



福井大学 リカレント 教育プログラム プログラミング応用

(2) JavaScriptによる
フロントエンドプログラミング (10/08)

プログラミング応用 講義資料URL

<https://tsaitoh.net/~t-saitoh/2022-10-recp/>

login: guest

password: Guest

簡単なパスワード
認証を追加しました

福井大学リカレント教育プログラム
プログラミング応用

簡単なパスワード
認証を追加しました

リンク

- Twitter @TohruSaitoh
- Facebook tsaitoh.net
- tsaitoh.net@google.com

講義内容と講義資料

- Webアプリケーションとプログラム言語(10/01)
 - 配布データ 2022-10-01-recp-1.zip
 - インターネットやWebの仕組みについて理解し、その中でJavaScriptやPHPなどのプログラム言語がどう使われるのか
 - 課題レポート
 - 理解度確認(10/01) (Google Formsに回答してください)
 - nslookup コマンドで、www.fukui-nct.ac.jp のIPアドレスを調べてください。
 - そのIPアドレスを使ってWebページを開いてください。
最近のブラウザはhttp://x.x.x.xで開くと、「安全か確認できないけど開きますか?」といった警告がでますが、「危険性を理解したうえで開く」を実行してみてください。
 - 自分の自己紹介の文章を C:\xampp\htdocs\recp\profile.html に作成し、ブラウザで http://localhost/recp/profile.html を表示させてください。
 - 2,3で確認した内容の画面をキャプチャしたものと、4で作成したページの画面をキャプチャしたものをレポートにまとめ、メールでtsaitoh@fukui-nct.ac.jpに提出してください。

配布データのダウンロードと解凍-1

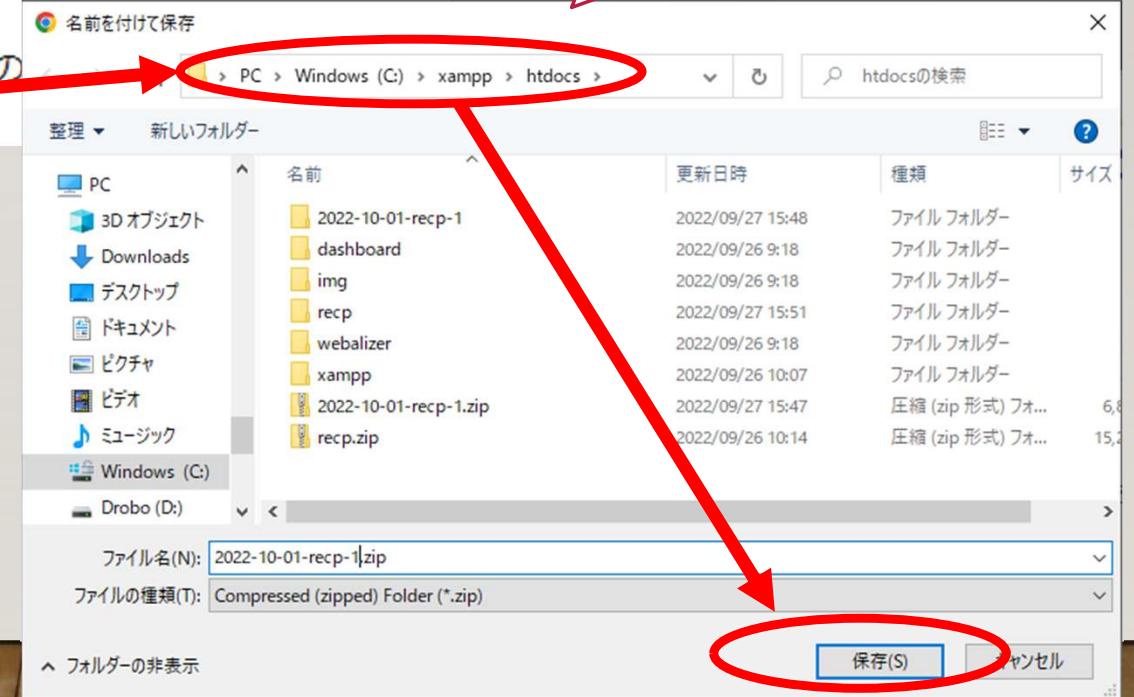
- JavaScriptによるフロントエンドプログラミング(10/08)
 - 配布データ [2022-10-08-recp-2.zip](#)
 - Webブラウザ側で動くプログラム言語としてのJavaScriptについて、

指

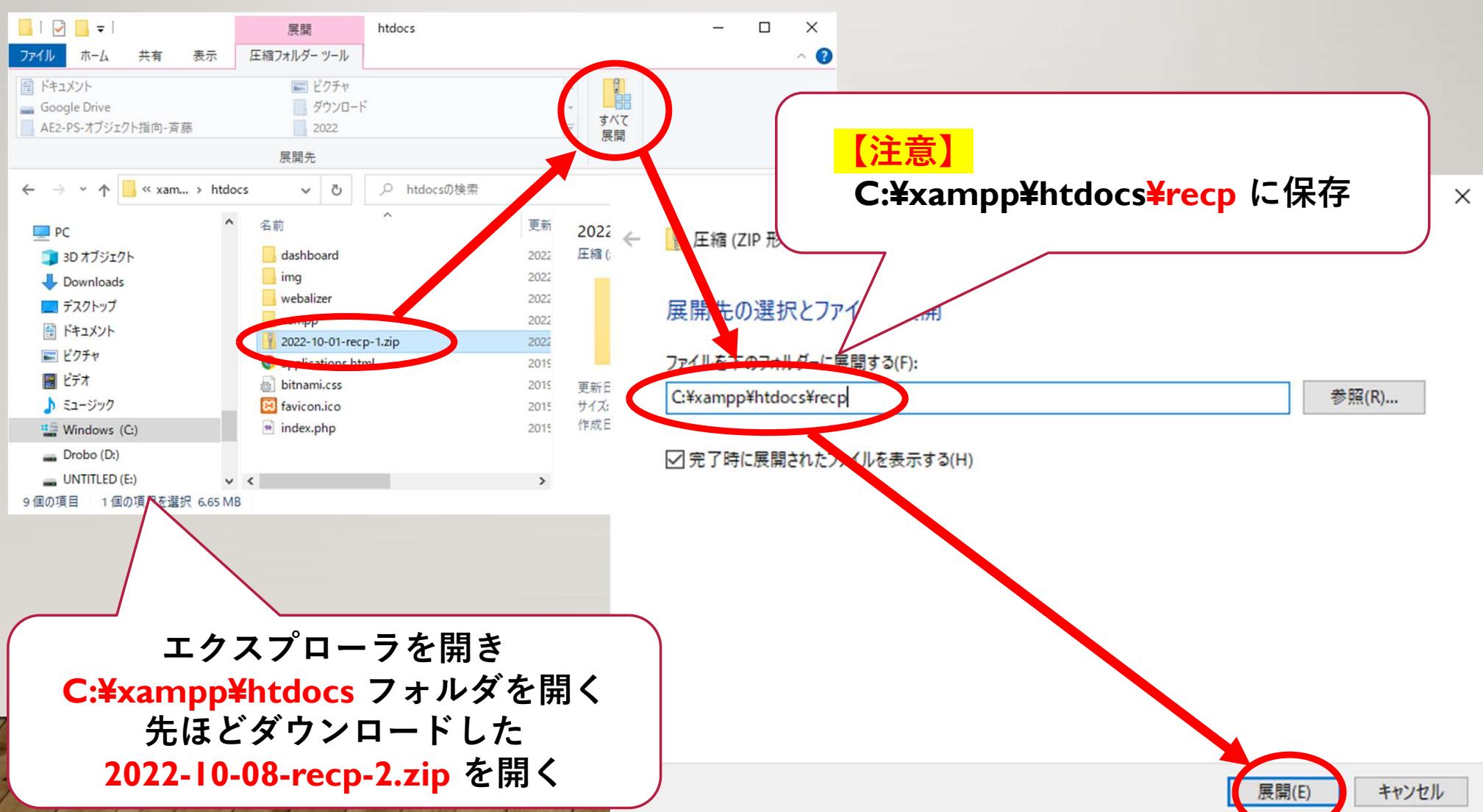
課

新しいタブで開く
新しいウィンドウで開く
シークレット ウィンドウで開く
名前を付けてリンク先を保存...
リンクのアドレスをコピー
検証

C:\xampp\htdocs に保存



配布データのダウンロードと解凍-2



本日の目標

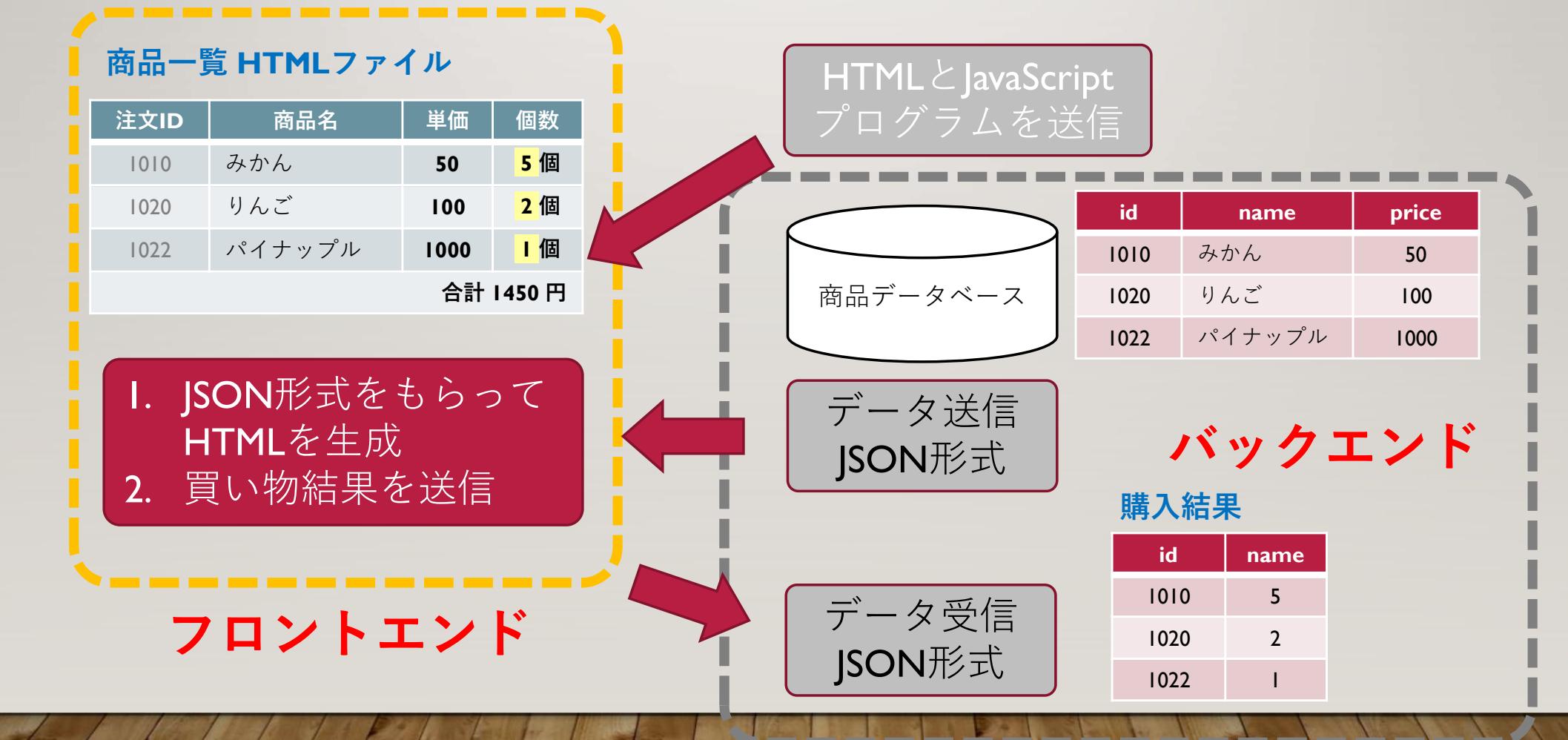
JavaScriptによる フロントエンドプログラミング(10/08)

- JavaScript とは
 - Hello World
 - 基本的な型
 - 制御構文
 - 練習問題(棒グラフ)
 - オブジェクトを使ったプログラム
 - オブジェクトと連想配列
 - オブジェクトの配列
 - 練習問題(配列の串刺し)
 - 発注ページ
- jQueryの使い方
 - JSONデータの読み込み
- Webの公開とWebサービスの公開
- まとめ

| 注文ID | 商品名 | 単価 | 個数 |
|----------|---------|------|--------------------|
| 1010 | みかん | 50 | 1 |
| 1020 | りんご | 100 | 2 |
| 1022 | パインアップル | 1000 | 3 |
| 合計=3250円 | | | 発注 |

前回資料より抜粋

JavaScriptによる フロントエンドプログラミング(10/01)



JavaScript とは



- ブラウザの表示内容を制御するためのプログラム言語
 - フロントエンド
- Java とは別の言語
- 最近は Node.js など、バックエンドでも活用
- ECMAScript – 標準化



JavaScript

paiza.io を使って 簡単に練習-I

The image shows two browser windows side-by-side. The left window is the English version of the paiza.io website (<https://paiza.io/en>) with a red callout pointing to the URL. The right window is the Japanese version (<https://paiza.io/ja>). Both windows have red circles highlighting specific elements: the language dropdown menu, the Japanese language button, the 'New code' button, the 'PHP' language selection, and the 'JavaScript' language button.

paiza.io を使って
簡単に練習-I

<https://paiza.io/>

Just code online!

Which language will you use?

Bash C C# C++ CoffeeScript D E Haskell Java Java Nadesiko Objective-C Python3 R Ruby Swift TypeScript

PHP

新規コード

日本語

English

Recent codes WebDev English Sign up

English

日本語 Spanish

For web development

paiza.IO is an online environment where you can start programming immediately. It's an online execution environment.

C, C++, Java, Ruby, Python, PHP, Perl, etc. are supported. Mainly 24 languages are supported. File upload function, connection to external API, and cloning are also possible.

コード作成を試してみる (無料)

paizaCloud

Leave a message

paiza.io を使って 簡単に練習-2

The screenshot shows the paiza.io web interface for practicing programming. At the top left is the logo with a red 'Beta' badge. A dropdown menu is open, showing 'JavaScript' with a red oval and arrow pointing to it. To the right is a text input field labeled 'Enter a title here'. Below the menu is a file list with 'Main.js *' and a '+' icon. The main area is a code editor with the following code:

```
Success ツイート
1 process.stdin.resume();
2 process.stdin.setEncoding('utf8');
3
4 console.log("Hello World");
```

A red dashed box highlights the last four lines of code (lines 3 and 4). A callout bubble points to this box with the text 'JavaScript のプログラムを入力' (Input JavaScript program).

At the bottom left is a green button labeled '実行 (Ctrl-Enter)' with a play icon, also highlighted with a red oval and arrow. A callout bubble points to this button with the text 'JavaScript のプログラムを実行' (Execute JavaScript program).

At the bottom center is an output console showing the result 'Hello World' in a red-bordered box. A callout bubble points to this result with the text '実行結果が表示' (Execution result displayed).

The footer of the interface includes links for 'JavaScriptを学ぶ' and 'プログラミング力診断', and a series of circular icons for user profile and settings.

Webの中で使う場合

Hello World

ページの一部を書き換える

- JavaScript ブラウザの表示内容を書き換える。
- 書き換えたい場所に **id 属性** をつけておく。
- **getElementById**(“属性名”)で見つけ出し、
- その内側 **innerHTML** を書き換える。

```
<div id="目印"></div>
```

```
document.getElementById("目印").innerHTML  
= "Hello World" ;
```

Hello World (I)

sample3.html

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="utf-8"/>
    <title>Sample Page 3(Hello World)</title>
    <script type="text/javascript">
      // <body onload="main()"> により
      // ページ全体が読み込まれたとき(onload)に
      // 関数 main() を実行する。
      function main() {
        /*
          <div id="output"></div>の場所をみつけて(getElementById())
          その内側innerHTMLを "Hello World" に書き換える。
        */
        document.getElementById( "output" ).innerHTML
          = "Hello World!" ;
      }
    </script>
  </head>
  <body onload="main()">
    <h1>Sample Page 3<br/>(Hello World)</h1>
    <!-- この部分を書き換える -->
    <div id="output"></div>
  </body>
</html>
```

Sample Page 3
(Hello World)

Hello World!

Hello World (2)

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="utf-8"/>
    <title>Sample Page 3(Hello World)</title>
    <script type="text/javascript">
      // <body onload="main()"> により
      // ページ全体が読み込まれたとき(onload)に
      // 関数 main() を実行する。
      function main() {
        /*
          <div id="output"></div>の場所をみつけて(getElementById())
          その内側innerHTMLを "Hello World" に書き換える。
        */
        document.getElementById("output").innerHTML
          = "Hello World!";
      }
    </script>
  </head>
  <body onload="main()">
    <h1>Sample Page 3<br/>(Hello World)</h1>
    <!-- この部分を書き換える -->
    <div id="output"></div>
  </body>
</html>
```

Sample Page 3
(Hello World)

Hello World!

JavaScriptとDOM



- DOMとは

Document Object Modelの略で、
Webページで表示する内容を
プログラムから利用するための機能。
ページの**HTML**の構造を書き換え、
表示内容を変更します。

Hello World (2-1)

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="utf-8"/>
    <title>Sample Page 3(Hello World)</title>
    <script type="text/javascript" src="sample3-2.js"></script>
  </head>
  <body onload="main()">
    <h1>Sample Page 3<br/>(Hello World)</h1>
    <!-- この部分を書き換える -->
    <div id="output"></div>
  </body>
</html>
```

sample3-2.html

表示(HTML)と
プログラム(JS)を分離

```
// <body onload="main()"> により
// ページ全体が読み込まれたとき(onload)に
// 関数 main() を実行する。
function main() {
  /*
  <div id="output"></div>の場所をみつけて(getElementById())
  その内側innerHTMLを"Hello World"に書き換える。
  */
  document.getElementById( "output" ).innerHTML = "Hello World!" ;
}
```

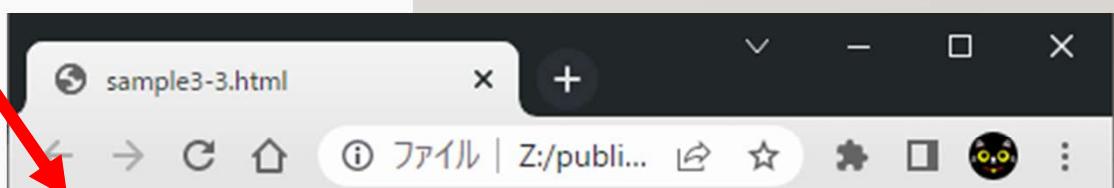
sample3-2.js

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="utf-8"/>
    <title>Sample Page 3(Hello World)</title>
    <script type="text/javascript">
      // <body onload="main()"> により
      // ページ全体が読み込まれたとき(onload)に
      // 関数 main() を実行する。
      function main() {
        document.write(
          "<h1>見出し</h1>" +
          "<p>段落</p>"
        );
      }
    </script>
  </head>
  <body onload="main()">
    <h1>Sample Page 3<br/>(Hello World)</h1>
    <p>
      ページの本文
    </p>
  </body>
</html>
```

本来の見出しの Sample Page 3 (Hello World) は表示されない。

様々な出力方法 sample3-3.html

document.write(...) は、
ページの中身全体を
書き換える



見出し

段落

様々な出力方法-I

sample3-4.html

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="utf-8"/>
    <title>Sample Page 3(Hello World)</title>
    <script type="text/javascript">
      // <body onload="main()"> により
      // ページ全体が読み込まれたとき(onload)に
      // 関数 main() を実行する。
      function main() {
        document.getElementById( "output" ).innerHTML = "ページに表示" ;
        console.log( "コンソールに表示" ) ;
        alert( "アラートで表示" ) ;
      }
    </script>
  </head>
  <body onload="main()">
    <h1>Sample Page 3<br/>(Hello World)</h1>
    <div id="output"></div>
  </body>
</html>
```

ポップアップの
警告画面で表示



様々な出力方法-2

sample3-4.html

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="utf-8"/>
    <title>Sample Page 3(Hello World)</title>
    <script type="text/javascript">
      // <body onload="main()"> により
      // ページ全体が読み込まれたとき(onload)
      // 関数 main() を実行する。
      function main() {
        document.getElementById("output")
          .innerHTML = "コンソールに表示";
        alert("アラートで表示");
      }
    </script>
  </head>
  <body>
    <div id="output"></div>
  </body>
</html>
```



デベロッパツールを表示

A screenshot of the Chrome DevTools Console tab. The console output shows the message "コンソールに表示" (Console output) and "アラートで表示" (Alert output). The 'Console' tab is selected. A red box highlights the '選択したコンテンツのみ' (Only selected content) checkbox, which is checked. Another red box highlights the 'コンソールに表示' (Console output) button at the bottom of the list. A red arrow points from the 'Console' tab in the browser toolbar to the 'Console' tab in the DevTools. A red circle highlights the gear icon in the DevTools toolbar.

Sample Page 3
(Hello World)

要素 コンソール ソース Performance insights

1件の問題: F1

選択したコンテンツのみ

コンソールに表示

sample3-4.html:12

基本的な型(I)

sample4.html

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="utf-8"/>
    <title>Sample Page 4(基本的な型)</title>
    <script type="text/javascript">
      function main() {
        let num = 112233 ; // 数値
        let str = "abcdefg" ; // 文字列
        let array = [ 1 , 2 , 3 , 4 , 5 ] ; // 

        // + 演算子は、数値の加算 or 文字列の連結
        document.getElementById( "output" ).innerHTML
          = "number="+num+"<br/>" +
          + "string="+str+"<br/>" +
          + "array="+array+"<br/>" ;
      }
    </script>
  </head>
  <body onload="main()">
    <h1>Sample Page 4<br/>(基本的な型)</h1>
    <div id="output"></div>
  </body>
</html>
```

Sample Page 4 (基本的な型)

number=112233
string=abcdefg
array=1,2,3,4,5

基本的な型(2)

```
<!DOCTYPE html>
<html>
  <head>
    <title>4(基本的な型)</title>
    <script type="text/javascript">
      function main() {
        let num = 112233 ;
        let str = "abcdefg" ;
        let array = [ 1 , 2 , 3 , 4 , 5 ] ;
        // 数値
        // 文字列
        // 
        // + 演算子は、数値の加算 or 文字列の連結
        document.getElementById( "output" ).innerHTML
          = "number="+num+"<br/>" +
          + "string="+str+"<br/>" +
          + "array="+array+"<br/>" ;
      }
    </script>
  </head>
  <body>
    <h1>4(基本的な型)</h1>
    <p>“+”で文字列で繋げて  
書き並べてみた。</p>
  </body>
</html>
```

Sample Page 4 (基本的な型)

number=112233
string=abcdefg
array=1,2,3,4,5

JavaScript ▾

Enter a title

File0 ✘ +

基本的な型(3) Paiza.IO で動作確認

```
1 process.stdin.resume() ;
2 process.stdin.setEncoding('utf-8') ;
3
4 let num = 112233 ;
5 let str = "abcdefg" ;
6 let array = [ 1,2,3,4,5 ] ;
7
8 console.log( "number="+num ) ;
9 console.log( "string="+str ) ;
10 console.log( "array="+array ) ;
```

➡ 実行 (Ctrl-Enter)

◀ JavaScriptを学ぶ | プログラミング力診断

出力 入力 コメント 0

number=112233

string=abcdefg

array=1,2,3,4,5

制御構文(I)

sample5.html

```
<!DOCTYPE html>
<html lang="ja">
<head>
    <meta charset="utf-8"/>
    <title>Sample Page 5(制御構文)</title>
    <script type="text/javascript">
        function main() {
            let array = [ 1 , 2 , 3 , 4 , 5 , 6 ] ; /* 配列 */

            let sum = 0 ;
            let step = "" ;

            for( let i = 1 ; i <= 10 ; i++ ) {
                sum = sum + i ;
                // sum += i ; と書いてもいい
                step = step + ": " + sum + "," + i + "<br/>" ;
                // step += ": "...と書いてもいい
            }
            document.getElementById( "output" ).innerHTML
                = step + "合計=" + sum ;
        }
    </script>
</head>
<body onload="main()">
    <h1>Sample Page 5<br/>(制御構文)</h1>
    <div id="output"></div>
</body>
</html>
```

Sample Page 5 (制御構文)

: 1,1
: 3,2
: 6,3
: 10,4
: 15,5
: 21,6
: 28,7
: 36,8
: 45,9
: 55,10
合計=55

制御構文(I) Paiza.IO で for文

```
1 process.stdin.resume() ;
2 process.stdin.setEncoding('utf-8') ;
3
4 let sum = 0 ;
5 for( let i = 1 ; i <= 10 ; i++ ) {
6     sum = sum + i ;
7     console.log( "i=" + i + " , sum=" + sum ) ;
8 }
9 console.log( "合計=" + sum ) ;
```

実行 (Ctrl-Enter)

JavaScriptを学ぶ | プログラミング力診断

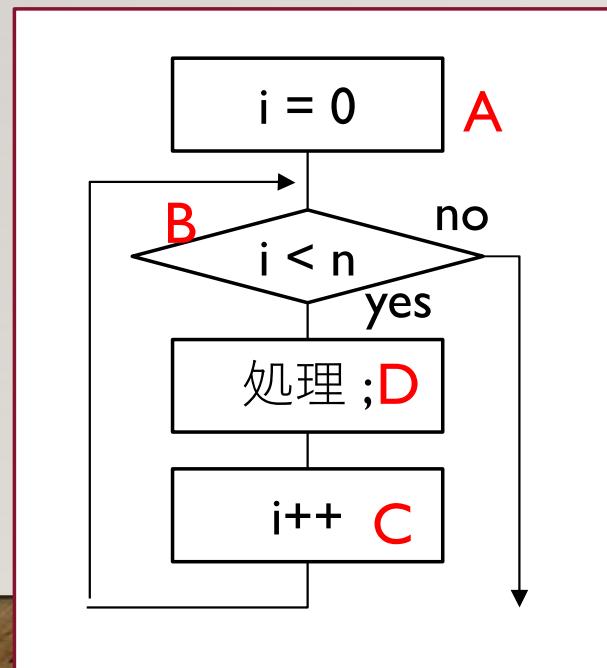
出力 入力 コメント 0

```
i=1 , sum=1
i=2 , sum=3
i=3 , sum=6
i=4 , sum=10
i=5 , sum=15
i=6 , sum=21
i=7 , sum=28
i=8 , sum=36
i=9 , sum=45
i=10 , sum=55
合計=55
```

制御構文(2)

数値を変化させながらの繰り返し

```
for( A i = 0 ; B i < n ; C i++ ) {  
    処理 ;  
} D
```



簡単なプログラムは
Paiza.IO で動作確認

The screenshot shows a code editor window for Paiza.IO. The code is:

```
1 process.stdin.on('data', function(data){  
2     process.stdout.write(data);  
3     // Your code here!  
4  
5     let i ;  
6     for( i = 0 ; i < 3 ; i++ ) {  
7         console.log( "i = " + i );  
8     }  
9 }
```

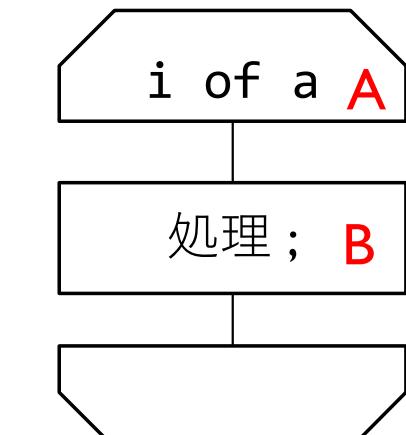
Below the code, there are buttons for "実行 (Ctrl-Enter)" and "JavaScriptを学ぶ | プログラミング". The output pane shows the results of the execution:

出力
i = 0
i = 1
i = 2

制御構文(3)

配列各要素の繰り返し処理

```
let a = [ 11 , 22 , 33 , 44 ] ;  
for( i of a ) {  
    处理 ;  
}
```



簡単なプログラムは
Paiza.IO で動作確認

The screenshot shows a code editor with the following JavaScript code:

```
4  
5 let a = [ 11 , 22 , 33 , 44 ] ;  
6 for( let i of a ) {  
7     console.log( "i = " + i ) ;  
8 }  
9
```

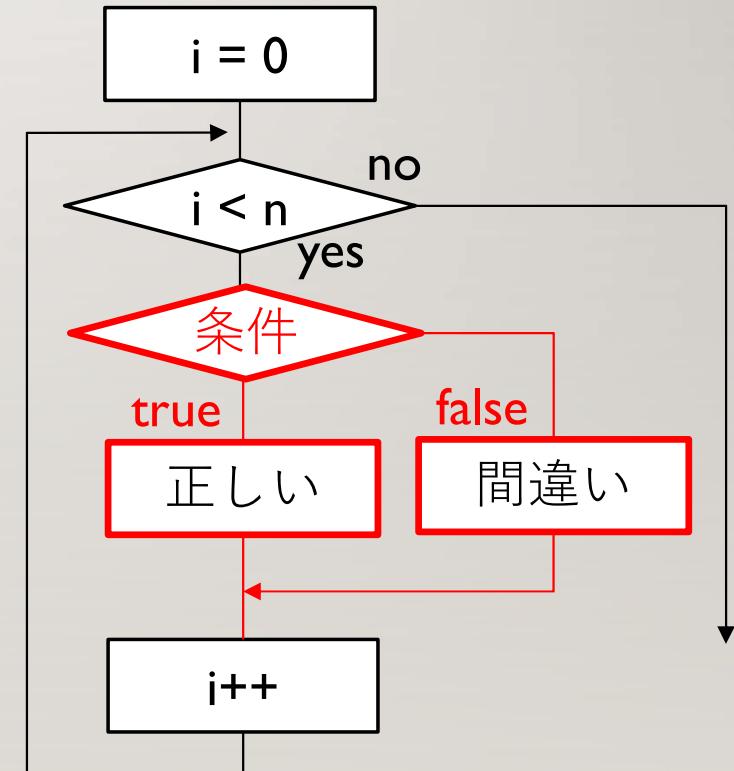
Below the code editor is a green button labeled "実行 (Ctrl-Enter)". To the right of the button is the text "JavaScriptを学ぶ | プログ". The output window below shows the results of the console.log statements:

i = 11
i = 22
i = 33
i = 44

制御構文(4)

条件判定

```
for( i = 0 ; i < n ; i++ ) {  
    条件  
    if ( i % 2 == 0 ) {  
        正しい時の処理 ; true  
    } else {  
        間違い時の処理 ; false  
    }  
}
```



平方根の表を表示

簡単なプログラムは
Paiza.IO で動作確認

```
1 process.stdin.resume();
2 process.stdin.setEncoding('utf8');
3
4 // 平方根の表
5 for( let _____ ; _____ ; _____ ) {
6     console.log( "sqrt(" + i + ") = "
7         + Math.sqrt( _____ ) );
8 }
9
```

実行 (Ctrl-Enter)

Java

Math.sqrt(式) で
平方根を求める

出力 入力 コメント 0

```
sqrt(1) = 1
sqrt(2) = 1.4142135623730951
sqrt(3) = 1.7320508075688772
sqrt(4) = 2
sqrt(5) = 2.23606797749979
sqrt(6) = 2.449489742783178
sqrt(7) = 2.6457513110645907
sqrt(8) = 2.8284271247461903
sqrt(9) = 3
```

練習問題(平方根の表)

九九の表を表示

```
1 process.stdin.resume();
2 process.stdin.setEncoding('utf8');
3 // Your code here!
4
5 for( let i = 1 ; i < 10 ; _____ ) {
6     let line = i + ":" ;
7
8     for( let j = 1 ; _____ ; j++ ) {
9
10        line += ( _____ ).slice(-2) + " " ;
11    }
12    console.log( line );
13 }
14
```

2文字表示にする

実行 (Ctrl-Enter)

JavaScriptを学ぶ | プログラミング力診断

出力 入力 コメント 0

```
1:  1  2  3  4  5  6  7  8  9
2:  2  4  6  8 10 12 14 16 18
3:  3  6  9 12 15 18 21 24 27
4:  4  8 12 16 20 24 28 32 36
5:  5 10 15 20 25 30 35 40 45
6:  6 12 18 24 30 36 42 48 54
7:  7 14 21 28 35 42 49 56 63
8:  8 16 24 32 40 48 56 64 72
9:  9 18 27 36 45 54 63 72 81
```

練習問題(九九の表)

制御構文(5)

関数と実引数,仮引数の値の受け渡し

```
関数名 仮引数1 仮引数2
function foo( v1 , v2 ) {
    let 変数1 , 変数2 ;
    // letで宣言した変数や仮引数は、
    // この関数fooの中で有効

    return v1 + v2 ;
}

var ans = foo( 123 , "abc" );
        実引数1 実引数2
```

関数foo()を呼び出すと...

仮引数1 = 実引数1 ;
v1 = 123 ;
仮引数2 = 実引数2 ;
v2 = "abc" ;

をしてから、関数の中身を実行。

return **返り値** ;
答えを持って帰る。

ans = "123abc" ;

```

<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="utf-8"/>
    <title>Sample Page 6(練習問題)</title>
    <script type="text/javascript">
      // array に入っている値で "*" を使った棒グラフを表示したい。
      // プログラムの _____ 部分にふさわしい命令を考えよう。

      // str を n 回繰り返した文字を作る関数
      function strtimes( str , n ) {
        let ans = "" ;
        for( let i = 1 ; _____ ; _____ )
          ans += _____ ;
        return ans ;
      }
      function main() {
        // グラフにしたいデータ
        let array = [ 1 , 3 , 5 , 9 , 10 , 6 , 3 , 2 ] ; /* 配列 */
      }

      // 配列全部を順次棒グラフにする。
      let out = "" ;
      for( let n of array ) {
        out += " " + strtimes( _____ , _____ ) + "<br/>" ;
        // ("00"+n).slice(-2) を使うと、数字の前に0を埋めて2桁表示してくれる。
      }
      document.getElementById( _____ ).innerHTML
        = _____ ;
    }
  </script>
</head>
<body onload="_____">
  <h1>Sample Page 6(練習問題)</h1>
  <h2>棒グラフ</h2>
  <div id="output"></div>
</body>
</html>

```

桁数が揃ってなくて
見づらいよね

// array に入っている値で "*" を使った棒グラフを表示したい。
// プログラムの _____ 部分にふさわしい命令を考えよう。

// str を n 回繰り返した文字を作る関数

function strtimes(str , n) {
 let ans = "" ;
 for(let i = 1 ; _____ ; _____)
 ans += _____ ;
 return ans ;
}
function main() {
 // グラフにしたいデータ
 let array = [1 , 3 , 5 , 9 , 10 , 6 , 3 , 2] ; /* 配列 */

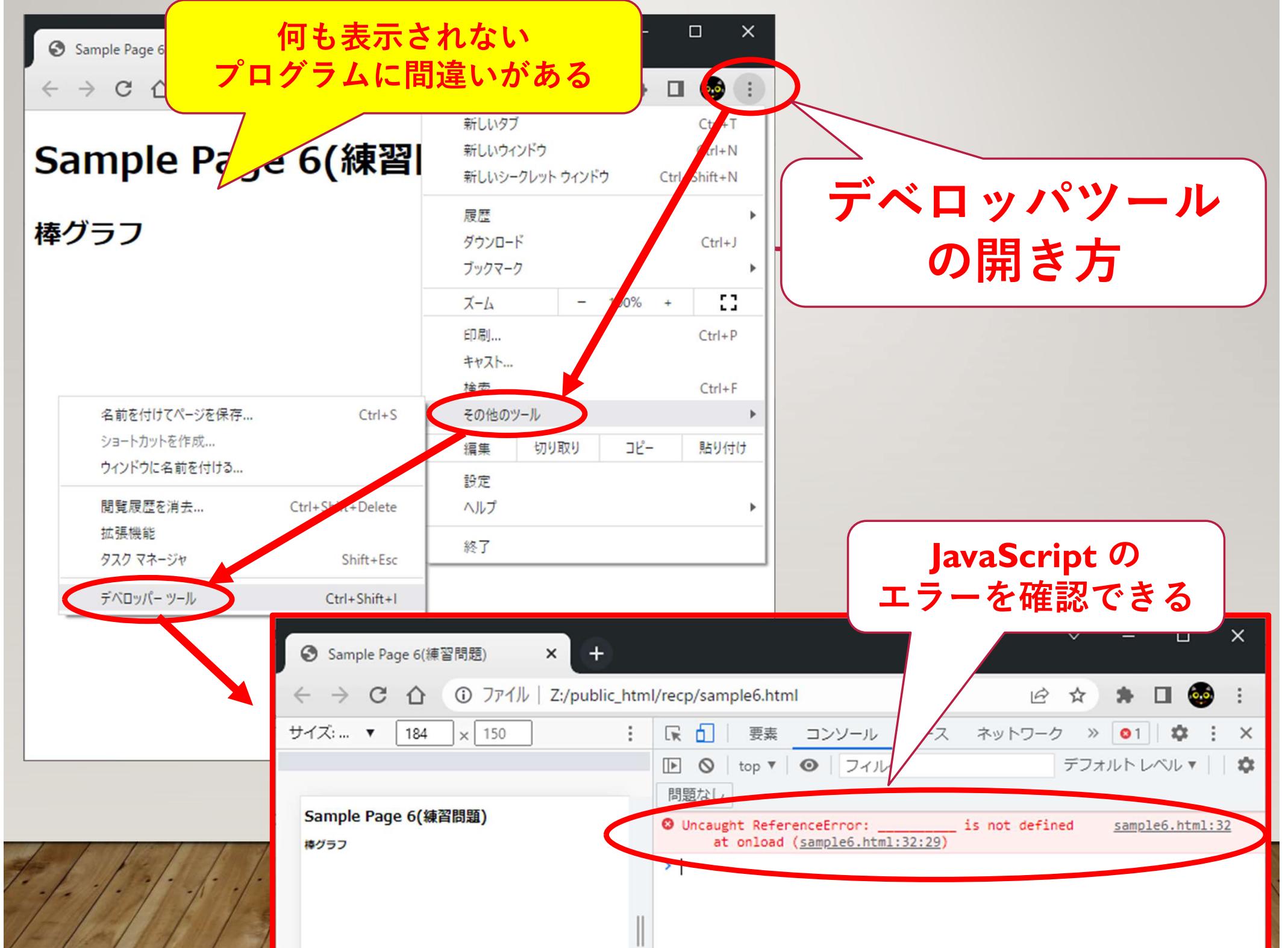
// 配列全部を順次棒グラフにする。

let out = "" ;
for(let n of array) {
 out += " " + strtimes(_____ , _____) + "
" ;
 // ("00"+n).slice(-2) を使うと、数字の前に0を埋めて2桁表示してくれる。

document.getElementById(_____).innerHTML
= _____ ;

1 : *
3 : ***
5 : *****
9 : *****
10 : *****
6 : *****
3 : ***
2 : **

練習問題(棒グラフ)
sample6.html





INTERMISSION

引用：風と共に去りぬ

オブジェクトを使ったプログラム



// 連想配列の初期化

```
let tsaitoh = {  
    "name": "齊藤 徹" ,  
    "age": 57 ,  
    "addr": "福井県越前市" ,  
} ;
```



名前： 齊藤 徹 ,
年齢： 57 ,
住所： 福井県越前市 ,

オブジェクトと連想配列(I)

// 連想配列の初期化

```
let tsaitoh = {  
    "name": "齊藤 徹" ,  
    "age": 57 ,  
    "addr": "福井県越前市" ,  
} ;
```

// 連想配列形式での参照方法

```
tsaitoh[ "name" ] 齊藤徹  
tsaitoh[ "age" ] 57  
tsaitoh[ "addr" ] 福井県越前市
```



名前： 齊藤 徹 ,
年齢： 57 ,
住所： 福井県越前市 ,

// オブジェクト形式での参照方法

```
tsaitoh.name 齊藤徹  
tsaitoh.age 57  
tsaitoh.addr 福井県越前市
```

sample7.html

オブジェクトと連想配列(2)

```
// 連想配列のキーを参照する繰り返し
for( let key in 連想配列 ) {
    key ... 連想配列[ key ] ;
}
```

```
((作りたいHTML)))
<table border='1'>
    <tr><th>name</th>
        <td>齊藤 徹</td></tr>
    <tr><th>age</th>
        <td>56</td></tr>
    <tr><th>addr</th>
        <td>福井県越前市</td></tr>
</table>
```

```
let tsaitoh = {
    "name": "齊藤 徹" ,
    "age": 56 ,
    "addr": "福井県越前市" ,
} ;   key      tsaitoh[key]
```

```
text = "<table border='1'>" ;
for( let key in tsaitoh ) {
    text += "<tr>
        + "<th>" +key+ "</th>" +
        + "<td>" +tsaitoh[key]+ "</td>" +
        + "</tr>" ;
}
text += "</table>" ;

document.getElementById( "output" )
    .innerHTML = text ;
```

連想配列

Paiza.IO で確認

```
1 process.stdin.resume() ;
2 process.stdin.setEncoding('utf-8') ;
3
4 let tsaitoh = {
5   "name": "斎藤 徹",
6   "age": 57,
7   "addr": "福井県越前市",
8 }
9 for( let key in tsaitoh ) {
10   console.log( "key=" + key + " , value=" + tsaitoh[ key ] );
11 }
```

実行 (Ctrl-Enter)

JavaScriptを学ぶ | プログラミング力診断

出力

入力 コメント 0

key=name , value=斎藤 徹

key=age , value=57

key=addr , value=福井県越前市

オブジェクトの配列 (I)

```
let item_list = [  
    { "id":1010, "name":"みかん", "price":50 } ,  
    { "id":1020, "name":"りんご", "price":100 } ,  
    { "id":1022, "name":"パイナップル", "price":1000 } ,  
];
```

item_list

The diagram illustrates the variable `item_list` pointing to a table of items. A yellow arrow points from the variable name to the opening bracket of the array. A dashed blue box encloses the table, and another yellow arrow points from the word `item` to the first row of the table, specifically highlighting the `id` column.

| | id | name | price |
|------|------|--------|-------|
| item | 1010 | みかん | 50 |
| | 1020 | りんご | 100 |
| | 1022 | パイナップル | 1000 |

```
text = "" ;  
// 各行毎の処理  
for( let item of item_list ) {  
    text += "<tr>"  
    + "<td>" + item.id + "</td>"  
    + "<td>" + item.name + "</td>"  
    + "<td>" + item.price + "</td>"  
    + "</tr>" ;  
}
```

オブジェクトの配列

Paiza.IO で確認

```
1 process.stdin.resume() ;
2 process.stdin.setEncoding('utf-8') ;
3
4 let item_list = [
5   { "id":1010, "name":"みかん", "price":50 } ,
6   { "id":1020, "name":"りんご", "price":100 } ,
7   { "id":1022, "name":"パイナップル", "price":1000 } ,
8 ];
9
10 for( let item of item_list ) {
11   console.log( "id=" , item.id ) ;
12   console.log( "name=" , item.name ) ;
13   console.log( "price=" , item.price ) ;
14 }
```

出力 入力 コメント 0

id= 1010
name= みかん
price= 50
id= 1020
name= りんご
price= 100
id= 1022
name= パイナップル
price= 1000

```

let item_list = [
  { "id" : 1010 , "name": "みかん" , "price" : 50 } ,
  { "id" : 1020 , "name": "りんご" , "price" : 100 } ,
  { "id" : 1022 , "name": "パイナップル" , "price" : 1000 } ,
];
let buy_list = {
  1010 : 5 , // みかん×5個
  1020 : 3 , // りんご×3個
  1022 : 1 , // パイナップル×1個
} ;
let sum = [REDACTED] ;
let text = "<table border='1'>" ;
// 表の属性を表示
text += "<tr>" +
  "<th>id</th><th>name</th>" +
  "<th>price</th><th>個数</th></tr>" ;
for( let item of [REDACTED] ) {
  // item.id の購入数
  let buy = [REDACTED] ;
  // 行の表示
  text += "<tr>" +
    "<td>" + item.id + "</td>" +
    "<td>" + item.name + "</td>" +
    "<td>" + item.price + "</td>" +
    "<td>" + [REDACTED] + "</td>" +
    "</tr>" ;
  // @単価 * 購入数
  [REDACTED] += [REDACTED] ;
}
text += "</table>" ;
text += "合計=" + sum ;
document.getElementById( "output" ).innerHTML = text ;

```

Sample Page 9 (配列との串刺し)

| id | name | price | 個数 |
|------|--------|-------|----|
| 1010 | みかん | 50 | 5 |
| 1020 | りんご | 100 | 3 |
| 1022 | パイナップル | 1000 | 1 |

合計=1550

練習問題
sample9.html

HTML の中に JavaScript 呼出し を埋め込む

```
<body onload="main(...)"> ... </body>
```

- HTMLのページ読み込みが完了したら
main(...) 関数を実行する。



```
<input type="submit" onclick="action(...)" />
```

- ボタンがクリックされたら
action(...) 関数を実行する。



```
<input type="text" onchange="action(...)" />
```

- 記入内容が変化したら
action(...) 関数を実行する。



```
// 商品データの1行分のHTMLを作る
function item_row( item ) {
    let ans ;
    (略)
    ans += ...
    + "<input type='text' size='3' class='COUNT'" +
    + "onchange='sum_items()' id='"+item.id+"' />" +
    + "</td>" ;
    return ans ;
}
```

onchange="関数()"
入力欄で値が
書き換えられた

```
// 個数の入力欄が書き換えられたら呼び出される処理
function sum_items() {
    let sum = 0 ;
    for( let item of item_list ) {
        let num = document.getElementById( item.id ).value ;
        (略)
    }
    document.getElementById( "sum_items" ).innerHTML =
        = "合計="+sum+"円"
        + " <input type='submit' value='発注' class='SUBMIT'" +
        + "onclick='submit_order()' />" ;
}
```

```
// 発注ボタンで呼び出される処理
function submit_order() {
    (略)
    for( let item of item_list ) {
        let num = document.getElementById( item.id ).value ;
        (略)
    }
    alert( "order" + order_text ) ;
}
```

onclick="関数()"
ボタンがクリックされた

alert(...);
 ポップアップ表示

発注ページ sampleA.html

| 注文ID | 商品名 | 単価 | 個数 |
|----------|---------|------|----|
| 1010 | みかん | 50 | 1 |
| 1020 | りんご | 100 | 2 |
| 1022 | パインアップル | 1000 | 3 |
| 合計=3250円 | | | 発注 |

個数の書き換えで
● 合計を計算
● 発注ボタン表示

おまけ 無名関数

```
1 process.stdin.resume();
2 process.stdin.setEncoding('utf8');
3
4 function add( x , y ) { // map 関数の処理
5     return x + y ;
6 }
7 function mul( x , y ) { // v = list[0] ;
8     return x * y ;
9 }
10
11 function my_map( list , func ) {
12     let v = list[ 0 ] ;
13     for( let i = 1 ; i < list.length ; i++ ) {
14         v = func( v , list[ i ] ) ;
15         console.log( "func( " + v + " , " + list[i] + ")" ) ;
16     }
17     return v ;
18 }
19
20 console.log( my_map( [ 4,3,2 ] , add ) ) ; // 足し算 add を渡す
21 console.log( my_map( [ 4,3,2 ] , mul ) ) ; // 掛け算 mul を渡す
22
23 console.log( my_map( [ 4,3,2 ] ,
24     function( x , y ) { // 無名関数
25         return x + y ;
26     } ) ) ; // 関数をこの場所で定義する
27
28 console.log( my_map( [ 4,3,2 ] ,
29     ( x , y ) => x * y ) ) ; // アロー関数
```

コールバック関数
を呼び出す

出力 入力 コメント 0

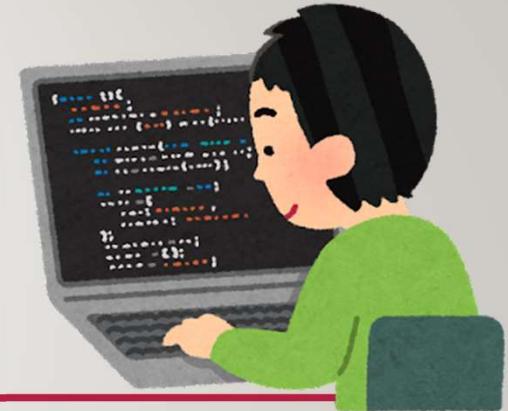
```
func( 7 , 3) add(...)
func( 9 , 2)
9
func( 12 , 3)
func( 24 , 2) mul(...)
24
func( 7 , 3)
func( 9 , 2)
9
func( 12 , 3)
func( 24 , 2)
24
```



INTERMISSION

引用：風と共に去りぬ

jQueryの使い方



- jQueryは、ウェブブラウザ用のJavaScriptコードをより容易に記述できるようにするための機能。

```
<div class="name">齊藤 徹</div>
$( function() {
    $("div.name").css( "color", "red" );
} );
```

jQuery は `$(...)` 関数名が “\$”

赤に書き換わる

divのclass=”name”的CSSで
文字の色を赤に変更

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="utf-8"/>
    <title>Sample Page 11(jQueryの基本)</title>
    <script src="jquery/jquery.min.js"></script>
  </head>
  <body>
    <h1>Sample Page 11<br/>(jQueryの基本)</h1>
    <div id="output"></div>
    <div class="word">こんにちは</div>
    <div class="hilight">今日は晴れています。</div>
    <div class="word">こんばんは</div>
    <div class="hilight">今日は月が綺麗だな。</div>
    <script type="text/javascript">
      $(function () {
        // タグの中を書き換え
        $("div#output").text( "Hello World" ) ;
        // タグの中のスタイルを変更
        $("div.word").css( "color", "red" ) ;
        $("div.hilight").hover(
          function() { $(this).css( "background-color" , "yellow" ) ; } ,
          function() { $(this).css( "background-color" , "white" ) ; } )
      })
    </script>
  </body>
</html>
```

jQueryの基本 sampleB.html

jqueryの命令を
読み込む設定
sampleプログラムの
フォルダに置いてある

書き換える場所に
id属性,class属性で目印

指定した場所の
内容を変更

JSONデータの読み込み



- JSONは、サーバとデータをやり取りするためのデータ形式。
- JavaScriptで読み取りやすい。
- jQueryを使うと、JSON読み込みを簡単に書ける。

```
$( function() {  
    $.getJSON( “ファイル場所” , function( data ) {  
        // data : 読み込んだJSONデータ  
        // ここに処理を書く  
    }) ;  
}) ;
```

JSONデータの読み込み sampleC.html

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="utf-8"/>
    <title>Sample Page 12(JSONデータの読み込み)</title>
    <script src="jquery/jquery.min.js"></script>
    <script type="text/javascript">
      $(function() {
        $.getJSON( "sampleC.json" , function(item_list) {
          let text = "<table border='1'>" ;
          text += "<tr><th>id</th><th>name</th><th>price</th></tr>" ;
          for( let item of item_list ) {
            text += "<tr>" +
              + "<td>" + item.id + "</td>" +
              + "<td>" + item.name + "</td>" +
              + "<td>" + item.price + "</td></tr>" ;
          }
          document.getElementById( "output" ).innerHTML
            = text ;
        })
      })
    </script>
  </head>
  <body>
    <h1>Sample Page 12<br/>(JSONデータの読み込み)</h1>
    <div id="output"></div>
  </body>
</html>
```

jQueryの処理

JSONの読み込み

jQueryの処理

sample8.html を書き換えたもの

Webの公開からWebサービスの公開へ



- ・自分のホームページを作りたい。
- ・会社のホームページを作りたい。
- ・独自のWebサービスを構築し公開したい。
- ・サービス提供用のコンピュータを借りる
 - ・Amazon – AWS
 - ・Microsoft – Azure
 - ・さくらインターネット



自分のホームページを作りたい 無料サービス...

- プロバイダのWebサーバを借りる
 - HTML, CSS, JavaScript で作成
 - ページのデータをアップロード
- ブログサービスを使う
 - Ameba ブログ
 - はてなブログ

(例) ASAHIネット

- <https://www.asahi-net.or.jp/~ユーザID/>
- <https://ameblo.jp/ユーザID/>



Webサーバーの一部を借りる

PaaS (Platform as a Service)



記事を
書込む



BLOGサービスを借りる

SaaS (Software as a Service)

会社のホームページを作りたい

- WordPress をインストールしてあるサーバを借りる
 - SaaS ? / PaaS ?
- Microsoft - Azure Marketplace
- Amazon – AWS に WordPress
- 会社のドメイン名も申請



独自のWebサービスを提供したい

- サーバを借りると何がいいの

- スケールアップ

- ユーザが増えて処理能力が不足したら
 - 速度の速いサーバを借りる
 - 並列処理するサーバを何台も借りる

- サーバ管理を委託

OS, サーバソフト,
ネットワークを
自由に組み合わせ

- OS = Linux / Windows
- Web = Apache / Nginx
- 言語 = PHP / Python
- DB = Oracle / MySQL

オンプレミス
のサーバ



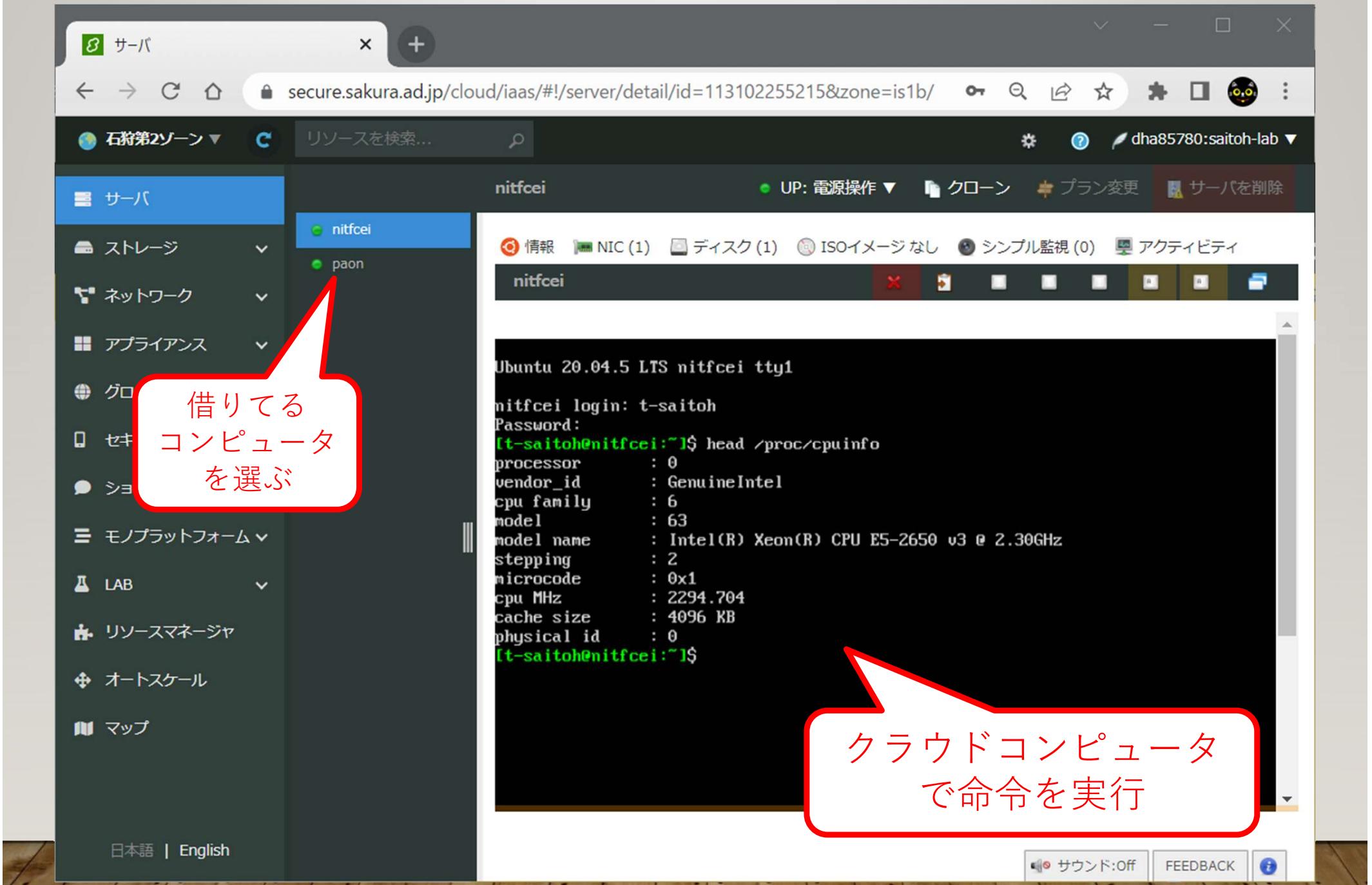
クラウド化

サーバを借りる
IaaS (Infrastructure as a Service)



さくらインターネットの例

The screenshot shows the Sakura Cloud interface for adding a new server. On the left, a sidebar lists categories like Server, Storage, Network, and Global. The main area is titled "サーバ追加" (Server Add) and shows "サーバプラン" (Server Plan) settings. Under "仮想コア" (Virtual Cores), there are two radio button options: "通常プラン" (Standard Plan) and "コア専有プラン" (Core Exclusive Plan). Below these are radio buttons for selecting the number of cores: 1, 2, 3, 4 (selected), 5, 6, 8, 10, and 12. A red speech bubble points to the core selection area with the text "CPUやメモリを選択" (Select CPU or Memory). Under "メモリ" (Memory), there are radio buttons for 4GB, 5GB, 6GB (selected), 8GB, 12GB, and 16GB. A red speech bubble points to the memory selection area with the text "月額費用" (Monthly Fee). At the bottom, a green button says "サーバプラン一覧から選択" (Select from Server Plan List). The footer displays the cost information: 時割 51円 日割 517円 月額 10,340円 月末までの予想料金 10,340円 (Hourly rate 51 yen, Daily rate 517 yen, Monthly fee 10,340 yen, Estimated end-of-month fee 10,340 yen).



まとめ

- **JavaScript** は、ブラウザの中で動くプログラム言語
- ページの内容を書き換えることができる
- **.onload, onclick, onchange** のタイミングで書き換え
- **jQuery**を使うと、指定した場所の表示変更が簡単
- **JSON**データの読み込みもできる

