アプリ
プログラミングシート

複利計算 編

学習目標

観点	学習目標

知識・技能	• モデル化とシミュレーション
	  • 「積み立て」概念を説明できる
	  • 「複利」概念を説明できる。複利によって資産がどのように増えるか、モデルを示すことができる
	• プログラミング
	  • 配列にデータを格納して、格納した値を利用できる
	  • 繰り返しの制御構造を用いて、配列の要素を操作できる
	  • プログラムで計算した値を、表形式で表示できる
	  • プログラムで計算した値を、グラフ形式で表示できる

思考力・判断力・表現力	• 積み立て額や積み立て期間、金利を変更すると、結果がどのようになるか調べ、比較・検討できる • シミュレーションの結果を表示するときに、どの形式を用いれば分かりやすい表示になるか判断できる • 表形式やグラフ形式を使って、データを分かりやすく表現する
学びに向かう力	• 積み立て額や金利、積立期間を変更すると、結果がどのようになるか調べている • 積み立てや複利の他に、資産形成に関連する概念を調べてみる • グラフ表示ライブラリーの機能を調べてみる

単元の流れ

コマ	内容	狙い
1	複利計算アプリの動作を確認する グラフ形式の表示と、表形式の表示を比較する 積み立ての金額、利率を変更して、貯蓄額がどのように変化するか確認する	「積み立て」と「複利計算」の概念を確認する。 積み立て：定期的に決まった金額を入金し、一定期間を経過するまで繰り返す。 任意のタイミングではなく、決まったタイミングで定期的に行うのが通常の預貯金と異なる。 複利：元金に利率をかけて求める利息が毎期付くだけでなく、前の期の利息分にも、次の期には利息がつく。なお、元金分にしか金利がかからない方式を単利と呼ぶ。 グラフは視覚に訴えるが、個々の年の数値（金額）を正確に読むのには表の方がよい。 同じデータでも、表示の方式によって見え方・強調されることが変わることを確認する。
2	プログラムを読み、処理内容を理解する	ソースコードを読み、処理内容を確認する。 学習項目 • ユーザーが画面に入力した値の取得方法（document.getElementById(id).value） • 繰り返し処理（積み立て期間の間、計算を繰り返す） • 配列の操作（配列に要素（計算結果）を追加する） • グラフ表示する
3	カスタマイズ課題に取り組む	提示されるカスタマイズ課題を実装する。 カスタマイズ課題 1. 期間を指定できるようにする 2. 目標値を設定して、目標を達成する時期を計算できるようにする

1コマ目の指導

過程	内容
導入	サンプルのアプリを動作させて、「積み立て」と「複利計算」について学ぶことを伝える。 Monaca Educationへのログインを行う。 APS複利計算を、ダイレクトインポートでインポートする。
展開1	短く、「積み立て」と「複利計算」の概念を確認する。 サンプルアプリを動作させて、結果を見る。 積み立て金額・利率を変更して、結果がどのように変わるか、確かめる。 ただし、試行ごとに積み立て金額と利率の両方を同時に変更すると、どちらの影響で結果がどのように変わったか分かりにくい。 「一方を変更するときは、他方を固定する」考え方を示唆するとよい。 時間を取って、生徒に何パターンか試させて、どのようなことが言えるかを問う。 発問１：積み立て金額と10年後の金額の関係は？ （考えられる回答） • 積み立て金額が大きいほど、10年後に貯まる額が多くなる。 発問２：利率と10年後の金額の関係は？ （考えられる回答） • 利率が大きいほど、10年後に貯まる額が多くなる。 さらに、10年後だけでなく、途中の年の金額がどうなるかを問う。 （考えられる回答） • 利率の差の影響は、最初のうち（～3年くらいまで）は小さい。徐々に影響が大きくなる
展開2	展開１の発問について考えるにあたり、表とグラフのどちらが役に立ったかを問う。 また、表とグラフのそれぞれの特徴・長所・短所を問う。
補足	金融商品の詳細に踏み込む必要はないが、現在の日本の経済状況では「利率が固定で2%の積み立て」のようなものは無いことは示唆してもよいかもしれない。 この単元は、積み立て・複利の理解のための例であり、実際に資産形成を考える際には、金融商品の特徴を調べ、リスクを理解する必要があることを伝えること。

2コマ目の指導(1)

過程	内容
導入	前回の振り返り：積み立てと複利のモデル、シミュレーション、結果の表示形式（グラフ、表）の特徴 この時間は、ソースコードを読み、処理内容を確認すること、可能であれば提示されているカスタマイズ課題に挑戦することを話す。

展開1	HTMLの説明：
	• <input>タグが入力欄になる。タグの周囲の文字列（“積み立てる金額”、“円”など）を編集・保存すると、プレビューに反映されることをデモすると伝わりやすいし、生徒が飽きない。
	• がグラフ表示欄になる。
	• <table>タグが結果一覧になる。
	• <button>タグで、ボタンが表示される。onclick=”calcInterest()”で処理を呼び出している。
	
	※ただし、結果表示欄については作成済みの記述を利用するので、<input>欄に注目することを伝える。
	
	JavaScriptの説明：
	• 「計算」ボタンが押されたら、関数calcInterest()が実行される。
	• document.getElementById()で、HTMLの値を取得する。引数でIDを指定していることを強調するとよい。わざとIDを変えて、値が取れなくなる様子をデモで見せるとよい。生徒が飽きない。
	• 配列の宣言 var sougaku = []; sougakuは配列になる。
	• 配列に要素を追加 sougaku.push(値);
	• 繰り返し制御 for
	  • 変数iはカウンター。数を数えるための変数。
	  • forの二つ目の式は、継続するかどうかの判定式。
	  • 三つ目の式は、カウンターの更新式。
	• calcInterest()の最後に、関数plot()を呼び出している。グラフ表示は、この関数内で行っている。
	  • グラフのタイトルや、横軸・縦軸の見出しを設定している。
	
	  • ※グラフ表示のライブラリPlotly.jsの詳細は説明しなくてもよいが、興味のある生徒には以下の説明をしてもよい。
	  • JavaScriptでは、オブジェクトを { … } で作れる。オブジェクトの中で「名前: 値」の組を作っている。
	  • Plotly.jsは、オブジェクトを受け取って、オブジェクトの中に記述されている値を使ってグラフ表示をしている

2コマ目の指導(2)

過程	内容

展開2	提示されているカスタマイズ課題を実装する。
	
	カスタマイズ課題(1)積み立てる年数を指定できるようにする
	• インポートしたサンプルアプリは、年数は10年で固定になっている。これを変更できるようにする。
	
	• 変更は、HTMLと、JavaScriptの両方に必要。
	
	• HTML: <input>タグを用いて、年数を入力する欄を追加する。他の入力欄（たとえば「利率」）の行をコピーして、名称などを修正する方法をとると、比較的簡単に追加できる。
	
	• JavaScript:
	  • HTMLの要素を指定して、年数を取得する。
	  • 取得した年数を使って、繰り返し処理の条件を変更する。
	
	• 編集が終わった後、動作確認をするよう伝える。

まとめ	• 前回の授業で学んだ概念（積み立て、複利計算）について、モデル化し、プログラムで実現していることを確認する。
補足	カスタマイズを生徒にさせる前に、「行のコピー」「ファイルの保存」操作について、デモを見せておくと、生徒の戸惑い・手間取りを減らすことができる。 「ファイルの保存」については、編集中で、未保存のファイルにはMonacaの画面上「★」の表示がされることも伝えておく。 HTMLに追加する<input>タグのidを、document.getElementById()の引数で指定すること。この二つが（大文字・小文字まで）正確に一致しないと正しく値が取れないので注意。

3コマ目の指導

過程	内容
導入	前回までの振り返り： • 積み立てと複利のモデル、シミュレーション、結果の表示形式（グラフ、表）の特徴 • ソースコードの確認。HTML、JavaScript（繰り返し、配列） • 提示されるカスタマイズ課題を実装する

展開	提示されるカスタマイズ課題を実装する。
	
	カスタマイズ課題(1)積み立てる年数を指定できるようにする
	（※前回からの続き）
	
	カスタマイズ課題(2)目標値を設定して、目標を達成する時期を計算できるようにする
	• インポートしたサンプルアプリは、年数で繰り返し回数を制御している。これを目標値と合計値の比較で制御するようにする。
	
	• 変更は、HTMLと、JavaScriptの両方に必要。
	
	• HTML：<input>タグを用いて、目標金額の入力欄を追加する。他の入力欄（たとえば「利率」）の行をコピーして、名称などを修正する方法をとると、比較的簡単に追加できる。
	
	• JavaScript:
	  • HTMLの要素を指定して、目標金額を取得する。
	  • 取得した目標金額を使って、繰り返し処理の条件を変更する。
	  • forループによる「繰り返し回数の制御」ではなく、whileループによる「合計値と目標値の比較による制御」にする。
	
	• HTMLに追加する<input>タグのidを、document.getElementById()の引数で指定すること。この二つが一致しないと正しく値が取れないので注意。

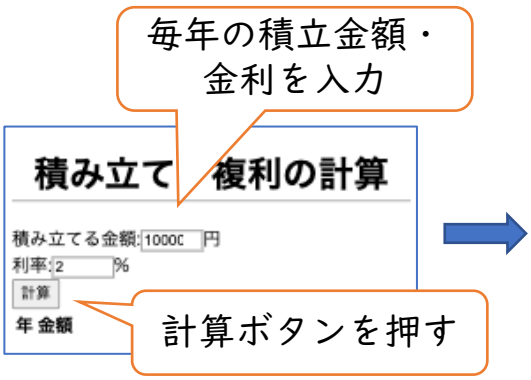
まとめ	• 前回の授業で学んだ概念（積み立て、複利計算）について、モデル化し、プログラムで実現していることを確認する。 • 当初のアプリのモデルに、「積み立てする期間」および「目標金額」の概念を追加した。作業は「プログラムの変更」であったが、実はその前に「モデルの変更」をしていたことを伝えると、モデル・シミュレーション・プログラミングの間の関係を整理することにつながる。
補足	進捗の早い生徒には、関数plot()の中の値（グラフのタイトルや横軸・縦軸の見出し、グラフのサイズなど）を変更するカスタマイズを指示してもよい。自由なカスタマイズをさせる場合、カスタマイズの前にプロジェクトのエクスポートを行うよう指導するなどして、元に戻せるようにしておくと安心して進めることができる。

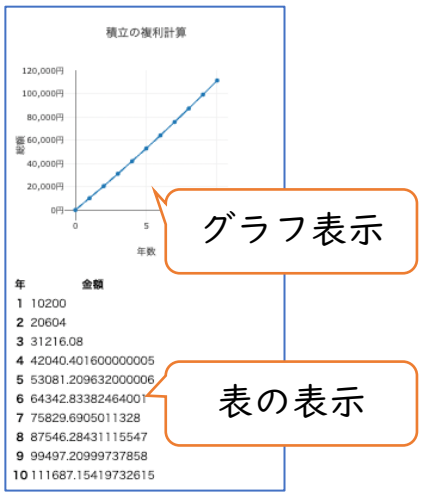
複利計算

アプリの概要

サポートページから
複利計算アプリを
クリック

インポートしてIDEで開く

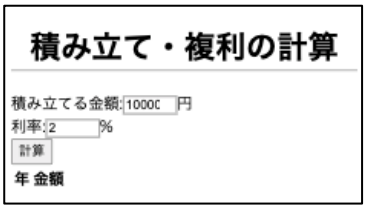




学習内容

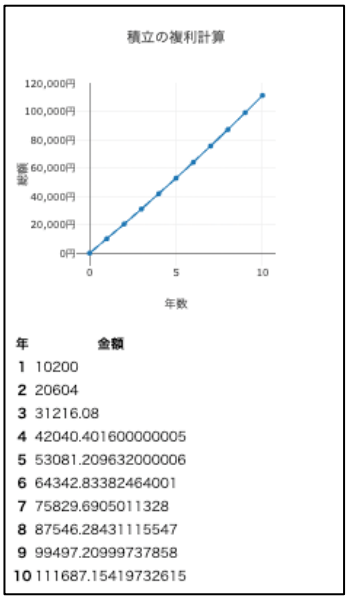
項目	内容
モデル	考察する対象について、単純化・抽象化すること。 このAPSでは、積み立ておよび複利計算をモデル化する。
シミュレーション	モデルに対して仮定を適用し、その仮定の下ではどのような結果が得られるか調べること。 このAPSでは、積み立て金額や積み立て期間、目標金額などを仮定し、結果を得るシミュレーションを行う。
表とグラフ	表形式 • 長所 • 短所 グラフ形式（折れ線グラフ） • 長所 • 短所
繰り返し処理	for、whileキーワードを使い、繰り返しを行う。
配列	配列を操作する（データを追加・データを参照）。
document.getElementById()	HTMLのDOMを操作する。

アプリの概要



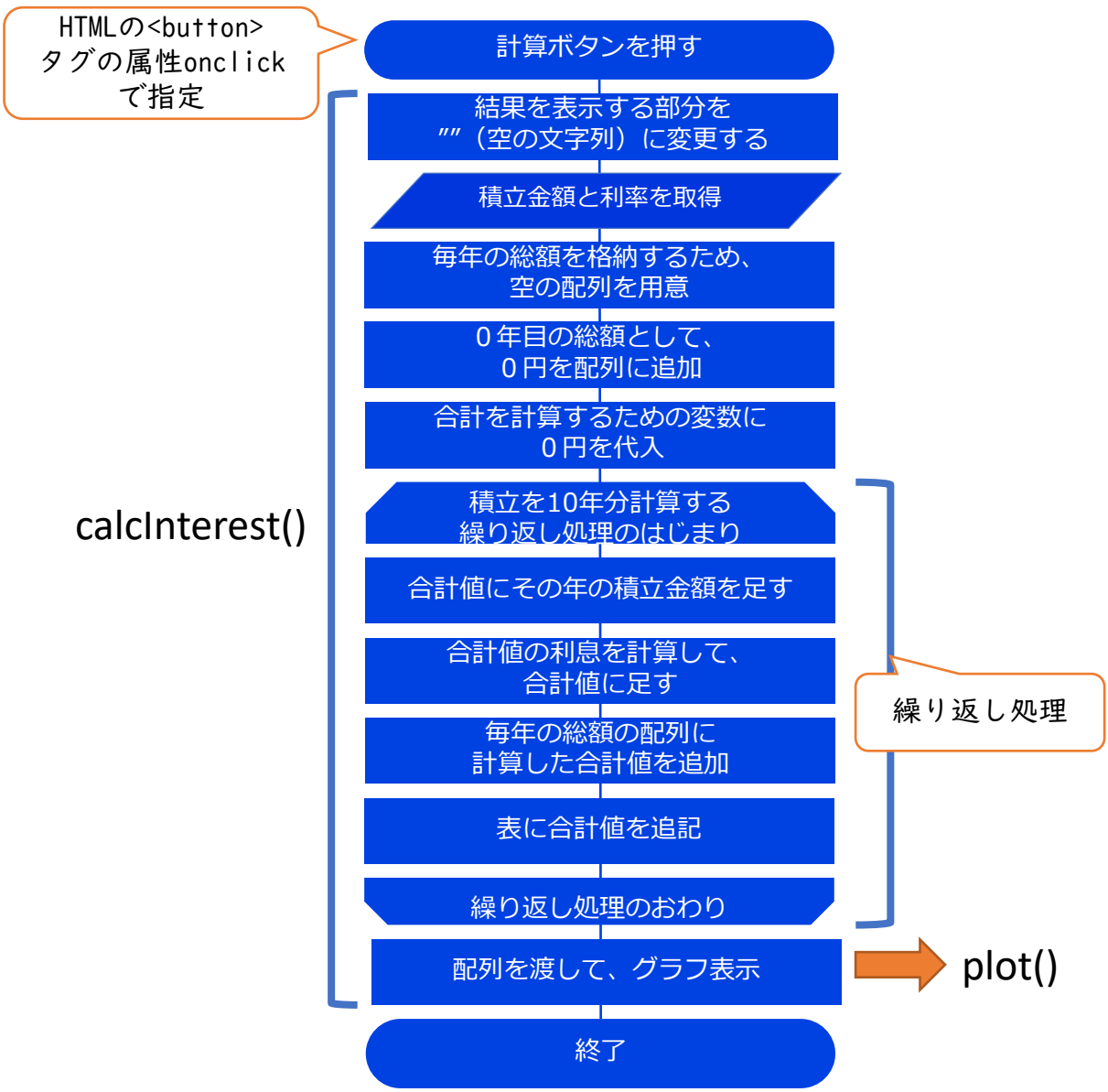
① 毎年積み立てる金額と、適用される金利を入力する

② 「計算」ボタンを押す

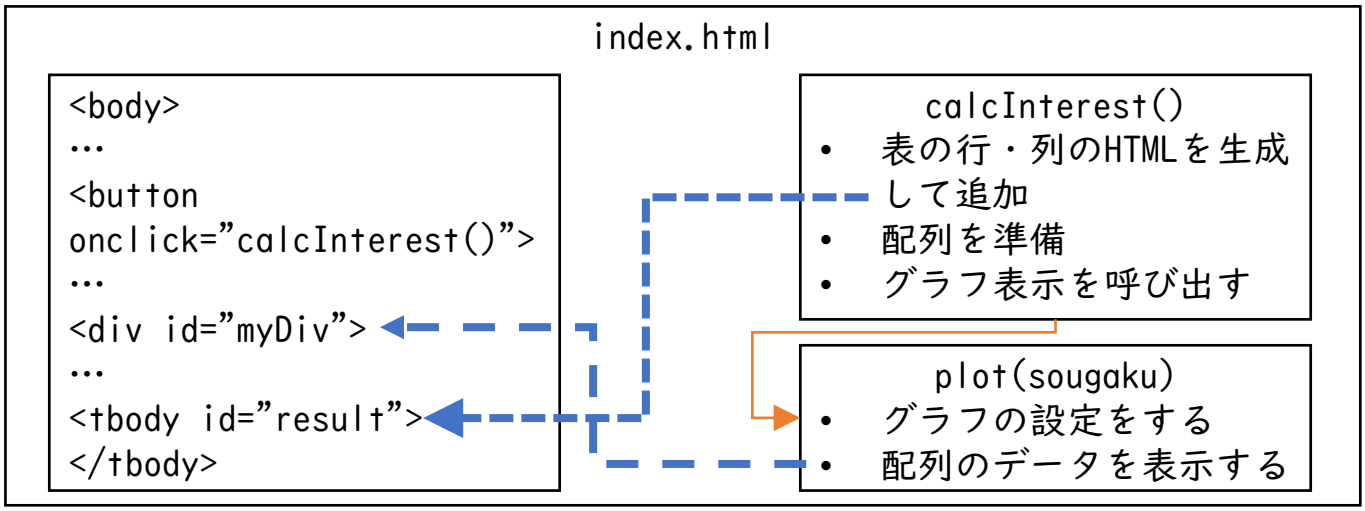


- X軸に年数、Y軸に総額をとったグラフが表示される
- グラフの下には、経過した年数と、その年までに溜まった総額が表の形式で示される

フローチャートで処理の流れを確認しよう



プログラムを読んでみよう①



HTMLの<body>部

```
<body>
    <h1>積み立て・複利の計算</h1>
    <div class="interest">
    積み立てる金額:<input type="number" value="10000" id="addition"
    placeholder="金額">円<br/>
    利率:<input type="number" value="2" id="rate" placeholder="利率(%)">%<br/>

    <button onclick="calcInterest()" >計算</button>
    </div>

    <div id="myDiv"></div>

    <table>
      <thead>
        <tr><th>年</th><th>金額</th></tr>
      </thead>
      <tbody id="result">
      </tbody>
    </dl>

</body>
```

見出し

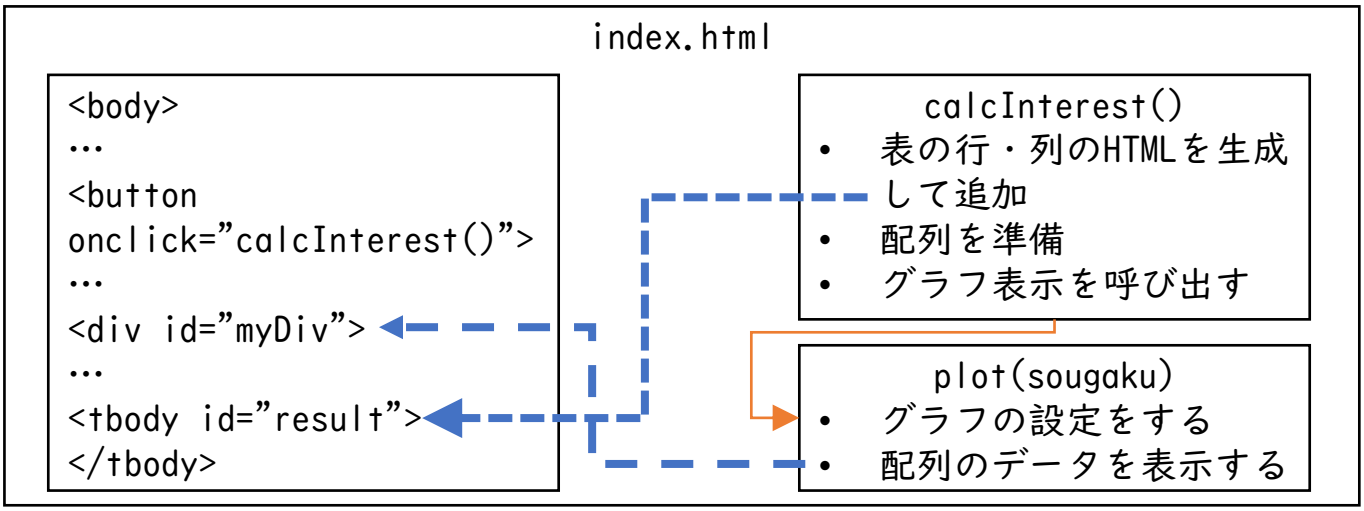
2つの入力欄

「計算」ボタン
onclickで、クリックされたときにcalcInterest()を呼び出す指定

myDivは、グラフを表示するエリア

<table>は、表を作るタグ

プログラムを読んでみよう②



<script>のcalcInterest関数（前半）

```
<script>
    function calcInterest() {
        // 結果を表示する部分を""（空の文字列）に変更する
        document.getElementById("result").innerHTML
        = "";

        // 積立金額と利率を取得
        //  HTMLからJavaScriptの変数に取る
        //  毎年の積立額
        var addition = parseInt
        (document.getElementById("addition").value);
        //  適用される利率
        var rate = document.getElementById("rate")
        .value;

        // 毎年の総額を格納するため、空の配列を用意
        var sougaku = [];

        // ０年目の総額として、０円を配列に追加
        sougaku.push(0);

        // 合計を計算するための変数に０円を代入
        var amount = 0;
```

関数の宣言

表の中身を空白に

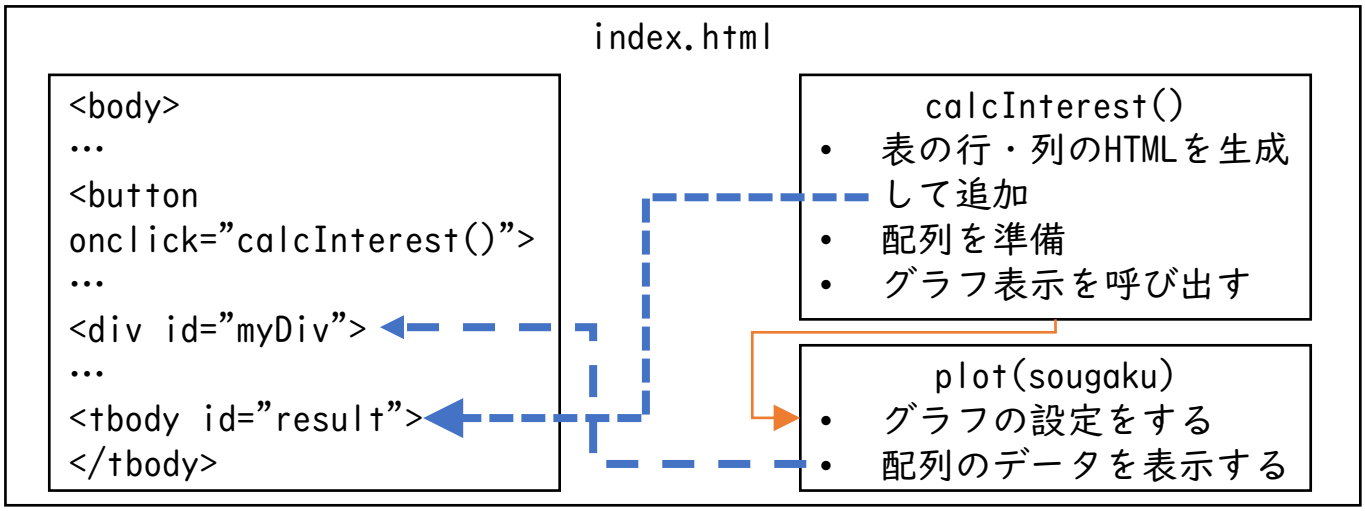
HTMLの積み立て額入力欄から値を取得し、整数型にして変数に代入する

HTMLの利率入力欄から値を取得し、変数に代入する

配列を用意

配列の最初の要素として、０を追加

プログラムを読んでみよう③



<script>のcalcInterest関数（後半）

```
// 合計を計算するための変数に０円を代入
var amount = 0;

// 積立を10年分計算する繰り返し処理のはじまり
//  1年ずつ計算する。1年目から10年目まで
for (var i = 1; i <= 10; i++){
  // 各年時点の総額を求める

  // 合計値にその年の積み立て金額を足す
  amount += addition;
  // 合計値の利息を計算して、合計値に足す
  amount += amount * rate / 100;

  // 毎年の総額の配列に計算した合計値を追加
  sougaku.push(amount);

  // 表に合計値を追記
  var tr = document.createElement("tr");
  var th = document.createElement("th");
  th.innerHTML = i;
  tr.appendChild(th);
  var td = document.createElement("td");
  td.innerHTML = amount;
  tr.appendChild(td);
  document.getElementById("result")
  .appendChild(tr);

}
```

合計を計算するための変数を用意

繰り返し処理の開始

その年の積み立て金額を、合計値に足す

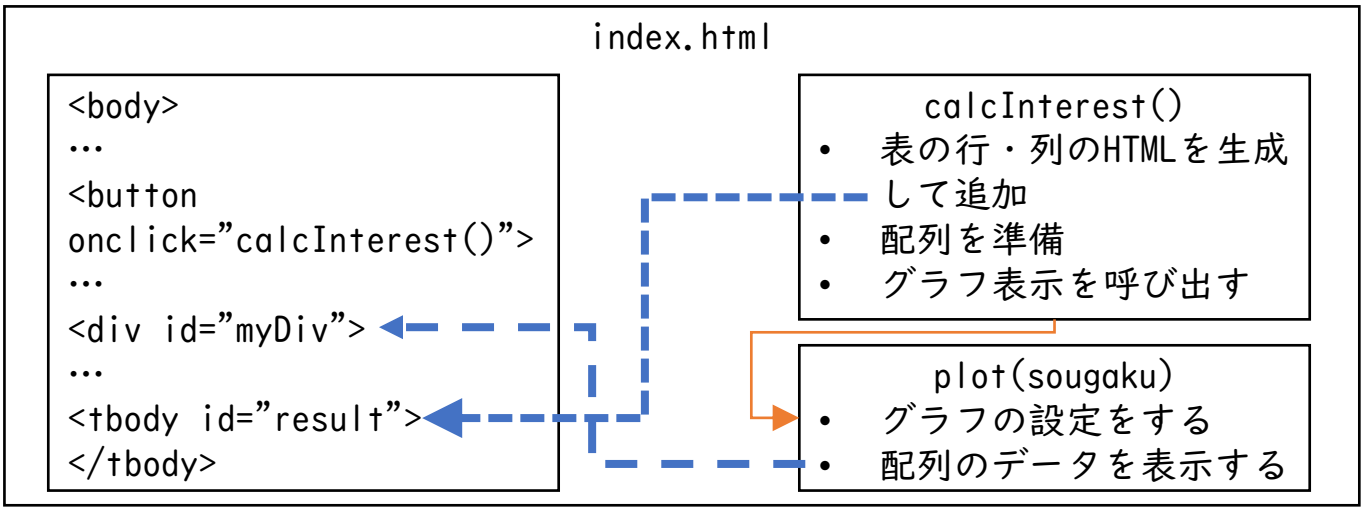
利率をかけて、利息分を求め、その結果も足す

その年の時点の合計値を配列の末尾に追加する

表の行・列を追加

繰り返し処理の終端

プログラムを読んでみよう④



<script>のplot()関数

```
    // 配列を渡して、グラフ表示
    plot(sougaku);
}

function plot(sougaku){
    // グラフを書き出す<div>のID属性の値
    var graph = "myDiv";
    var layout = {
        height: 400,
        width:  400,
        showlegend:false,
        title:"積立の複利計算",     // グラフのタイトル
        xaxis: {
            title: "年数"           // 横軸の見出し
        },
        yaxis: {
            title: "総額",          // 縦軸の見出し
            ticksuffix:"円",
            exponentformat:"none",
        },
                    };
        let trace = {
            y: sougaku,
            mode: 'lines+markers',
            type: 'scatter'
        };
    let data = [trace];

    Plotly.newPlot(graph, data, layout);
}
```

表示位置を指定

calcInterest()の末尾で
plot()関数を呼び出している

plot()関数の宣言

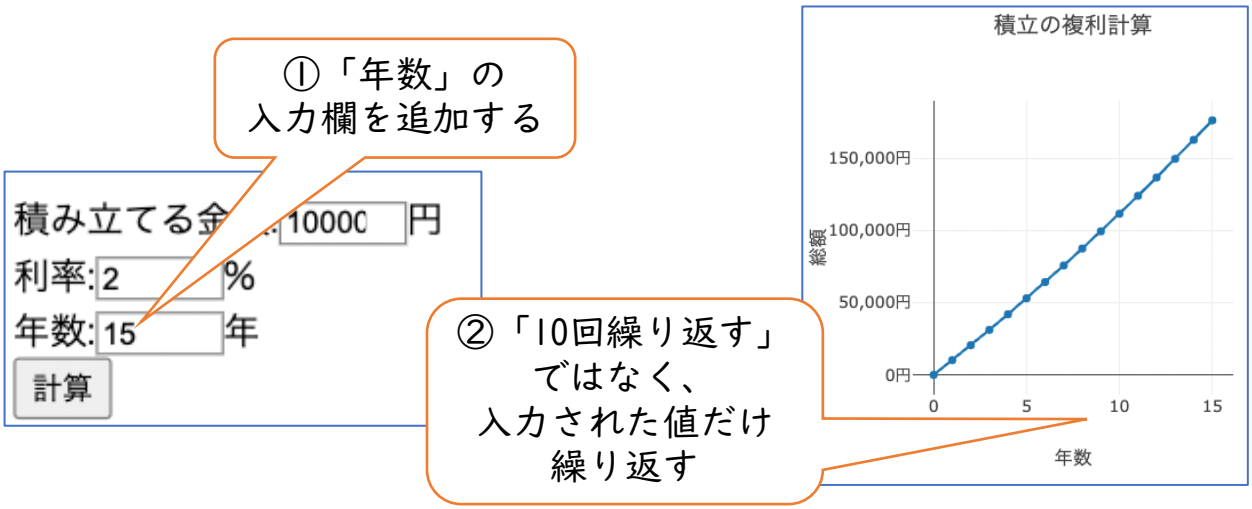
グラフのサイズの指定

グラフのサイズの指定

グラフのタイトル、
グラフの横軸、縦軸の指定

グラフ表示のライブラリを呼び出す

カスタマイズ(1)積立の年数を変更できるようにしよう



HTMLの変更

```
利率:<input type="number" value="2" id="rate" placeholder="利率(%)">%<br/>
年数:<input type="number" value="10" id="years" placeholder="年">年<br/>
```

①HTMLの<body>タグの中に、
「年数」欄を追加する。
id="years"とする。
「利率」の行をコピーして、
書き換えると簡単

JavaScriptの変更

積立金額と利率、年数を取得

毎年の総額を格納するため、
空の配列を用意

0年目の総額として、
0円を配列に追加

合計を計算するための変数に
0円を代入

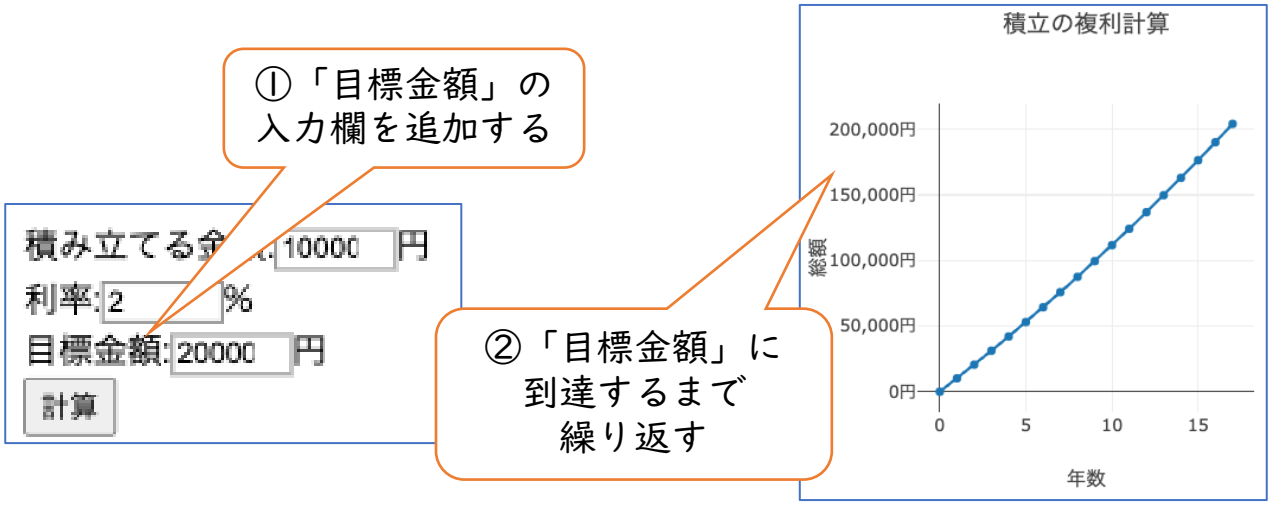
積立をyears年分計算する
繰り返し処理のはじまり

```
//  適用される利率
var rate = document.getElementById("rate").value;
//  積み立てる年数
var years = document.getElementById("years").value;
```

②変数yearsの値を使って、
ループの繰り返し回数を決める

```
// 積立を10年分計算する繰り返し処理のはじまり
//  1年ずつ計算する。1年目からyears年目まで
for (var i = 1; i <= years; i++){
```

カスタマイズ(2)目標金額までの計算にしよう



HTMLの変更

```
利率:<input type="number" value="2" id="rate" placeholder="利率(%)">%<br/>
目標金額:<input type="number" value="200000" id="target" placeholder="目標">円<br/>
```

①HTMLの<body>タグの中に、
「目標金額」欄を追加する。
id="target"とする。
「利率」の行をコピーして、
書き換えると簡単

JavaScriptの変更

積立金額と利率、
目標金額を取得

毎年の総額を格納するため、
空の配列を用意

0年目の総額として、
0円を配列に追加

合計を計算するための変数に
0円を代入

年数を数える変数ｉに０を代入

合計が目標金額を下回る間、
繰り返す処理のはじまり

変数ｉに１を足して代入

```
//  適用される利率
var rate = document.getElementById("rate").value;
//  目標金額
var target = document.getElementById("target").value;
```

②目標金額を超えるまで繰り返す。
年数を数えるための変数iを更新する

```
// 繰り返し処理のはじまり
//  1年ずつ計算する。目標金額を超えるまで
//  変数iは、年数を表示するために使用
var i = 0;
while( amount < target ){
  i = i + 1;
```

確認テスト

問題	回答
「積み立て」の概念を説明してください。	• 定期的に • 決まった金額を • 一定期間 貯めていく、貯蓄方法
「複利」の概念を説明してください。	• 元金に対して、金利が付く • 前の期についた金利分も含めて（元金となり）、次の期に金利が付く
毎日お小遣いがもらえるとします。ある日のお小遣い金額は、前の日の金額の2倍です。 1日目に10円もらえたとき、５日目までにいくら貯まりますか？５日間の間は、１円も使わないとします。 表を作って整理して、答えを求めてください。	<table><tr><th></th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th></tr><tr><td>日額</td><td>10</td><td>20</td><td>40</td><td>80</td><td>160</td></tr><tr><td>総額</td><td>10</td><td>30</td><td>70</td><td>150</td><td>310</td></tr></table>

AさんとBさんの二人が、毎月使ったお金の金額を記録しました。1年分（12ヶ月）の記録について、二人の記録を見比べるとき、
• 表形式
• グラフ形式（折れ線グラフ）
それぞれについて、特徴（長所・短所など）を説明してください。
	表形式
	• 長所
	  • 数値を一覧で確認できる
	• 短所
	  • 変化・変化の度合いをつかむのが難しい
	
	グラフ形式（折れ線グラフ）
	• 長所
	  • 変化・変化の度合いを目で見て確認できる
	• 短所
	  • 一つ一つの数値は見えにくい

以下のプログラムの下線部に当てはまるプログラムのコードを示してください。 なお、このプログラムでは、ユーザーが、HTMLの中にあるID属性”rate”の<input>タグに入力した値を変数rateに代入します。 var rate = _____;	document.getElementById(”rate”).value

１からはじめて、１０まで繰り返す処理を、forキーワード、変数iを使った書き方で示してください。	for (var i = 1; i <= 10; i++){
	
	}

プログラムのコードによる回答については、考え方が正しければ、JavaScriptの文法に完全に従っていなくても正解とします。