



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 (11/14)

プログラミング応用 講義資料URL

<https://tsaitoh.net/~t-saitoh/2021-11-recp/>

login: guest

password: Guest

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

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\(11/07\)](#)
 - 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
 1. [理解度確認\(11/07\)](#) (Google Formsに回答してください)
 2. 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。
 3. 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、「安全が確認できないけど開きますか?」といった警告がですが、「危険性を理解したうえで開く」を実行してみてください。
 4. 2,3で確認した内容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。

サーバ設定

サーバ名

tohrusaitoh

サーバにつける名前
空白ナシの分かり易い
名前をつけてください。

初期インストール&設定するものを選択してください

※こちらにかかれていないものも手動設定は可能です。

• Web開発:

Node.js

PHP

Tomcat(Java, JSP, Servlet)

Ruby on Rails

Go(Revel)

PHP, Apache を
選んで下さい。

• データベース:

MySQL

PostgreSQL

MongoDB

phpMyAdmin

• アプリケーション:

WordPress

Jupyter Notebook

• サーバ:

Apache

☐ 常時起動(ベーシックプランのみ)

☐ SSH利用

サーバタイプ: free, リージョン: tokyo

新規サーバ作成

キャンセル

教材データのダウンロード

ターミナル画面では、サーバで動かす命令を入力
(先頭の\$は入力しなくていい)

Webページ用フォルダに移動して

\$ cd public_html

この講義のサーバ用資料をまとめてダウンロード

\$ git clone https://github.com/tohrusaitoh/recp.git



```
~/public_html
~$ cd public_html
~/public_html$ git clone https://github.com/tohrusaitoh/recp.git
Cloning into 'recp'...
remote: Enumerating objects: 52, done.
remote: Counting objects: 100% (52/52), done.
remote: Compressing objects: 100% (39/39), done.
remote: Total 52 (delta 15), reused 46 (delta 9), pack-reused 0
Unpacking objects: 100% (52/52), done.
~/public_html$
```



メッセージを残します

本日の目標

JAVASCRIPTによる フロントエンドプログラミング(11/14)

- 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
- 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	パイナップル	1000	3
合計=3250円			発注

JAVASCRIPTによる フロントエンドプログラミング(11/14)

商品一覧 HTMLファイル

注文ID	商品名	単価	個数
1010	みかん	50	5 個
1020	りんご	100	2 個
1022	パイナップル	1000	1 個
合計 1450 円			

1. JSON形式をもらってHTMLを生成
2. 買い物結果を送信

フロントエンド

HTMLとJavaScript
プログラムを送信

商品データベース

データ送信
JSON形式

データ受信
JSON形式

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

バックエンド

購入結果

id	name
1010	5
1020	2
1022	1

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” ;
```

```
<!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>
```

Hello World (I)

sample3.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、
表示結果をコントロールします。



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)

let 変数名 = 初期値 ;
型: 数値, 文字列, 配列

```
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/>" ;
```

“+”で文字列で繋げて
書き並べてみた。

Sample Page 4 (基本的な型)

```
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>
```

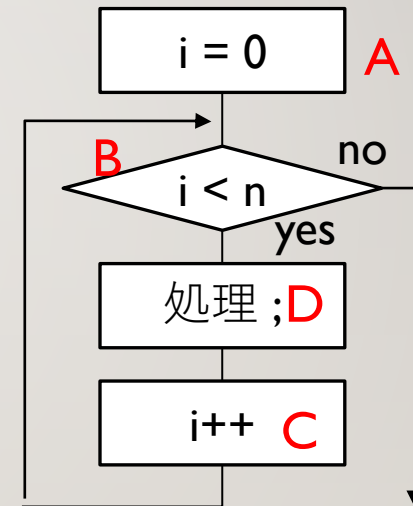
Sample Page 5 (制御構文)

: 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(2)

数値を変化させながらの繰り返し

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

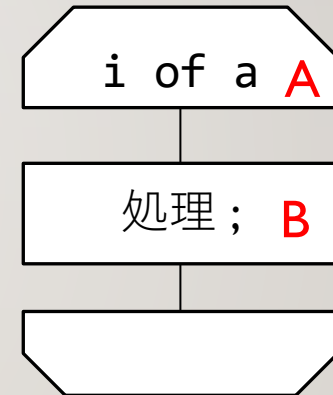


制御構文(3)

配列各要素の繰り返し処理

```
let a = [ 11 , 22 , 33 , 44 ] ;
```

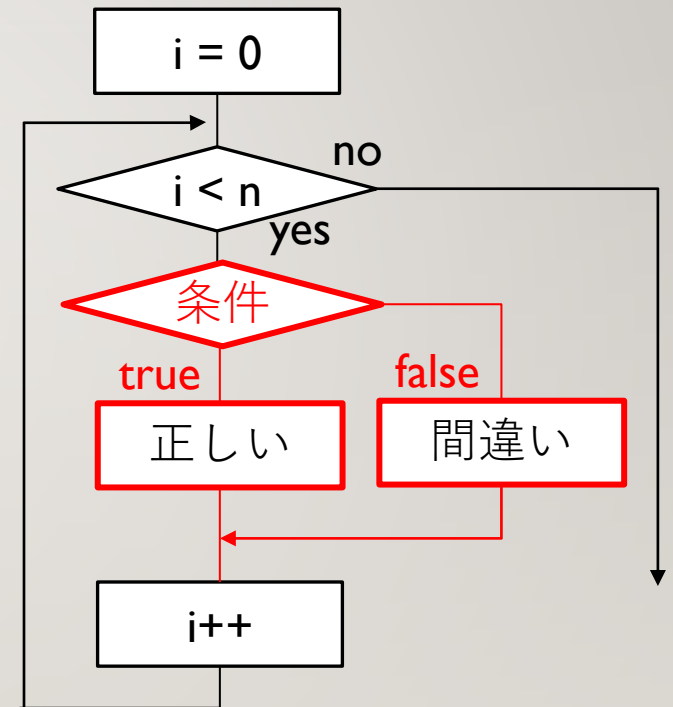
```
for( i of a ) {  
    処理 ;  
}
```



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  
    }  
}
```



制御構文(5)

関数と実引数,仮引数の値の受け渡し

```
function foo( 仮引数1 仮引数2 ) {  
  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 += n + " : " + 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

```

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

```

練習問題(棒グラフ)
sample6.html

オブジェクトと連想配列(1)

// 連想配列の初期化

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



名前： 斉藤 徹 ,
年齢： 56 ,
住所： 福井県越前市 ,

// 連想配列形式での書き方

```
tsaitoh[ "name" ] 斉藤徹  
tsaitoh[ "age" ] 56  
tsaitoh[ "addr" ] 福井県越  
前市
```

// オブジェクト形式での書き方

```
tsaitoh.name 斉藤徹  
tsaitoh.age 56  
tsaitoh.addr 福井県越前市
```

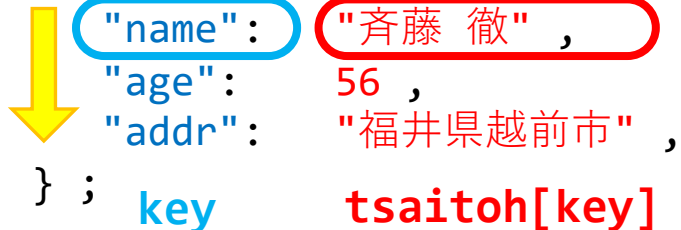
オブジェクトと連想配列(2)

```
// 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>斉藤 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": "斉藤 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 ;
```


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

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

item

```
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>" ;  
}
```

```
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 =  ;
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  ) {
  // item.id の購入数
  let buy =  ;
  // 行の表示
  text += "<tr>"
    + "<td>"+item.id+"</td>"
    + "<td>"+item.name+"</td>"
    + "<td>"+item.price+"</td>"
    + "<td>"++"</td>"
    + "</tr>" ;
  // @単価 * 購入数
   +=  ;
}
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

発注ページ sampleA.html

```
// 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

注文ID	商品名	単価	個数
1010	みかん	50	<input type="text" value="1"/>
1020	りんご	100	<input type="text" value="2"/>
1022	パイナップル	1000	<input type="text" value="3"/>
合計=3250円			<input type="button" value="発注"/>

個数の書き換えで

- 合計を計算
- 発注ボタン表示

jQueryの基本

- 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">斉藤 徹</div>

\$(function() {

\$("div.name").css("color", "red");

});

ここの文字の色を
赤に書き換え

jQueryの基本 sampleB.html

```
<!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>
```

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

```
</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">
```

書き換えたい場所に
id属性,class属性で目印

```
$(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>
```

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

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
 - **onload** -- ページの読み込みが終わった
 - **onclick** -- マウスでクリックされた
 - **onchange** -- 入力項目が書き換えられた
- **jQuery**を使うと、指定した場所の表示変更が簡単