化 WordPressなどで使うモジュール版PHPの事らしい。

マルチドメイン　文字通り，複数のドメインを使える＝複数のサイト運営が可能。

　但し通信量やリスク分散の観点で考えると，それほどお得感はない。

データベース数　これも文字通りだが，Webアプリ，ブログ事毎に１つのデータベースを使うイメージ。

有名なサーバー

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| ConoHa WING | 800円～4,700円 | 応答速度が速い。 | nginx |
| エックスサーバー | 720円～ | 長年人気のサーバー。 |  |
| Colorful Box | 640円～13,980円 | 高性能でありがなら安い。 | 30日お試し  アダルトサイト対応 |
| ロリポップ | 月100円～2,000円 | 初心者向け，サポート充実。サーバー性能はいまいち。  すべて国内サーバー。 | Apache  一部HDD  10日お試し |
| heteml |  |  | クリエイター向き。大容量。 |
| スターサーバー | 126円～4,000円 | 激安。サーバー性能はそこそこ。 | Nginx  オールSSD  ２週間お試し |
| さくら | 129円～4,628円 | 格安。もっとも歴史が長い。 |  |

## WordPress

必要なアプリは

Apache2 言わずとしれたWebサーバー

PHP PHPはサーバーサイドなので，サーバー側に実行環境が必要

MariaDB MySQLの派生として開発されたサーバー

WordPress 20.09.04 これはまだ良く分からない

サーバーを借りる時は上記に対応している必要がある。（WordPress対応！とか書いてあるが）

ラズパイなどにこれらを入れれば，少なくともテスト環境として使う事ができる

sudo apt -y install apache2 php php-mysql mariadb-server

## Html以外の用途例

メールサーバーにする

Webアプリを作る

データベースサーバーにする

# フレームワーク別メモ

## CMS

CMS；Contents Management System

専門的な知識が無くても、webサイトやコンテンツを構築・管理・更新できるシステムのこと。

### WordPress

無料版、パーソナル、プレミアム、ビジネスプランなどがある

Jimdo

ページ数少なめ

Drupal

大規模サイト向き

EC-CUBE

ネットショップ構築ツール

concrete5

Peraichi

Wix

## RubyOnRails

言語：Ruby

理念：「繰り返しを避ける」「設定より規約」

## Laravel

言語：PHP

理念：

## django

言語：python

理念：「暗黙より明示」「疎結合と高凝集性」

# 用語

パンくず

この様なもの →

「ヘンゼルとグレーテル」の迷子防止の為に置いたパンくずが由来だとか

DOM；Document Object Model

ドム。JavaScriptからHTMLを動的に変更することが可能になる技術。

Carousel

別名：カルーセル

複数のアイテムを水平にスクロール、または回転させるスライダーのような要素

RTLサポート

Right To Leftの意。アラビア語など右から左のテキストへの対応の事