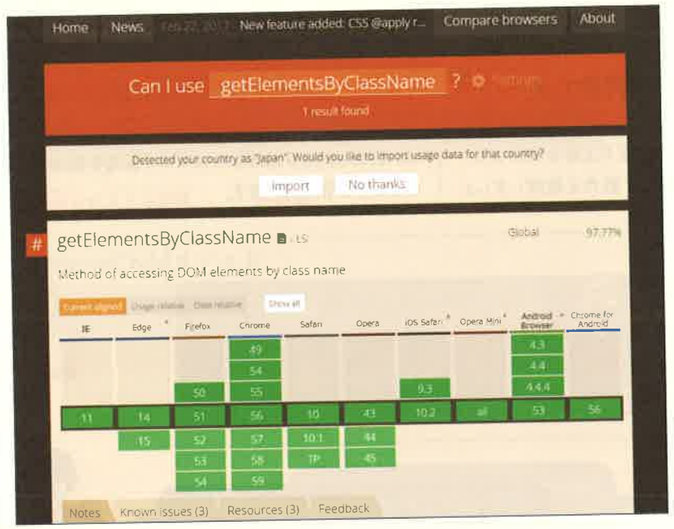
ワンポイント ブラウザごとにDOMに微妙な違いがある

実は、DOMはJavaScriptの言語仕様として決められたものではなく、ブラウザによって提供される仕組みです。そのため、ブラウザによって一部のメソッドやプロパティが利用できなかったり、挙動が違ったりする場合があります。本書では、Chromeをはじめとした主要なブラウザで利用できるものを厳選して学んでいきます。
そのため、HTML/CSSを操作するプログラムを書くと、ブラウザによってはうまく動かないというケースが出てきます。
でも、ブラウザの違いを調べながらプログラムを書くのは大変ですよね。そんなときに便利なのが、ブラウザ間の違いを修正してくれる「ライブラリ」です。ライブラリとは、汎用性の高いプログラムを再利用しやすくまとめたもので、中にはブラウザ間の違いを修正してくれるものもあります。本書でも、ブラウザ間の違いを吸収しつつ、便利な機能を提供してくれるライブラリ「jQuery」を扱います。
jQueryはGoogleやYahoo! などのサイトでも利用され、JavaScriptを扱う人なら必ず知っているといってもいいほど人気の高いライブラリです。
この章ではライブラリに頼らずHTML/CSSを操作する方法を学びますが、複雑な操作を行う際は「jQuery」などのライブラリを使用するほうが、ブラウザごとの違いに対応することもでき、より一般的です。どちらも重要な知識なので、後のChapter 10では、jQueryを利用したHTML/CSSの操作方法も学んでいきます。

▶ Can I use...



http://caniuse.com
JavaScriptのメソッドやCSSプロパティのブラウザごとの対応状況を確認できるWebサイト。

Chapter 7

ユーザーの操作に対応させよう

この章では、ユーザーの操作にあわせてプログラムを実行するイベント処理について学んでいきます。操作に応じて処理を切り替えることで、より実用的なプログラムを記述することができるようになります。

Lesson 40 [イベントの概要]

イベントとは何かを知りましょう

このレッスンのポイント

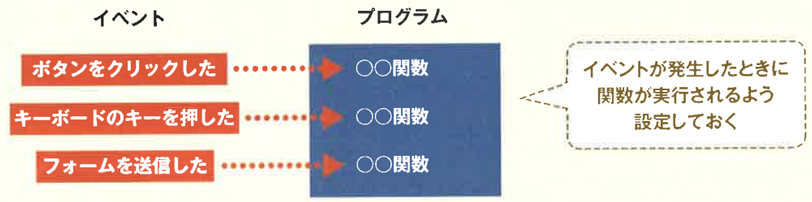
「クリック」や「キー入力」など、プログラムが動作するきっかけとなるできごとを、プログラミングの用語で「イベント」といいます。「イベント」の仕組みを使うと、ユーザーの操作に応じて動作する実用的なプログラムを記述できます。

ユーザーの操作に応じてプログラムを動かす仕組み

これまでは、HTMLファイルを読み込むとすぐに実行されるプログラムを記述してきましたが、Webページで動作するプログラムの多くは、「ボタンをクリックする」「キーボードのキーを押す」などのユーザーの操作に対応して結果を出します。

このようなプログラムを動かすキッカケとなるできごとを、プログラミングの用語で「イベント」といいます。JavaScriptでは、このイベントに対応して実行される関数を指定しておくことで、ユーザーが操作できるWebページを実現しています。

▶ イベントの働き



イベントの設定方法はイベントリスナーが主流

イベント発生時に動作するプログラムを指定する方法は、大きく以下の3つの方法があります。
ただ、「イベントリスナーで指定する」方法以外は、「1要素・1イベントにつき1つの設定」しか行うことができないため、最近では複数の設定を行える「イベントリスナーで指定する」方法が主流となっています。本書ではこの方法を主に扱い、他の2種類についてはコラムで紹介します。

- **イベントリスナーで指定する**
- **要素のプロパティで指定する（イベントハンドラ方式）**
- **HTMLファイルで要素の属性に指定する（イベントハンドラ方式）**

イベントリスナーによるイベントの設定

イベントにプログラムを関連付けるには、そのイベントに対して「イベントリスナー」を追加します。イベントリスナーとは、文字通り「イベント発生の聞き手」のことで、イベントが発生したときにそれを聞きつけて、あらかじめ指定しておいた関数を実行してくれます。
なお、関数を名前で指定する場合、その関数に引数を指定することができません。そのため、無名関数を使って行いたい処理を指定する方法がより一般的になっています。
イベントで名前のある関数に引数を渡したい場合は、イベントリスナーとして引数付きの無名関数を登録し、その無名関数の中で名前のある関数を呼び出して引数を渡すという手間がかかる書き方になります。名前のある関数も、無名関数の中で呼び出せば、引数を指定することができます。

▶ addEventListenerメソッドで名前のある関数を指定

```
element.addEventListener('click', func);
```

対象の要素　イベントタイプ名　実行したい関数名

▶ addEventListenerメソッドで無名関数を指定

対象の要素　イベントタイプ名　無名関数

```
element.addEventListener('click', function(……){
  行いたい処理
});
```

イベントの種類

JavaScriptでプログラムに利用できるイベントの種類を以下にまとめました。表中の「イベントタイプ名」をaddEventListenerメソッドの第1引数に指定します。

▶ マウス操作

イベントタイプ名	発生タイミング
click	要素をクリックしたとき
dblclick	要素をダブルクリックしたとき
mouseout	マウスポインタが要素上から出たとき
mouseover	マウスポインタが要素上に乗ったとき
mouseup	マウスボタンを放したとき
mousedown	マウスボタンを押し下げたとき
mousemove	マウスを動かしている間

▶ キーボード操作

イベントタイプ名	発生タイミング
keyup	キーを離したとき
keydown	キーを押したとき
keypress	キーを押し続けている間

▶ その他

イベントタイプ名	発生タイミング
blur	フォーカスが外れたとき
focus	フォーカスが当たったとき
change	内容が変更されたとき
select	テキストが選択されたとき
submit	フォームを送信しようとしたとき
reset	フォームがリセットされたとき
abort	画像の読み込みを中断したとき
error	画像の読み込み中にエラーが発生したとき
load	ページや画像の読み込みが完了したとき
unload	アンロード時（ページ遷移時など）

ワンポイント イベントハンドラを用いたイベントの設定

イベントの設定には、イベントリスナーを用いる以外にも、「イベントハンドラ」を用いた方法があります。イベントハンドラとは、特定のイベントが発生した際に実行される処理と、その登録方法のことで、イベントリスナーと異なり1つのイベントに複数の設定をすることができません。まだまだ使用されていることの多い方式なので、プログラムを読めるようにしておきましょう。

イベントハンドラでの設定方法はさらに2通りに分かれており、「要素のプロパティに指定する」方法と「HTMLファイルで要素の属性に直接記述する」方法があります。HTMLファイルに直接記述する場合は、関数名だけでなく「();」まで書きます。なお、イベントハンドラ名は、基本的にイベントタイプ名の先頭に「on」を付けたものになります。

▶ 要素のプロパティを利用して指定する方法

```
element.onclick = elementclick;
```

対象の要素　イベントハンドラ名　実行したい関数名

▶ HTMLファイルで要素の属性に指定する直接記述する方法

```
<button id="test" onclick="elementclick();">○○○○</button>
```

イベントハンドラ名　関数の呼び出し

これまでイベントを設定する3つの方法を見てきましたが、複数の方法を併用するとプログラムがわかりにくくなるので、特に理由がなければ、一番柔軟性の高いイベントリスナーを使用しましょう。

Lesson 41 [イベント:click]

クリックイベントでお問い合わせフォームを表示しましょう

このレッスンのポイント

イベントの概要は理解できましたか？ ここからはイベントを活用した「お問い合わせフォーム」を作りながら、イベントの設定方法を具体的に学んでいきましょう。ボタンをクリックするとJavaScriptのプログラムが働いて、フォームが表示されるようにします。

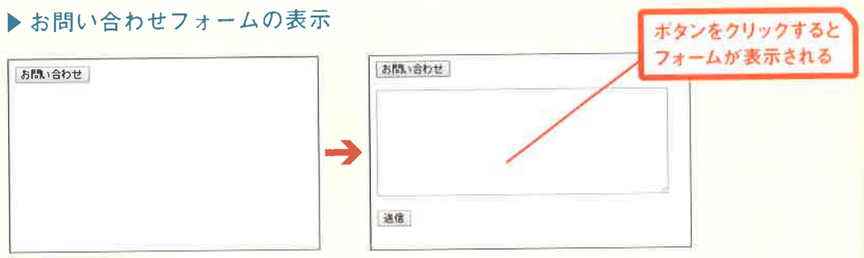
イベントを利用した「お問い合わせフォーム」を作る

今回はイベント利用の実践として、ボタンをクリックすると表示される「お問い合わせフォーム」を作ってみましょう。今回はフォームに使用していますが、いろいろと応用が利く方式です。

JavaScriptで要素を後から表示させる場合、JavaScriptで新しい要素を作って挿入する方法と、要素自体はあらかじめHTMLで用意しておいてJavaScriptで表示／非表示のみを切り替える方法があります。今回は後者の方式で表示します。

また、ボタンがクリックされたことを検出するためにclickイベントを使用します。使い方が比較的シンプルで、非常によく使うイベントです。

▶ お問い合わせフォームの表示



クリックで表示される「お問い合わせフォーム」を作る

1 HTMLファイルを編集する 07/from/practice/index.html

まずは、HTMLファイルを編集してボタンとフォームを作成しましょう。

このレッスンのindex.htmlファイルをBracketsで開いて、button要素❶とform要素❷を追加し、上書き保存してください。Webブラウザで開くと、ボタンとフォームが表示されます。

```
008 <body>
009 __<button_id="button">お問い合わせ</button>
010 __<form_action=""_id="form">
011 ____<p><textarea_id="textarea"_name="textarea"_cols=50_rows=10_
    maxlength="500"></textarea></p>
012 ____<p><input_id="submit"_type="submit"_value="送信"></p>
013 __</form>
014 __<script_src="js/app.js"></script>
```

❶ button要素を追加

❷ form要素を追加

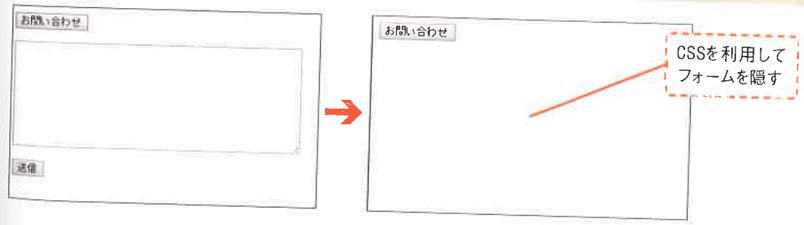
2 CSSファイルを編集する 07/from/practice/css/style.css

次に、CSSファイルを編集して、フォームを非表示にしてしまいましょう。 ボタンを押した際にJavaScriptでCSSを「表示」に書き替えることで、ボタンをクリックすると表示される機能を作ることができます。

このレッスンのstyle.cssファイルをBracketsで開き、フォームを非表示にするCSSを記述して上書き保存してください❶❷。

```
001 #form_{
002 __display:_none;
003 }
```

❶ フォームを非表示に



NEXT PAGE

3 JavaScriptファイルを編集する 07/from/practice/js/app.js

続いて、JavaScriptでボタンがクリックされたときにフォームを表示する機能を作成します。このレッスンのapp.jsファイルをBracketsで開いて、以下のコードを記述し上書き保存してください。

まず「getElementById()」メソッドを使って、ボタンの要素とフォームの要素を取得します❶。次に、ボタン要素に対して「addEventListener()」メソッドを使って、イベントリスナーを登録します❷。第1引数にはイベントタイプを示す文字列「'click'」、第2引数には無名関数「function() { ... }」を指定して、「{}」の中でフォームを表示するための、スタイル変更の処理を記述しています❸。

```
/*_プログラムで使用する変数の設定_***************/
//_フォームの要素を取得
var_button_=_document.getElementById('button');
var_form_=_document.getElementById('form');

/*_イベント処理_***************/
//_お問い合わせボタンを押したとき
button.addEventListener('click',_function()_{
__//_フォームを表示
__form.style.display_=_'block';
});
```

❶ 要素を取得
❷ イベントリスナーを登録
❸ スタイルを変更

Point 無名関数を利用する

ここではaddEventListenerメソッドの引数に、名前を持たない「無名関数」を使用しています。以下に示すように名前のある関数を使用することもできますが、無名関数を使うことがよくあるので慣れておきましょう。

▶ 名前がある関数を使った場合

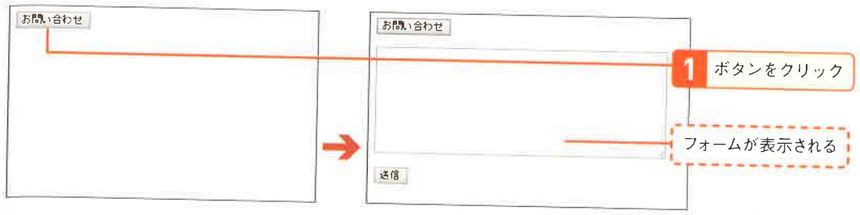
```
button.addEventListener('click', showForm);

function showForm(){
__form.style.display_=_'block';
}
```


4 プログラムが完成した

プログラムが完成したら、内容を上書き保存して、index.htmlをブラウザで開いて動作を確認しましょう❶。

無事にフォームは表示されましたか？ click以外の他のイベントタイプでも、イベントと関数をひも付ける方法は同じです。



ワンポイント 複数のイベントが同時に起こることもある

今回は「click」のイベントを用いましたが、イベントタイプの中には「mouseup」や「mousedown」といった、clickと似たようなイベントが用意されています。

mousedownはマウスボタンを押したとき、mouseupはマウスボタンを離したとき、そしてclickは、mousedownとmouseupが連続して起きたときに発生します。つまり、clickが発生する際は、mousedownとmouseupのイベントも同時に発生しているのです。

イベントタイプ名	発生タイミング
click	要素をクリックしたとき（mousedown + mouseup）
mouseup	マウスボタンを離したとき
mousedown	マウスボタンを押したとき

Lesson 42 [イベント:keyup]

フォームに残り文字数のカウント機能を付けましょう

このレッスンのポイント

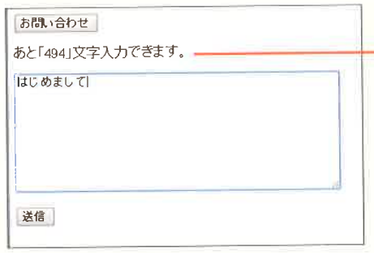
入力中に「後○○文字」と表示される機能を見たことはありませんか？今回はその機能を作成してみましょう。文字入力に関するイベントなので、キーボードのキーを押して放したときに発生する「keyup」のイベントを利用します。

残り文字数のカウント機能の概要

今回作成する「残り文字数のカウント機能」は以下の図のようなイメージです。文字入力を行うと、残り文字数をカウントしてリアルタイムで画面に表示してくれます。

今回のプログラムでは、文字入力が行われるたびに表示が切り替わるので、キー入力に関するイベントと残り文字数を表示する処理を関連付ける必要がありそうです。

▶ 完成イメージ



残り文字数が表示される

利用できるイベントをたくさん知っておけば、ユーザーに対して細やかに応答するプログラムを作れるようになります。

keydown、keypress、keyupの使い分け

キー入力に関するイベントタイプは3種類で、kwydownはキーを押したとき、keypressはキーを押し続けている間、keyupはキーを離したときにそれぞれ発生します。keypressは少しわかりづらいですが、テキストエディタでキーを押しっぱなしにすると、同じキーが何度も入力されるイメージと同じで、最初の1回＋押した時間に応じてイベントが何度も発生します。今回作成するプログラムでは、文字が入力された後に文字数をカウントしたいので、文字入力が終わった後に発生する「keyup」イベントを利用するのがよさそうです。

イベントタイプ名	発生タイミング
keyup	キーを離したとき
keydown	キーを押したとき
keypress	キーを押し続けている間

イベントとプログラムを関連付ける際は、発生タイミングが最も適切なものを選びましょう。

押されたキーを調べられる「イベントオブジェクト」

keyイベントでは、押されたキーが何だったのか知りたい場面も多いと思います。
そんなときに利用できるのが、イベントに関する情報がまとめられた「イベントオブジェクト」です。
イベントオブジェクトは、イベントリスナーで指定した関数の第1引数に自動で受け渡されます。

イベントオブジェクトを使用したい際は、その代入先となる引数（下記ではevent）を指定するだけでOKです。押されたキーの情報はkeyプロパティに格納されているので、以下のように確認することができます。

▶ 例文:押されたキーを取得する

イベントオブジェクトが引数に渡される

```
document.addEventListener('keydown',_function(event)_{
__console.log(event.key);_//_コンソールにキーを表示
});
```

イベントオブジェクトを利用すると、発生したイベントに関するさまざまな情報を取得することができます。

1 JavaScriptファイルを編集する 07/from/practice/js/app.js

残り文字数を表示するための機能をJavaScriptで作成していきます。このレッスンのapp.jsファイルをBracketsで開いて、以下のコードを記述し上書き保存してください。

まず、文字数をカウントしたいtextareaの要素を取得します❶。次に入力可能な最大文字数を調べたいので、getAttributeメソッドを使用してmaxlength属性の値を取得します❷。

```
002 //_フォームの要素を取得
003 var_button_=_document.getElementById('button');
004 var_form_=_document.getElementById('form');
005 var_textarea_=_document.getElementById('textarea');
006
007 //_文字数制限
008 var_maxTextNum_=_textarea.getAttribute('maxlength');
```

1 要素を取得

2 属性値を調べる

Point getAttributeメソッドで属性を取得

getAttributeメソッドを使用すると、その要素の任意の属性値を取得することができます。ここではtextarea要素のmaxlength属性の値を取得して、入力可能な文字数を調べるために使っています。

▶ getAttributeメソッド

```
element.getAttribute('maxlength');
```

element：要素　'maxlength'：属性名

```
<textarea_id="textarea"_name="textarea"_cols=50_rows=10
maxlength="500">
```

HTML

この属性値を取得

2 残り文字数を表示するための要素を追加する

続いて、残り文字数を表示するためのdiv要素を新たに作り、textareaの前に設置していきます。今回はJavaScriptを使って生成します❶。

```
009
010 /*_要素の追加_***************/
011 //_残り文字数を表示する要素の追加
012 var_textMessage_=_document.createElement('div');
013 var_parent_=_textarea.parentElement;
014 parent.insertBefore(textMessage,_textarea);
```

1 要素を作成して追加

3 残り入力文字数を表示する

残り文字数はキー入力とともに変化していくので、今回はキーを押して離したときに発生するkeyupイベントを使用し、addEventListenerメソッドでイベントリスナーを追加します❶。無名関数内でまず現在入力されている文字数を取得し❷、それを元に何文字入力可能かを算出します。textMessage要素のinnerHTMLプロパティに代入して文字数を更新します❸。

```
022
023 //_テキストエリアでキーをタイプしたとき
024 textarea.addEventListener('keyup',_function()_{
025 __var_currentTextNum_=_textarea.value.length;
026 __textMessage.innerHTML_=_'<p>あと「'_+_(maxTextNum_-_currentTextNum)
    +_'」_文字入力できます。</p>';
027 });
```

1 イベントリスナーを登録

2 文字数を調べる

3 残り文字数を表示

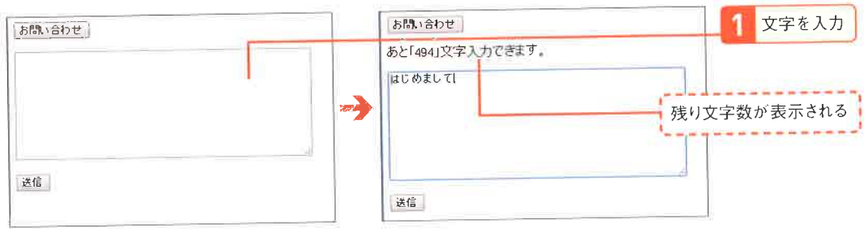
Point テキストエリアの入力済み文字数を調べる

ここでは入力済み文字数を調べるために「textarea.value.length」という書き方をしています。valueプロパティはテキストエリアに入力された文字列を取得する働きを持ち、さらにlengthプロパティ（P.156参照）を使って長さを調べています。

NEXT PAGE

4 プログラムの動作を確認する

ここまで入力したらファイルを上書き保存し、index.htmlをブラウザで開いて動作テストしましょう。テキストエリアに文字を入力すると、キーを押すたびにイベントが発生し、残り文字数の表示が更新されます。



ワンポイント Stringオブジェクトを利用する

前ページで文字数を数えるときに使った「length」は「Stringオブジェクト」のプロパティです。
Stringオブジェクトとは、文字列に関するオブジェクトのことで、文字の長さを知ることができるlengthプロパティの他に、文字の前後の空白を消すtrimメソッド、検索置換を行うことができるreplaceメソッドなど、さまざまなプロパティが用意されています。
実はJavaScriptでは、文字列型のデータでも、Stringオブジェクトが持つプロパティをそのまま利用できるようになっています。例えば以下の例文のように、「"あいうえお".length」というような記述で、文字数を数えることもできます。

▶ 例文：lengthプロパティの利用

```
'あいうえお'.length;  // 5
```

Stringオブジェクトについて詳しく知りたい人は、Chapter 13で紹介する「MDN」などのリファレンスを利用してみてください。

Lesson 43 [タイマー処理]

フォームを時間制限付きの回答フォームに改造しましょう

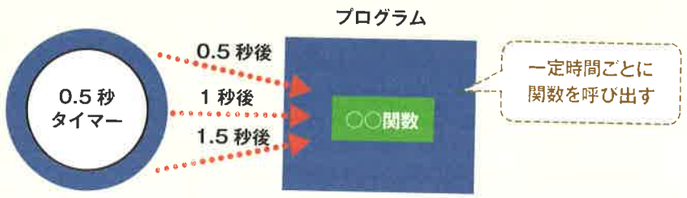
このレッスンのポイント

イベント以外にプログラミングの実行タイミングを指定できるものとして「タイマー」があります。タイマーを使うと、一定時間後にプログラムを動かしたり、一定間隔でプログラムを実行したりすることができます。

一定間隔で繰り返す処理

プログラムの中には、一定間隔で自動的に繰り返す処理がたくさん使われています。身近な例では、時刻などの表示です。
HTML/CSSは一度読み込むと原則的に変化しないので、時計の時刻の用に常に変化するものは、一定間隔で最新の値を「表示する」処理を繰り返す必要があります。また、スライドショーなどのアニメーションの処理も「パラパラ漫画」と同じ原理で、一定間隔で「要素を移動する」処理を繰り返して実現しています。

▶ タイマーがイベントを呼び出す



一定間隔で繰り返す処理は、アニメーションなどの動きのある処理によく使われます。

setIntervalメソッドの使い方

JavaScriptで一定間隔で処理を繰り返すには、setIntervalメソッドを使います。

setIntervalメソッドの戻り値には、セットしたタイマーを解除するためのIDが発行されるので、通常は使用開始と同時に「タイマー識別用の変数」にその値を記録しておきます。第1引数には実行したい処理、第2引数には実行したい間隔をミリ秒で指定します。タイマーを解除する際はclearIntervalメソッドの引数に、記憶しておいた「タイマー識別用の変数」を指定すればOKです。

▶ setIntervalメソッド

```
var timer = setInterval(timerfunc, 200);
```

タイマー識別用の変数　実行したい関数　間隔

▶ clearIntervalメソッド

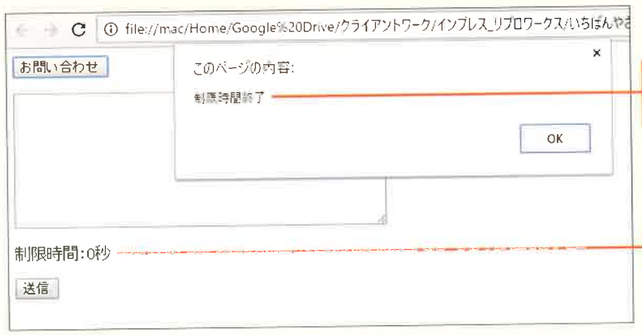
```
clearInterval(timer);
```

タイマー識別用の変数

setIntervalメソッドで処理を開始するときは、同時にclearIntervalメソッドでタイマーを止める必要があるか、忘れずに検討しましょう。

お問い合わせフォームを改造しよう

タイマー処理の実践として、先のレッスンで作成したお問い合わせフォームを時間制限付きの回答フォームに改造してみましょう。今回は、タイマー処理を使って制限時間を表示し、制限時間になると、制限時間切れと表示する仕組みを作ってみましょう。



時間制限付きの回答フォームに改造する

1 JavaScriptファイルを編集する　07/from/practice/js/app.js

まずは、入力の残り時間の設定や、それを表示する要素の準備を行いましょう。このレッスンのapp.jsファイルをBracketsで開いて、以下のコードを記述してください。

まず、残り時間を変数で設定します❶。変数で設定することで、後で時間の変更が簡単になります。今回は、結果がすぐ確認できるように10秒と設定しておきましょう。次に、残り時間を表示する要素を追加します❷。

```
007 //_文字数制限
008 var_maxTextNum_=_textarea.getAttribute('maxlength');
009 //_残り時間（秒）
010 var_reminingTimeNum_=_10;
011
012 /*_要素の追加_***************/
013 //_残り文字数を表示する要素の追加
014 var_textMessage_=_document.createElement('div');
015 var_parent_=_textarea.parentElement;
016 parent.insertBefore(textMessage,_textarea);
017
018 //_残り時間を表示する要素の追加
019 var_timeMessage_=_document.createElement('div');
020 parent.insertBefore(timeMessage,_null);
```

1 変数を定義

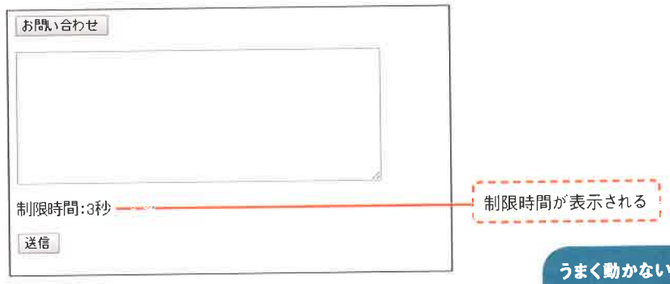
2 要素を作成して追加

2 タイマー処理で時間を表示する

フォームが表示されてからカウントダウンを開始したいので、ボタンをクリックした際の処理の中に、setIntervalメソッドを使用したタイマー処理を追加します❶。
第1引数には、無名関数で時間を更新する処理を追記します❷。「--」と書かれている部分はデクリメント演算子といい、変数の数値を1減らす働きを持ちます❸。第2引数には、1000ミリ秒、つまり1秒を指定して、1秒ごとに残り時間が1減るように処理を記述します❹。

```
024 button.addEventListener('click',_function(){
025 __//_フォームを表示
026 __form.style.display_=_'block';
027
028 __//_タイマー処理で残り時間を表示
029 __var_timerId_=_setInterval(function(){
030 ____timeMessage.innerHTML_=_'<p>制限時間：'_+_reminingTimeNum_+_'秒</p>';
031 ____reminingTimeNum--;
032 __},1000);
033 });
```

1 setIntervalメソッドを追加
2 残り時間を更新
3 変数を減らす
4 タイマーの間隔を指定



うまく動かないときは慌てずにコンソールを表示しましょう（P.43参照）。エラーが発生した行が表示されるので、そこにミスが隠れている可能性が高いです。

3 制限時間が切れたらメッセージを出す

最後に、制限時間が切れたらメッセージを出す処理を追記します。
条件分岐で残り時間が0秒以下になったとき❶、警告ダイアログでメッセージを表示して❷、clearIntervalメソッドでタイマー処理を終了しています❸。

```
028 __//_タイマー処理で残り時間を表示
029 __var_timerId_=_setInterval(function(){
030 ____timeMessage.innerHTML_=_'<p>制限時間：'_+_reminingTimeNum_+_'秒</p>';
031 ____if_(reminingTimeNum_<=_0)_{
032 ______alert('制限時間終了');
033 ______clearInterval(timerId);
034 ____}
035 ____reminingTimeNum--;
036 __},1000);
```

1 残り時間をチェック
2 メッセージを表示
3 タイマーを解除

4 制限時間が表示される
5 制限時間が終了するとメッセージが出る

無事に制限時間を設定できましたか？　問題なく動作したら、短めに設定しておいた「残り時間」の値を実用的な値に修正しておきましょう。

ワンポイント 一定時間後に一度だけ実行するタイマー処理

タイマー処理にはsetIntervalメソッドの他にsetTimeout()というメソッドが用意されています。使い方は基本的に同じですが、こちらは、一定間隔で処理を繰り返すのではなく、一定時間後に一度だけ処理を実行します。第1引数に実行したい処理を記述した関数を、第2引数に、実行までの待ち時間（ミリ秒）を指定します。

一定時間ごとの繰り返し処理はsetIntervalメソッドで記述するのが一般的ですが、一回の処理が繰り返し時間内に終わり切らなかった場合、前の処理が終了する前に次の処理を開始してしまう危険性があります。そんなときには、setTimeoutメソッドを使用して繰り返し処理を記述することもできます。

setIntervalメソッドと違って必ず「処理が終了してから」間隔をあけて次の処理を実行するので、実行時間がわからないときは、こちらの書き方が便利です。

▶ setTimeoutメソッド

```
var timer = setTimeout(timerfunc, 1000);
```

タイマー識別用の変数　実行したい関数　待ち時間

▶ clearTimeOutメソッド

```
clearTimeOut(timer);
```

タイマー識別用の変数

▶ setTimeoutメソッドによる繰り返し処理の例

```
function foo() {
  //setTimeoutメソッドで1秒後に関数fooを呼び出す
  setTimeout(foo, 1000);
  console.log('繰り返し');
}
foo();
```


Chapter

8

データをまとめて扱おう

この章では、データをまとめて扱うことのできる「配列」という仕組みや、これまで利用してきた「オブジェクト」を自分で作る方法について学びます。