



---

# コンテンツメディア プログラミング実習Ⅱ

## JavaScript と各種ライブラリ

---

中村, 宮下, 斉藤, 菊池

# 本日の内容

## • 1コマ目

- Nitrous.io の使い方
- JavaScript について
- DOMについて

## • 2コマ目

- ライブラリを使ってみる
  - Google Charts
  - Google Maps
  - jQuery について



ノンプログラマのためのJavaScriptはじめの一步  
(WEB+DB PRESS plus)

Claim your Ruby  
Development Box in 60 seconds.



Open Dashboardをクリックして  
自身のメインページを開こう！

Used by developers at:





# wwwフォルダでNew Folder

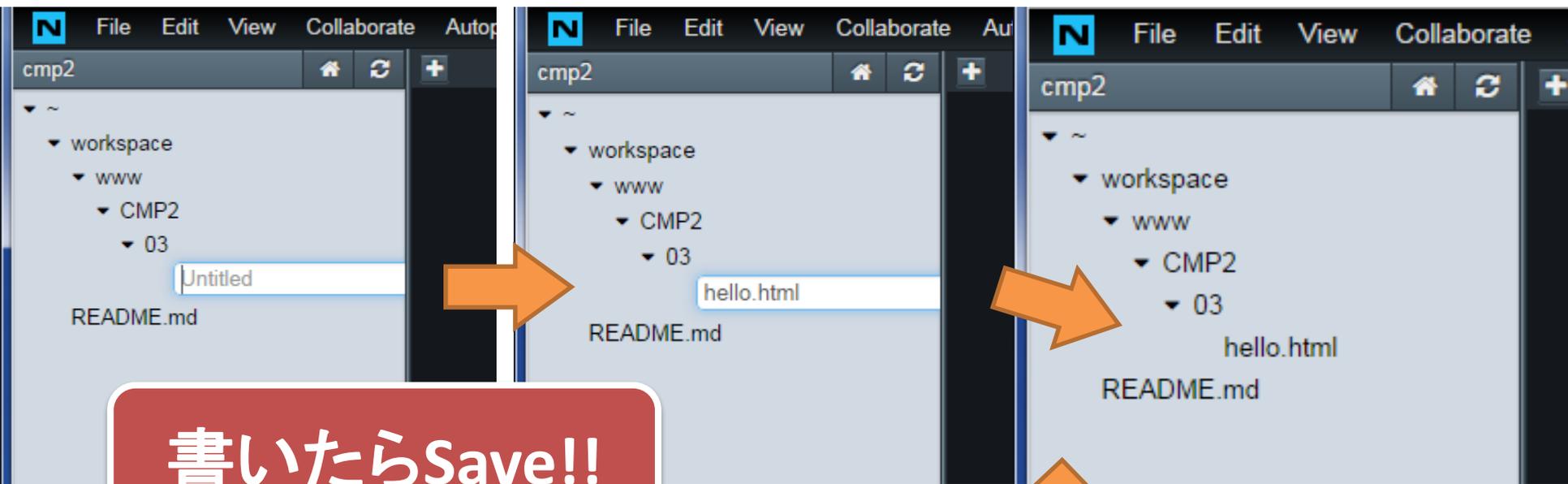
wwwフォルダを右クリック

wwwフォルダがない場合は  
近くの中村研学生へ

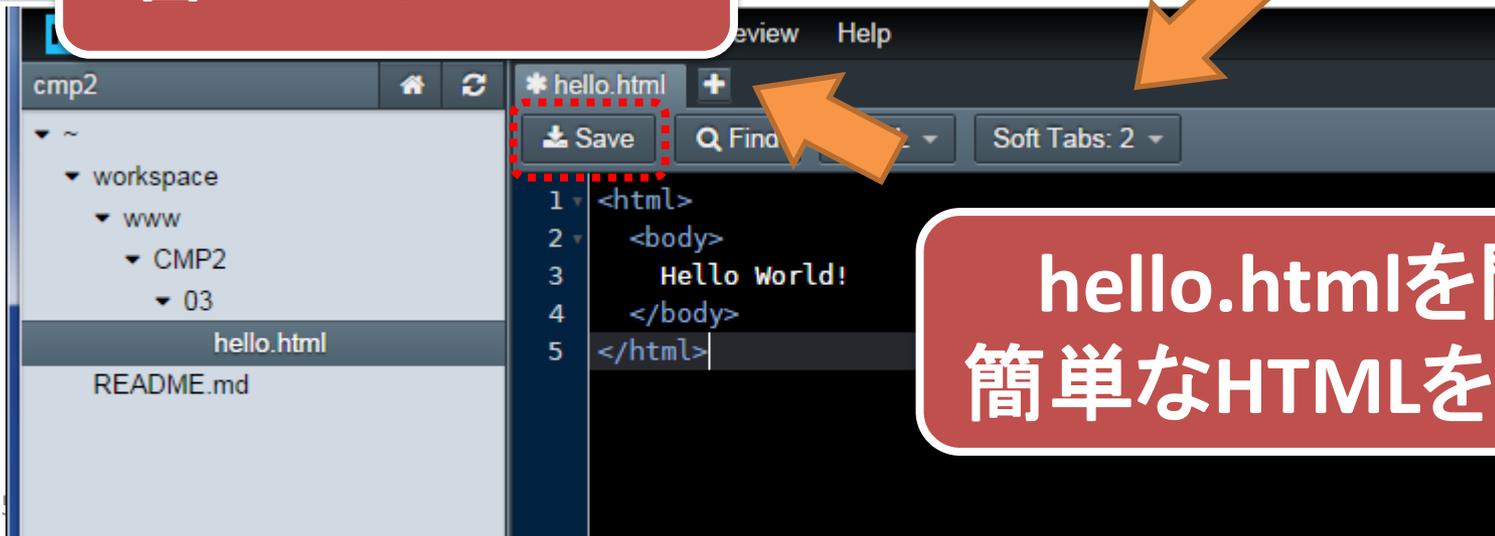
CMP2というフォルダと  
02というフォルダを作成  
そしてNew File



# hello.html を作成しよう！



書いたらSave!!



hello.htmlを開いて  
簡単なHTMLを書こう！



# Preview!!

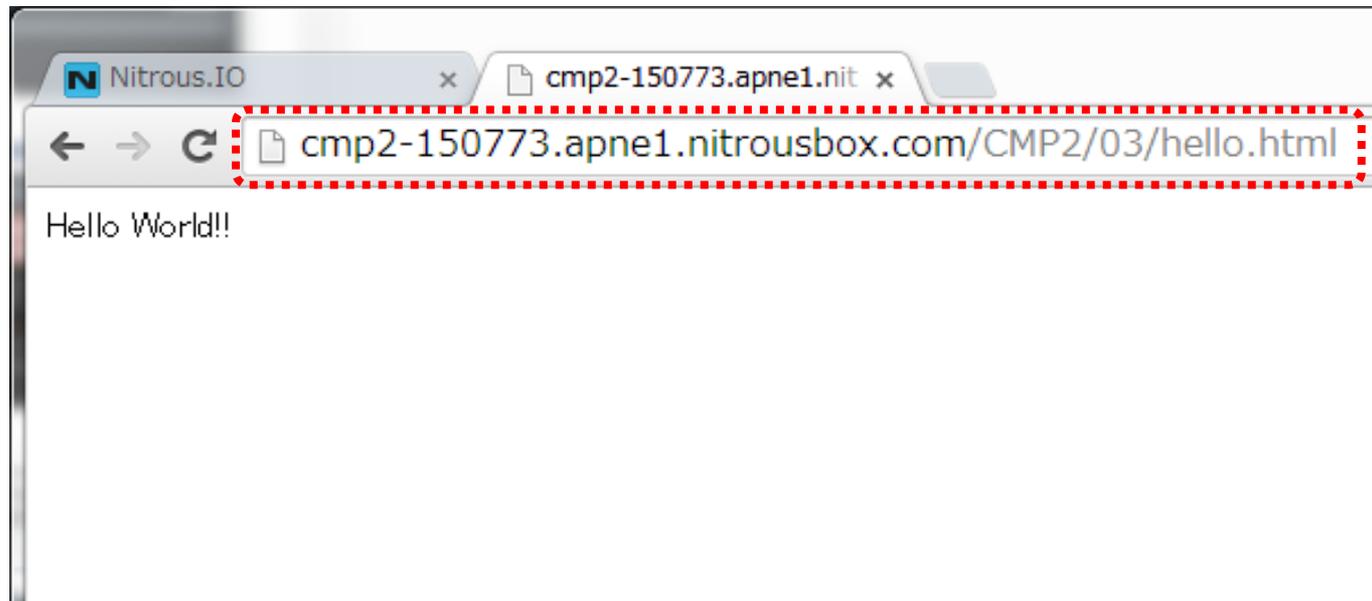
- 保存したらPreviewしてみよう！
  - Previewメニューから、Port 3000を選んで下さい

こんなページが開く！  
Index of / は www 直下のこと

リンクをクリックしてhello.htmlを開こう

# 自分のサーバ！

- 自分のサーバで，情報も公開できる！
  - URLを他人に教えることで，他の人に閲覧してもらうことも出来るよ！



# 作った課題をアップロードして閲覧！

ファイルをドロップ  
してUpload

Drop files here or  (max 20MB/file)

SS20141001\_194745.png

Remove

Cancel

Upload

適当なフォルダを作成して  
Upload Files to ... を選択

Refresh "html"

Find in "html"...

New File

New Folder

Rename

Delete "html"

Upload Files to "html"...

Set "html" As Root



# 隣の人にURLを教えよう！

- 隣の人に先週の課題で作成したファイルのURLを教え、アクセスしてもらいましょう！



# Webでユーザの操作を取得

- テキスト入力していないのに送信ボタンを押さないで！
- ユーザが入力する前には入力例を示しておいて、入力開始しようとしたら消したい！
- リアルタイムに値を取得して表示したい！
- ユーザが地図上で操作をしたら、表示内容を変更したい！

JavaScript



# JavaScript超入門

- イベントが発生した時にどうするのかを記述する言語
  - HTMLの様々な部品に対してイベントを追加していく
    - 画像にマウスがホバーされたら
    - ボタンがクリックされたら
    - 入力フォームからマウスが外れたら
  - などなど
- HTML内部で何らかのユーザ操作が行われた時に、リロードせずにページ上で動作するためのもの
  - クリックされた！じゃあ...
  - 文字入力された！じゃあ...
  - マウスカーソルが上に来た！じゃあ...
  - マウスカーソルがどこかに行った！じゃあ...



# ProcessingとJavaScript

- 同じ所: プログラミング言語なので変数, 計算, 条件分岐, 繰返し, メソッド, クラスなど同じ

## 12345の約数の数を数えるプログラム

```
int i = 1;
int count = 0;
while( i <= 12345 ){
  if( (12345 % i) == 0 ){
    // 12345をiで割った余りが
    // 0だったらcountを増やす
    count++;
  }
  i++;
}
println( "約数の数は"+count );
```

```
<script>
var i = 1;
var count = 0;
while( i <= 12345 ){
  if( (12345 % i) == 0 ){
    // 12345をiで割った余りが
    // 0だったらcountを増やす
    count++;
  }
  i++;
}
alert( "約数の数は"+count );
</script>
```



# JavaScript超入門

- スクリプトタグで囲う
- 変数の定義は var 変数名
  - intとかfloatとかの型はない(何でも一緒)
- 行の最後にはセミコロン
- 関数の定義は function 関数名( 引数 );
  - 引数は引数名だけで良い(varは不要)
  - 返り値は return で！(一緒)
- 変数とか文字列をくっつける際は「+」
- ifとかforとかwhileとかは一緒

# 出力して確認

- `console.log( "出力したい内容" );`
  - Processingの`println`と同じ振る舞い。Google ChromeではF12で出力内容を確認することが可能
- `alert( "出力したい内容" );`
  - ダイアログボックスを表示してメッセージを表示
- `document.write( "出力したい内容" );`
  - HTMLのページ中にメッセージを表示

```
test
Hello world
中村聡史
50
```



# もし~だったら

```
if( 条件A ){  
    条件Aのときの動作  
} else if( 条件B ){  
    条件Bのときの動作  
} else {  
    どの条件にもあわなかったときの動作  
}
```



## 条件演算子

演算子	意味
$x > y$	$x$ が $y$ より大きい
$x < y$	$x$ が $y$ より小さい
$x >= y$	$x$ が $y$ 以上
$x <= y$	$x$ が $y$ 以下
$x == y$	$x$ と $y$ が等しい
$x === y$	$x$ と $y$ が等しい(厳密)
$x != y$	$x$ と $y$ が等しくない
$x !== y$	$x$ と $y$ が等しくない(厳密)
$!x$	$x$ は false

演算子	意味
$\&\&$	かつ
$\ \ $	または
$!$	true/false反転

```
console.log( 10 == '10' );
console.log( 10 === '10' );
```



# 配列とオブジェクト

- [] で囲まれた部分が配列の定義となる

```
var a = [ 1, 2, 3, 4, 5 ];
```

```
a[0] = 5;
```

```
console.log( a[1] );
```

- {} で囲まれた部分がオブジェクトの定義となる

```
var human = { name: "宮下", age: 38,
```

```
              position: "准教授" };
```

```
human.position = "教授";
```

```
console.log( human.name + human.position );
```



# オブジェクト

- {} で囲まれた部分がオブジェクトの定義となる
- オブジェクトの各要素はコロン「:」で繋いで定義

要素名: 要素の値

オブジェクト名.要素名

```
var human = { name: "宮下", age: 38, position: "准教授" };
```

```
var human = new Object();  
human.name = "宮下";  
human.age = 38;  
human.position = "准教授";
```

同じ意味



# 関数の定義

- 主に下記の2つのタイプが良く利用される(基本的に両者は同じ意味)

```
function hoge(){ ... } (関数宣言)  
hoge = function() { ... }; (関数式)
```

- どちらの場合も下記のように呼び出すが, 関数式の場合は, 定義以降でしか呼び出せない

```
hoge();
```



# JavaScriptの特徴

- オブジェクトに関数を追加可能

```
var human = {  
  name: "宮下",  
  position: "教授",  
  drink: function(){  
    alert( "エナジードリンク!" );  
  }  
};
```

**human.drink();  
で実行可能**

```
var human = {  
  name: "宮下",  
  position: "教授"  
};  
  
human.drink = function(){  
  alert( "エナジードリンク!" );  
}
```



# ユーザの行動に連動させる

- 要素にイベントと対応する関数を埋め込む！
  - マウスのボタンが押された時, カーソルが上に移動してきた時, テキストボックスにキー入力された時に何か実行させたい！

## [方法]

- [1] HTMLタグ(要素)にイベントと関数を埋め込む

```
<input type="button" value="OK" onclick="hoge()">
```

- [2] 要素を取得してイベントと関数を割り当てる



# 要素に付与可能なイベント一覧

- onload: 読み込みが終わった時
- onresize: ウィンドウの幅が変わった時
- onclick: 要素上でクリックした時
- ondblclick: 要素上でダブルクリックした時
- onmouseover: 要素にマウスカーソルが乗った時
- onmouseout: 要素にマウスカーソルが乗った時
- onmousedown: 要素上でマウスのボタンを押した時
- onmouseup: 要素上でマウスのボタンを離した時
- onmousemove: 要素にマウスカーソルがのった時
- onkeydown: 要素上でキーボードのキーを押した時
- onkeyup: 要素上でキーボードのキーを離した時
- onfocus: 要素にフォーカスが当たった時
- onblur: 要素からフォーカスが外れた時
- onchange: フォームの値が変わった時
- onsubmit: フォームを送信する時



# 要素をどうやって取得する？

- DOMとはDocument Object Model
  - HTMLやXMLの各要素についてアプリケーションから利用するためのAPIのこと
  - HTMLやXMLの任意のタグの情報を取得したり，差し替えたりすることができる
- DOMツリーとは，HTMLやXMLの木構造情報
  - 木構造の情報

```
▼ <html lang="ja">
  ▶ #shadow-root
  ▶ <head>...</head>
  ▼ <body>
    ▶ <header>...</header>
    ▶ <div class="main">...</div>
    ▼ <div class="right">
      ▼ <div class="navi">
        <div class="conts">JavaScript入門</div>
        ▶ <div class="links">...</div>
      </div>
    ▶ <div class="navi">...</div>
    <br>
    ▶ <div class="ads">...</div>
    //...
```



# 要素をどうやって取得する？

- 1つの要素を取得
  - `document.getElementById( "id名" );`
  - `document.querySelector( "セレクトタ名" );`
- 複数の要素を配列として取得
  - `document.getElementsByClassName( "class名" );`
  - `document.getElementsByTagName( "tag名" );`
  - `document.querySelectorAll( "セレクトタ名" );`
- セレクトタ名はCSSの表記方法
  - idは「`#id名`」、クラスは「`.class名`」、tagは「`tag名`」
  - 子の指定は「`>`」。「`#id名 > tag名`」のように指定



# 要素へのイベントと関数の割り当て

**ボタン . クリックされたら → メッセージを表示**

```
ボタン . クリックされたら = function(){  
    メッセージを表示;  
}  
  
button.onclick = function(){  
    alert( "クリックされたよ！" );  
}
```



# DOMを取得してイベント追加

- ボタンのIDを取得して, クリックされたら「やあ」というメッセージを表示する

```
<script>
window.onload = function(){
  var button = document.getElementById("btn");
  button.onclick = function(){
    alert("やあ!");
  }
}
</script>
```

`document.querySelector("#btn");` でもOK

```
<input type="button" id="btn">
```



# DOM自体の操作

- 取得される要素はオブジェクト型
  - 要素 . 属性名 という形で, 様々な情報へアクセス可能 + 情報の差し替え可能

(例) `var node = getElementById(何か);` で取得

- テキスト入力ボックスの場合
  - `console.log( node.value );` // テキストの入力内容を取得
- 一般的なテキストの場合
  - `node.textContent = "Hello!!!";` // テキストを変更
  - `node.innerHTML = "<b>Hello!!!</b>";` // HTMLを変更
  - `node.style.background = "red";` // スタイルシートの背景色を変更
- 画像の場合
  - `node.src = "http://snakamura.org/nakamura.jpg";` // 画像を変更
  - `node.style.width = "400px";` // 画像の横幅を変更



# DOMを取得してイベント追加

- 「ほげほげ」と表示されている. この「ほげほげ」をクリックすると, 「ひげひげ」という文字に変更したい! (onclickはクリックされた時という意味)

```
<script>  
window.onload = function(){  
  var node = document.getElementById("hoge");  
  node.onclick = function(){  
    node.innerHTML = "ひげひげ";  
  }  
}  
</script>
```

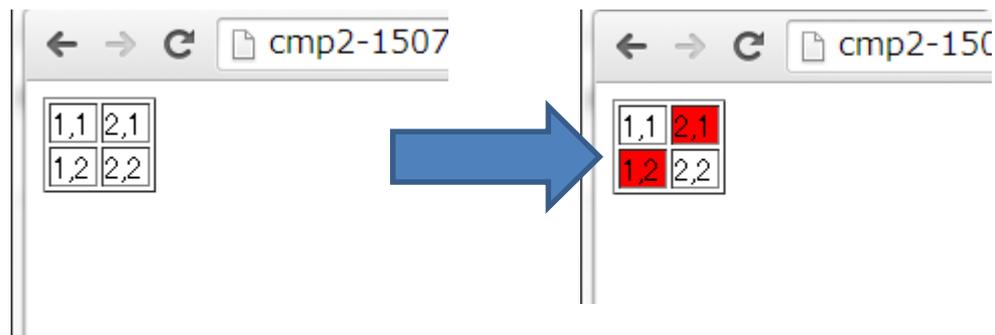
document.querySelector("#hoge"); でもOK

```
<div id="hoge">ほげほげ</div>
```

資料: 3-2

# 表でクリックされた場所を赤色に

- 2x2の表を作成し, その表の中でクリックされた場所を赤色にするプログラムの作成

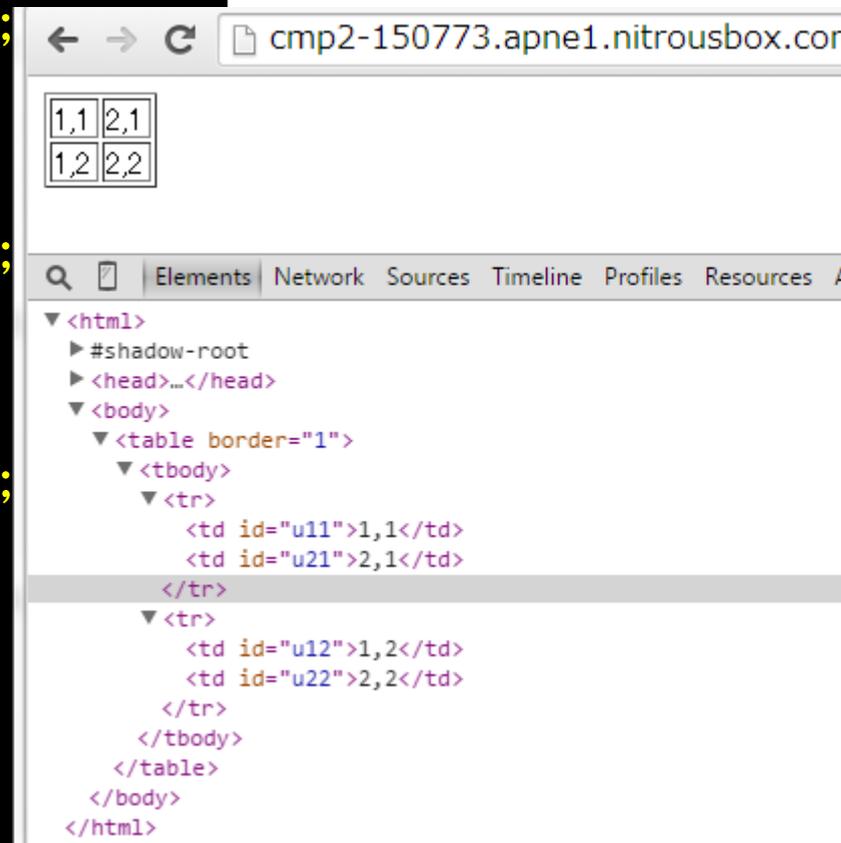


- ヒント
  - table, tr, tdタグを利用して2x2の表を作成
  - そのそれぞれにIDを付与
  - それぞれのクリックイベントに関数を割り当て
  - 割り当てた関数で背景色を変更



document.querySelector("#u11"); でもOK

```
<script>  
window.onload = function(){  
  var td11 = document.getElementById("u11");  
  td11.onclick = function(){  
    this.style.background = "red";  
  }  
  var td12 = document.getElementById("u12");  
  td12.onclick = function(){  
    this.style.background = "red";  
  }  
  var td21 = document.getElementById("u21");  
  td21.onclick = function(){  
    this.style.background = "red";  
  }  
  var td22 = document.getElementById("u22");  
  td22.onclick = function(){  
    this.style.background = "red";  
  }  
}  
</script>
```



```
<table border=1>  
  <tr><td id="u11">1,1</td><td id="u21">2,1</td></tr>  
  <tr><td id="u12">1,2</td><td id="u22">2,2</td></tr>  
</table>
```



```
<script>
  window.onload = function(){
    var tdunit = new Array( 2 );
    for( var x=0; x<2; x++ ){
      tdunit[x] = new Array( 2 );
    }
    for( var x=0; x<2; x++ ){
      for( var y=0; y<2; y++ ){
        tdunit[x][y] = document.getElementById("u" + (x+1) + (y+1));
        tdunit[x][y].onclick = function(){
          this.style.background = "red";
        }
      }
    }
  }
</script>
```

```
<table border=1>
  <tr><td id="u11">1,1</td><td id="u21">2,1</td></tr>
  <tr><td id="u12">1,2</td><td id="u22">2,2</td></tr>
</table>
```



# 複数取得する場合は？

- 下記のメソッドの場合結果を複数取得
  - getElement<sup>s</sup>ByClassName, getElement<sup>s</sup>ByTagName, document.querySelector<sup>All</sup>
  - 結果は配列として取得できるため、下記のように処理することが可能

```
<script>  
  window.onload = function(){  
    var elems = document.getElementsByTagName("td");  
    for( var i=0; i<elems.length; i++ ){  
      elems[i].style.background = "red";  
    }  
  }  
</script>
```

document.querySelectorAll("td"); でもOK



# 配列を作らないでも...

```
<script>
  window.onload = function(){
    var units = document.getElementsByTagName("td");
    for( var i=0; i<units.length; i++ ){
      units[i].onclick = function(){
        this.style.background = "red";
      }
    }
  }
</script>
```

```
<table border=1>
  <tr><td id="u1 1">1,1</td><td id="u21">2,1</td></tr>
  <tr><td id="u1 2">1,2</td><td id="u22">2,2</td></tr>
</table>
```



# DOMの情報を取得する

- 他の実装方法
  - 引数としてthis(自分自身)を渡す
  - ここでthisはそのタグ(ノード)自体になるので  
node.innerHTML = "ひげひげ"; とできる

```
<script>  
function change( node ){  
    node.innerHTML = "ひげひげ";  
}  
</script>  
  
<div id="hoge" onclick="change( this )">ほげほげ</div>
```



# window.onload って何？

- ウェブページがロードされたら～という意味
  - この中にウェブページがロードされてからのプログラムを追加しよう！（ロードされる前に実行してしまうとおかしくなることがあるので要注意！）

```
<script>  
  window.onload = function(){  
    // ここにプログラムを書く！  
  }  
</script>
```



# 定期的に行

- `setInterval( function(){ 実行する内容 }, ミリ秒);`
  - 指定ミリ秒後に指定の操作を実行する

表でクリックされた場所を赤色に下記プログラムを追加！

```
setInterval(function(){  
    if( td11.textContent === "○" ){  
        td11.textContent = "";  
    } else {  
        td11.textContent = "○";  
    }  
},1000);
```



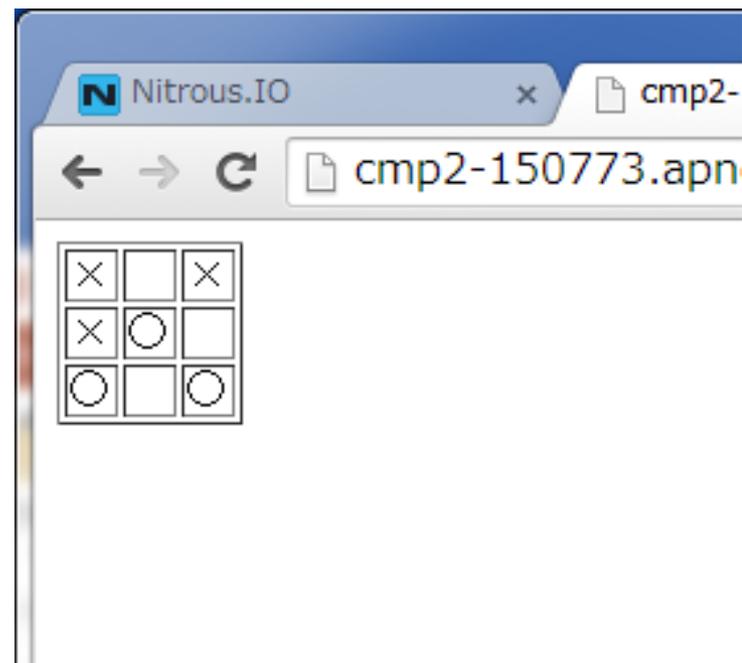
# 演習

- 資料 3-4 を利用して, 名前をクリックするとなにか適当な呼び名に切り替わるようにせよ

- 荒川薫
- 阿原一志
- 五十嵐悠紀
- 菊池浩明
- 小林稔
- 小松孝徳
- 斉藤裕樹
- 嵯峨山茂樹
- 鈴木正明
- 中村聡史
- 橋本直
- 福地健太郎
- 宮下芳明
- 渡邊恵太
- 鹿喰善明

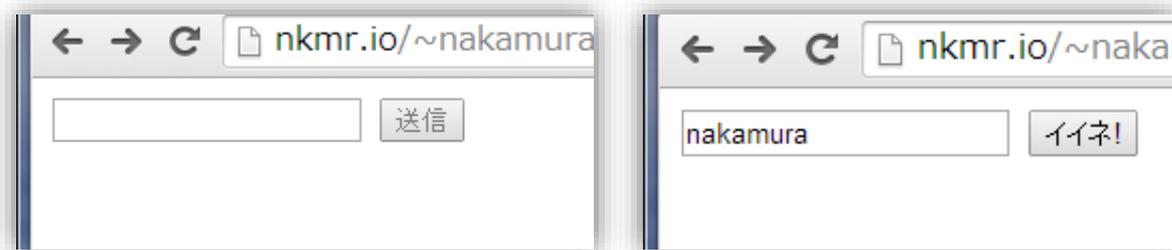
# 演習

- 2x2のテーブルのプログラムの拡張して、3x3の表の色が変わるページを作成してみよう！
- 3x3のテーブルのそれぞれの中にある「○」「×」「 」をクリックする度に
  - 「○」から「×」
  - 「×」から「 」
  - 「 」から「○」へと変化するようにせよ



# 演習

- テキスト入力ボックスと送信ボタンを用意し，入力ボックスで入力されている内容に応じて送信ボックスの状態を切り替えてみよう
- ここでは，入力ボックスに自分の名前が入力されていたら，送信ボックスのボタンを「イイネ!」と変更して，送信可能にせよ





# 手順(ヒント)

- フォームタグやテキストボックス, 送信ボックスを用意し, それぞれにIDを付与する
- 送信ボックスを最初利用不可にするため disabled に設定しておく
- テキストボックスで何か文字が入力されていたら入力内容をチェックし, 何らかの文字が入力されていたら送信ボックスを押せるようにせよ (XXX.disabled = false)
- 入力内容が自分の名前だったらテキストを変更する
- テキストボックスへの入力は onkeydown で取得できるのでそのイベント(関数)を追加しよう!



# JavaScriptは別ファイルへ

- HTMLとJavaScriptが混在しているとぐちゃぐちゃになってしまう（PHPとJavaScriptが混在すると恐ろしいことに）
- 可能な限りJavaScriptファイルは別ファイルへ
- main.js などに保存し script タグで呼び出し
  - src で呼び出すファイルのパスを指定する

```
<script src="main.js"></script>
```



# JavaScriptの良いところ

- 開発者がかなり多いため、ライブラリも豊富
  - グラフを描画する (GeoChart)
  - 地図を表示する
  - Processingのプログラムを実行する
  - 色々便利にするためのフレームワークもある！
    - JavaScriptを一から全部書いていくのはかなり面倒

```
<script src="呼び出すJSのURL"></script>
```

<https://developers.google.com/chart/interactive/docs/gallery?hl=ja>

# Google Charts



Google Charts

🔍

nakamura.satoshi@gmail.com  
ログアウト


サービス > Google Charts
Google Developers アンケートにご協力ください

## Google Charts 👍 9

- ▶ Overview
- ▼ Chart Gallery
  - Miscellaneous Examples
  - Annotation Charts
  - Area Charts
  - Bar Charts
  - Bubble Charts
  - Calendar Charts
  - Candlestick Charts
  - Column Charts
  - Combo Charts
  - Diff Charts
  - Gauge Charts
  - Geo Charts
  - Histograms
  - Intervals
  - Line Charts
  - Maps
  - Org Charts
  - Pie Charts
  - Sankey Diagrams
  - Scatter Charts
  - Stepped Area Charts
  - Table Charts

## Chart Gallery

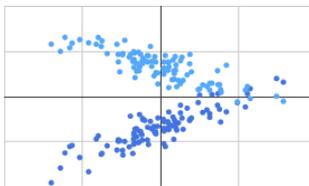
Our gallery provides a variety of charts designed to address your data visualization needs. These charts are based on pure HTML5/SVG technology (adopting VML for old IE versions), so no plugins are required. All of them are interactive, and many are pannable and zoomable. Adding these charts to your page can be done in [a few simple steps](#).

Some additional community-contributed charts can be found on the [Additional Charts page](#).

### Geo Chart



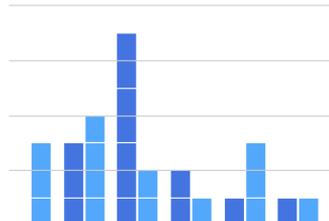
### Scatter Chart



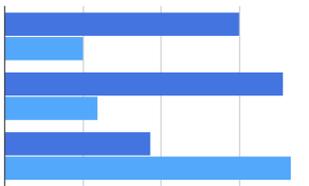
### Column Chart



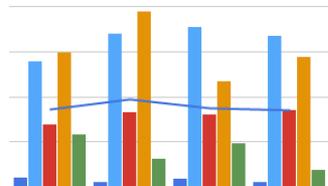
### Histogram



### Bar Chart



### Combo Chart



# 演習

- 総合数理学部 学生数のパイチャートを作成
  - 現象数理 80人
  - FMS 100人
  - ND 80人
- 日本の各都道府県の人口毎の地図を作成
  - Tokyo 13286735人
  - Kyoto 2620210人
  - Fukuoka 5090712人
- Speed メーターを作成せよ

```
<html>
<head>
  <script type="text/javascript" src="https://www.google.com/jsapi"></script>
  <script type="text/javascript">
    google.load('visualization', '1.0', {'packages':['corechart']});
    google.setOnLoadCallback(drawChart);

    function drawChart() {
      var data = new google.visualization.DataTable();
      data.addColumn('string', 'Name');
      data.addColumn('number', '人数');
      data.addRows([[ '現象数理', 80], ['FMS', 100], ['ND', 80]]);

      var options = {'title':'学生数比較', 'width':400, 'height':300};
      var chart = new google.visualization.PieChart(document.getElementById('chart_div'));
      chart.draw(data, options);
    }
  </script>
</head>

<body>
  <div id="chart_div"></div>
</body>
</html>
```

```
<html> <head>
  <script src='https://www.google.com/jsapi'></script>
  <script>
    google.load('visualization', '1', {'packages': ['geochart']});
    google.setOnLoadCallback(drawMarkersMap);
    function drawMarkersMap() {
      var data = google.visualization.arrayToDataTable([
        ['県', '人口'], ['Tokyo', 13286735], ['Kyoto', 2620210], ['Fukuoka', 5090712]
      ]);
      var options = {
        region: 'JP',
        displayMode: 'markers',
        colorAxis: {colors: ['green', 'blue']}
      };
      var chart = new google.visualization.GeoChart(document.getElementById('chart_div'));
      chart.draw(data, options);
    };
  </script>
</head>
<body>
  <div id="chart_div" style="width: 900px; height: 500px;"></div>
</body>
</html>
```

```
<html> <head>
  <script src="https://www.google.com/jsapi"></script>
  <script>
    google.load("visualization", "1", {packages:["gauge"]});
    google.setOnLoadCallback(drawChart);
    function drawChart() {
      var data = google.visualization.arrayToDataTable([
        ['Label', 'Value'], ['Speed', 80],
      ]);
      var options = {
        width: 400, height: 120, redFrom: 90, redTo: 100,
        yellowFrom:75, yellowTo: 90, minorTicks: 5
      };
      var chart = new google.visualization.Gauge(document.getElementById('chart_div'));
      chart.draw(data, options);
      setInterval(function() {
        data.setValue(0, 1, 40 + Math.round(60 * Math.random()));
        chart.draw(data, options);
      }, 1000);
    }
  </script>
</head>
<body>  <div id="chart_div" style="width: 400px; height: 120px;"></div> </body>
</html>
```

# Processing.js / p5.js

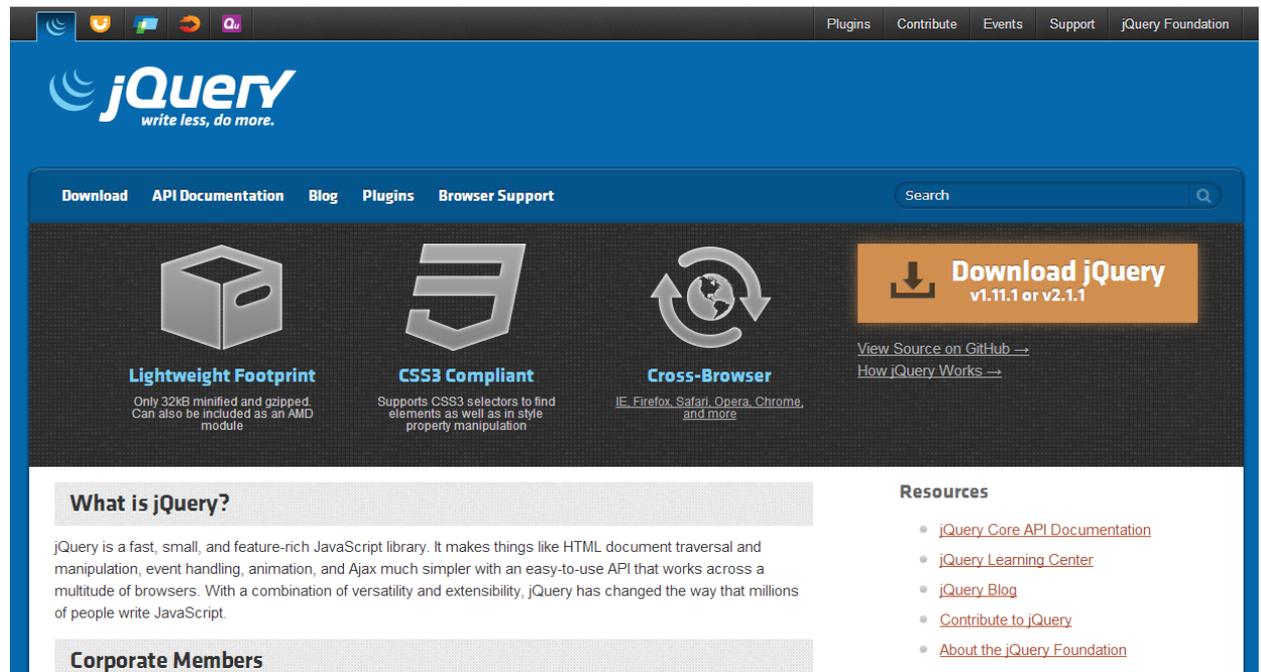
- Processing.js や p5.js はProcessingのプログラムを実行するためのライブラリ



The screenshot shows the website [processingjs.org/download/](https://processingjs.org/download/). The page title is "Processing.js" and it is described as "a part of the Processing Visualization Language". A red callout box with white text says "ダウンロードしてアップロードしよう" (Download and upload). Below the callout, the "Download" section lists two options: "Development - processing.js v1.4.8" and "Production - processing.min.js v1.4.8", with the latter highlighted by a red box. Below this, it says "Alternatively, install through Bower by using `bower install Processing.js`". The "Previous releases" section is also visible at the bottom.

# アプリケーションフレームワーク

- Prototype.js
  - jQuery
  - Google Web Toolkit
- などなど



jQuery  
write less, do more.

Download API Documentation Blog Plugins Browser Support Search

**Lightweight Footprint**  
Only 32KB minified and gzipped.  
Can also be included as an AMD module

**CSS3 Compliant**  
Supports CSS3 selectors to find elements as well as in style property manipulation

**Cross-Browser**  
IE, Firefox, Safari, Opera, Chrome, and more

**Download jQuery**  
v1.11.1 or v2.1.1

View Source on GitHub →  
How jQuery Works →

**What is jQuery?**  
jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.

**Resources**

- [jQuery Core API Documentation](#)
- [jQuery Learning Center](#)
- [jQuery Blog](#)
- [Contribute to jQuery](#)
- [About the jQuery Foundation](#)

**Corporate Members**



# jQueryを導入しよう

- <http://jquery.com/> にアクセスして最新版のファイルをダウンロード
- `www` に直接置いても良いが、できれば機能で整理した方が良かったため、新たに `lib` というフォルダ（他のフォルダ名でも良い）を作成し、そこに入れたほうが良い
  - `lib` はライブラリの意味
  - `<script src="lib/jquery-2.1.1.min.js"></script>`



# jQuery + main.js + HTML

```
<html>
<head>
<script src="lib/jquery-2.1.1.min.js"></script>
<script src="main.js"></script>
</head>
<body>

<div id="hoge" class="text">ほげほげ</div>
<form action="test.html" method="get" id="input_form">
<input type="text" name="msg" id="msgbox">
<input type="submit" name="send" id="sendbutton" disabled>
</form>

</body>
</html>
```



# jQueryの\$()って何？

- \$(function(){ ... }); や \$('#hoge').on( ... ); って？
  - 「\$()」は「jQuery()」の略
  - 本来は
    - jQuery( function(){ ... } );
    - jQuery( '#hoge' ).on( ... );
    - jQuery( '#hoge' ).css( ... );

```
$(function(){  
  $('#hoge').on('click', function(){  
    $('#hoge').css('background', '#f00');  
  });  
});
```

クリックされた時に色を変更する



# main.js というファイルを作成

- main.js というファイルの中に下記のプログラムを書いてみよう
  - `$(function(){ プログラム });` で jQuery に関するプログラムを書く(そういうものだと思って下さい)
  - `$('#クラス名')`
    - `.on( 'イベント名', function(){ イベント時の処理 });`

```
$(function(){  
  $('#hoge').on('click', function(){  
    $('#hoge').css('background', '#f00');  
  });  
});
```

クリックされた時に色を変更する

# main.js というファイルを作成

- 以下だけだとちゃんと動作しない

```
$('#hoge').on('click', function(){  
    $('#hoge').css('background', '#f00');  
});
```

- コードが読み込まれた時点では, id=hogeのdivタグを読み込んでいないため
- `$(function(){ プログラム });` は, とりあえずコードを全部読み込んでから~という意味になる



# Element(Node)の指定方法

- elementを選択するには
  - id : `$("#idname")`
  - class : `$(".classname")`
  - tag : `$("tagname")`
  - name : `$("[name=target]")`



# イベント一覧

- mouseover: 要素にマウスが乗った時
- mouseout: 要素からマウスが離れた時
- mousedown: 要素上でクリックボタンが押された時
- mouseup: 要素上でクリックボタンが離れた時
- mousemove: 要素上でマウスが動かされた時
- click: 要素がクリックされた時
- dblclick: 要素がダブルクリックされた時
- keydown: 要素にフォーカスした状態で, キーボードのキーが押された時
- keyup: 要素にフォーカスした状態で, キーボードのキーが離された時
- focus: 要素にフォーカスが当たった時
- blur: 要素からフォーカスが外れた時
- change: 入力内容が変更された時
- resize: 要素がリサイズされた時
- scroll: 要素がスクロールされた時



# Element(Node)の操作

- `$( 指定 ). css( 'プロパティ名', '変更後の値' );`
  - スタイルシートを変更する
- `$( 指定 ). attr( '属性名', '変更後の値' );`
  - 属性値を変更する
- `$( 指定 ). on( 'イベント名', その時の動作 );`
  - その時の動作は `function(){ ... }` で書かれる
- `$( 指定 ). animate( アニメーションの指定 );`
  - アニメーションの指定では最初に `{}` でCSSの内容を指定し, カンマでミリ秒を指定する



# main.js を編集

```
$(function(){
  $('#hoge').on('click', function(){
    $('#hoge').css('background', '#f00');
  });

  $('#msgbox').on('keyup', function(){
    if( $('#msgbox').val() == "" ){
      $("#sendbutton").attr("disabled", "disabled");
    } else {
      $('#sendbutton').removeAttr("disabled");
      if( $('#msgbox').val() == 'nakamura' ){
        $('#sendbutton').attr( 'value', "いいね" );
      }
    }
  });
});
```



# jQuery + Google Maps

- gmaps.js をダウンロードしてlibにでも放り込む
  - <http://hpneo.github.io/gmaps/>

```
<script src="lib/jquery-2.1.1.min.js"></script>  
<script src="http://maps.google.com/maps/api/js?sensor=false"></script>  
<script src="lib/gmaps.js"></script>  
<script src="main.js"></script>
```

- 地図用のHTML要素を用意(サイズは適当)

```
<div id="map" style="width:400px;height:300px"></div>
```



# jQuery + Google Maps

- main.js に下記のコードを書いてみましょう！

```
$(document).ready(function(){  
  map = new GMaps({  
    div: '#map',  
    zoom: 15,  
    lat: 35.7066,  
    lng: 139.6633,  
  });  
});
```

# 地図上にパスも描けるよ！

```
$(document).ready(function(){  
  map = new GMaps({  
    div: '#map',  
    zoom: 15,  
    lat: 35.7066,  
    lng: 139.6633,  
  });  
  
  var path = [  
    [35.706821, 139.659765],  
    [35.706839, 139.663305],  
    [35.705854, 139.665719]  
  ]  
  map.drawPolyline({  
    path: path,  
    strokeColor: '#f00',  
    strokeOpacity: 0.6,  
    strokeWeight: 6,  
  });  
});
```





# 他にも色々

- マーカーを立てる
  - 適当な画像を表示する
  - ポリゴンで囲う
  - などなど
- 
- 詳しくは下記サイトへ
    - <http://hpneo.github.io/gmaps/>
    - <http://taklog.info/2013/07/26/googlemaps/>



# 課題

- nitrous.io 上または自身のサーバ上で下記の課題に取り組むこと(課題チェックはChrome)
  - 課題3-1: JavaScriptを用いて, リロードする度に結果が変わるページを作成してみましょう!
  - 課題3-2: jQueryを使ってアニメーションが盛り沢山のページを作成してみましょう!
  - 課題3-3: Google Chartsを使って何らかの身近な値のグラフ表現をせよ
  - 課題3-4: Google Mapsを使って道案内を作成せよ
  - 課題3-5: Processing.js または p5.js を利用して, これまでに作成したProcessingのプログラムを動かせ!



# 課題の提出方法

- すべて nitrous.io 上またはどこかのサーバで動作するようにすること
  - 第X回課題として、ウェブページを作成し、そこから説明を付与して課題へのリンクを作成すること
  - 課題のトップページへのリンクは、講義資料サイトからURLとして登録すること

## コンテンツプログラミング実習 第5回課題成果

### ■ 課題5-1 (a)

JavaScript (jQuery)+現在位置+地図+何らかのWeb APIを使ったアプリケーションを作る

- 地図とレストランサイトと天気予報を使った超絶便利なサイト！

### ■ 課題5-2

FMSIにまつわる何かのデータを格納するXMLファイルを作成せよ。また、そのXMLファイルから状況に応じてデータを読み込み結果を出力せよ

- 作成したXML(シラバスのXML)
- 作成したウェブページ(受講に関するお手軽サイト)