最終更新　2025.1.31

# 基本技

## ボタン

　まずはボタンを作ってみましょう。次のhtmlファイルをエディタで作って、漢字コードをutf-8に設定してセーブしてください。

<!DOCTYPE html>

<html lang="ja">

<head>

<meta charset="utf-8">

<title>ボタン</title>

</head>

<body>

<p>あいうえお</p>

<button>押す</button>

</body>

</html>

　htmlファイルは <!DOCTYPE html> で始めます。<html> のタグの中にヘッダとボディがあります。<title> はタブに表示される文字列を指定します。<body> の部分の内容がブラウザに表示されます。

　実行してみると、「あいうえお」と書かれ、その下に "押す" と書かれたボタンが出現しました。ボタンを作るタグは <button> です。ボタンを押しても何も起こりません。ボタンを押したら「押した」と表示するようにプログラムを改造しましょう。

<!DOCTYPE html>

<html lang="ja">

<head>

<meta charset="utf-8">

<title>ボタン</title>

</head>

<body>

<p>あいうえお</p>

<button type="button" id="button-1">押す</button>

<script>

"use strict";

let btn1;

btn1 = document.getElementById("button-1");

btn1.addEventListener("click", btn\_click);

function btn\_click(){

 alert("押した");

}

</script>

</body>

</html>

　プログラムは <script> ～ </script> の中に記述します。プログラムの位置は body の末尾に配置します（headの中に書くことも可能ですが、少しややこしくなります）。htmlは上から順番に解釈されていくので、scriptの場所までに、プログラム中で利用するボタンなどの定義が存在する必要があります。以下がプログラムの解説です。

　"use strict"; を1行目に書いて、このプログラムは変数宣言を強要することを宣言します。

　let btn1; でbtn1という変数を使うことを宣言します。複数の変数を使うときは、let btn1, btn2, btn3; のようにコンマで区切ります。ここでの関数宣言はfunctionの外側で行っています。この場合letで宣言した変数はグローバル変数になります。

　操作したいオブジェクトにはidを付けます。<button type="button" id="button-1" で、ボタンに "button-1" というidを付けます。

　document.getElementById("button-1") で "button-1" というidを持つボタンのオブジェクトを取得し、btn1に代入します。

　btn1.addEventListenerでクリックしたときに呼ばれる関数名を指定します。

　ボタンを押すと、btn\_click() という関数が呼ばれます。関数は以下の形で記述します。

function 関数名(){

}

　alert("文字列") を実行すると、指定された文字列が書かれたダイアログが開きます。Excel VBAでMsgBox("文字列") と書くのと同じです。

## プログラムの実行とデバッグの方法

　プログラムをエディタで作成したら、拡張子をhtmlとしてファイルに保存します。

　そのhtmlファイルをブラウザにドラッグします。

　プログラムが動かないときはF12を押します。ここからはChromeにおける操作法を述べます。Consoleタブをクリックしてください。エラーが発生してプログラムが実行できない場合、 印とともにエラーの内容と、発生した箇所の行番号が示されています。file-name.html:71:33　のように表示されている場合、アンダーラインが引かれている文字列をクリックすると、該当箇所を表示します。この場合は71行目の33文字目に誤りがあります。

## 段落の内容の変更

　次に、ボタンを押したら、「あいうえお」の文字列が「かきくけこ」に変化するよう、プログラムを変更しましょう。

<!DOCTYPE html>

<html lang="ja">

<head>

<meta charset="utf-8">

<title>ボタンと段落</title>

</head>

<body>

<p id="paragraph-1">あいうえお</p>

<button type="button" id="button-1">押す</button>

<script>

"use strict"; // 変数の宣言を強要する

let btn1, p1;

btn1 = document.getElementById("button-1");

btn1.addEventListener("click", btn\_click);

p1 = document.getElementById("paragraph-1");

function btn\_click(){

 p1.innerHTML = "かきくけこ";

}

</script>

</body>

</html>

　idと変数名に同じ名前を付けるべきではありません。筆者は名前の衝突を避けるため、idにはハイフンを含む名前を付けることにしています。変数名にハイフンは使えないので（アンダースコアは使える）、変数名に同じ名前をつけることを回避できます。

　段落の中の文字を書き換えたいときは以下のように書きます。

p1.innerHTML = "書き換え後の文字列";

　間違ってinnerHtmlと書くと、何も起こらないので注意してください。この場合、innerHtmlというプロパティを追加し、値を設定します。JavaScriptは大変デバッグがしづらい言語ですが、その一つがこの「存在しないプロパティを使ってもエラーにならない」です。

### スクリプトの配置場所（発展）

　ここではスクリプトをbodyの末尾に記述しました。<head> に配置する方法もあります。以下のように書きます。

<!DOCTYPE html>

<html lang="ja">

<head>

<meta charset="utf-8">

<title>ボタンと段落</title>

<script>

"use strict"; // 変数の宣言を強要する

window.onload = init; // html が全てロードされた直後に init() を実行

let btn1, p1;

function init(){

 btn1 = document.getElementById("button-1");

 btn1.addEventListener("click", btn\_click);

 p1 = document.getElementById("paragraph-1");

}

function btn\_click(){

 p1.innerHTML = "かきくけこ";

}

</script>

</head>

<body>

<p id="paragraph-1">あいうえお</p>

<button type="button" id="button-1">押す</button>

</body>

</html>

　window.onload = initはhtmlファイルを全て読み取ったら、最初に function init() を実行するという意味です。init() の中に初期化処理を書いておきます。一般的にはスクリプトをbodyの末尾に記述することが推奨されるようです。この方がプログラムはシンプルになります。headに記述する方法は、「ファイルの先頭にプログラムがあった方が編集しやすい」「初期化の処理が明確に分離できる」というメリットがあります。

課題1

　bodyの色を変えるには以下のようにします。

<body id="body-1">

body = document.getElementById("body-1");

body.style.backgroundColor = "#ffc0c0";

　ボタンを2つ作り、1つめのボタンを押したらbodyの色を "#ffc0c0"、もう一つのボタンを押したら "#c0ffc0" にするページを作りなさい。

(補足)

　ボタンにイベントリスナを設定する方法として、以下のような方法もあります。シンプルですが、htmlの中にプログラム（スクリプト）を書くのと同等なので、情報の場所が分散し、プログラムは分かりにくくなります。

<button type="button" value="押す" onclick="btn\_click()">押す</button>

## 引数付きの関数を使う（発展）

　2つのボタンがあり、片方のボタンを押すと画面が薄黄色、もう片方のボタンを押すと画面が薄ピンクになるプログラムを作ってみます。

　ボタン毎に関数を作ると以下のようになります。

<!DOCTYPE html>

<html lang="ja">

<head>

<meta charset="utf-8">

<title>button and text</title>

</head>

<body id="body-1">

<button type="button" id="button-1">押す</button>

<button type="button" id="button-2">押す2</button>

<script>

"use strict"; // 変数の宣言を強要する

let body1, btn1, btn2;

btn1 = document.getElementById("button-1");

btn2 = document.getElementById("button-2");

body1 = document.getElementById("body-1")

btn1.addEventListener("click", change\_color1);

btn2.addEventListener("click", change\_color2);

function change\_color1(){

 body1.bgColor = "#ffff00";

}

function change\_color2(){

 body1.bgColor = "#00ffff";

}

</script>

</body>

</html>

　ここではボタン1個につき1個の関数を用意し、合計2個の関数を書きました。change\_color("#ffc0c0") のように、引数付きで呼び出せたら、関数は1個ですむので便利です。以下のように書くと関数を1個にまとめることができます。

　関数に引数を渡すには () => func\_name(arg) のように書きます。これはアロー関数と呼ばれるテクニックです。ここでは「このように書く」とだけ覚えて下さい。

<!DOCTYPE html>

<html lang="ja">

<head>

<meta charset="utf-8">

<title>button and text</title>

</head>

<body id="body-1">

<button type="button" id="button-1">押す</button>

<button type="button" id="button-2">押す2</button>

<script>

"use strict"; // 変数の宣言を強要する

let body1, btn1, btn2;

btn1 = document.getElementById("button-1");

btn2 = document.getElementById("button-2");

body1 = document.getElementById("body-1")

btn1.addEventListener("click", () => change\_color("#ffff00"));

btn2.addEventListener("click", () => change\_color("#00ffff"));

function change\_color(color){

 body1.bgColor = color;

}

</script>

</body>

</html>

## コンソールの使い方

　Excel VBAはイミディエイトウィンドウがあり、Debug.Print "abc" と書くとイミディエイトウィンドウに abc と書くことができました。同様の機能を持つのがconsole.log() です。F12を押して開くデバッグ用ウィンドウのConsoleタブに表示されます。console.logの引数は、表示したい文字列や変数を与えます。通常、変数をコンマで区切ると、半角空白1個を挟んで表示します。例えば、console.log("a =", a); と書くと、"a =" という文字列と、変数aの内容を半角空白1文字を挟んで表示します。

　JavaScriptのデバッグは、文法エラーについてはF12 → Consoleとしてエラー発生箇所を突き止めます。アルゴリズムの誤りなどをチェックするために、変数の内容を確認したい場合、console.log( ) を使います。

## テキストボックスの使い方

　テキストボックスの中の数値を処理するプログラム例を以下に示します。2つのテキストボックスに数値を入力し、「計算」ボタンを押すと、積を表示します。

<!DOCTYPE html>

<html lang="ja">

<head>

<meta charset="utf-8">

<title>html のひな形</title>

</head>

<body>

<p>

<input type="text" id="text-1" value="10"> ×

<input type="text" id="text-2" value="20">

</p>

<button type="button" id="button-1">計算</button>

<p id="result">

計算結果

</p>

<script>

"use strict";

let btn1, p1, t1, t2;

btn1 = document.getElementById("button-1");

t1 = document.getElementById("text-1");

t2 = document.getElementById("text-2");

p1 = document.getElementById("result");

btn1.addEventListener("click", btn\_click);

function btn\_click(){

 let a, b, c;

 a = parseInt(t1.value);

 b = parseInt(t2.value);

 c = a \* b;

 p1.innerHTML = "計算結果は " + String(c);

}

</script>

</body>

</html>

　テキストボックスは <input type="text" id="id名"> です。getElementById でテキストボックスオブジェクトを取得したなら、「オブジェクト名.value」で中に入っている文字列を取得したり書き換えたりできます。テキストボックスの中の文字列は "123" のように数字だけであったとしても、文字列です。計算するためにはparseIntあるいはparseFloatで数値に直してください。

　文字列の接続は + です。

　parseIntで数値に直すのを怠っても、自動的に型変換が行われます。また、p1.innerHTML = "計算結果は " + cとしても、自動的な型変換が行われ、エラーは発生しません。JavaScriptの自動型変換は便利な面もありますが、自動型変換がどのようなときに起こるのかを正確に把握しておかないと、発見しづらいバグを引き起こす可能性があります。筆者はpythonのように型に厳格である方を好みます。

　少し改造して、2つの数の最大公約数を求めるプログラムは以下のようになります。無限ループはwhile(true){} でループからの脱出はbreakです。

<!DOCTYPE html>

<html lang="ja">

<head>

<meta charset="utf-8">

<title>button and text</title>

</head>

<body>

<p>最大公約数の計算</p>

<p>数値1：<input type="text" id="text-1" value="200"></p>

<p>数値2：<input type="text" id="text-2" value="124"></p>

<button type="button" id="button-1">計算</button>

<p id="paragraph-1">ここに計算結果を表示します。</p>

<script>

"use strict"; // 変数の宣言を強要する

let t1, t2, b1, p1, a, b, gcm;

t1 = document.getElementById("text-1");

t2 = document.getElementById("text-2");

b1 = document.getElementById("button-1");

b1.addEventListener("click", calc\_gcm);

p1 = document.getElementById("paragraph-1");

function calc\_gcm(){

 let tmp, c;

 a = parseInt(t1.value);

 b = parseInt(t2.value);

 if ( b > a ){ // a > b となるよう入れ替える

 tmp = a;

 a = b;

 b = tmp;

 }

 while(true){

 console.log(String(a) + " " + String(b))

 c = a % b;

 if ( c == 0 ) break;

 a = b;

 b = c;

 }

 p1.innerHTML = "最大公約数は " + String(b)

}

</script>

</body>

</html>

## getElementByIdのエラーに対する対処

　ここまでで、ボタン、段落、テキストボックスなどのオブジェクトをgetElementByIdで取得してきました。一つ重大な落とし穴があります。

a = getElementById("id-name");

で存在しないid-nameを指定した場合、エラーは発生せず、aにnull（何もないことを表す特別な値）が入ります。そして、a.addEventListenerあるいはa.innerHTML, a.valueを使用した段階でエラーが発生します。初心者は「なぜ、エラーが発生するのか分からない」と悩むかもしれません。getElementByIdの段階でエラーを検知する方法として、以下のような関数を用意する方法が考えられます。

p1 = getElementSafe("paragraph-11");

(略)

function getElementSafe(id) {

 let element = document.getElementById(id);

 if ( element == null ) {

 alert("Error: Element with ID " + id + " is not found.");

 }

 return element;

}

　この関数を使ってボタンや段落のオブジェクトを取得するなら、nullが得られた段階でエラーを検知できます。

　ただし、多くのJavaScriptのコードではこのような関数は使わずに、getElementByIdを直接呼んできます。以下の理由によるものと思われます。

(1) getElementSafeを使うとコードが冗長になる

(2) 存在しないidを指定するような初歩的なミスは犯さない

(3) 初心者でないならa.innerHTMLなどにおけるエラーに対して、すぐに「IDがない」ことに気がつく。

## キーボードとマウスのイベント

<!DOCTYPE html>

<html lang="ja">

<head>

<meta charset="utf-8">

<title>キーボードイベントとマウスイベント</title>

</head>

<body>

<p>イベントについて学習します。キーボードを押すかマウスを左クリックして下さい。</p>

<p id="paragraph-1">キーボードに関する情報</p>

<p id="paragraph-2">マウスのクリックに関する情報</p>

<p id="paragraph-3">マウスの位置に関する情報</p>

<script>

"use strict";

let p1, p2, p3;

p1 = document.getElementById("paragraph-1");

p2 = document.getElementById("paragraph-2");

p3 = document.getElementById("paragraph-3");

window.addEventListener("keydown",key\_manage);

window.addEventListener("click",click\_manage);

window.addEventListener("mousemove",move\_manage);

function key\_manage(evt){

 let code = evt.keyCode;

 p1.innerHTML = "keydown keycode = " + String(code);

}

function click\_manage(evt){

 let x, y;

 x = evt.screenX;

 y = evt.screenY;

 p2.innerHTML = "click at (x,y): " + x + " " + y;

}

function move\_manage(evt){

 let x, y;

 x = evt.screenX;

 y = evt.screenY;

 p3.innerHTML = "move at (x,y): " + x + " " + y;

}

</script>

</body>

</html>

　キーボードのイベントとマウスのイベントを取得して、取得した情報を表示するサンプルです。ゲームなどのリアルタイム処理に生かせそうです。