

# ネットワークアプリケーション

## 第11回 JavaScriptによる クライアントサイドウェブプログラミング

石井 健太郎

(423研究室・オフィスアワー火3限)

# スケジュール

- 9月27日 第1回「TCP/IPプロトコルスイート」
- 10月4日 第2回「Javaによるウィンドウプログラミング」
- 10月11日 第3回「ネットワークアプリケーションのプログラミングモデル」
- 10月18日 第4回「Javaによるネットワークプログラミング」
- 10月25日 第5回「Javaによるネットワークプログラミング」
- 11月8日 第6回「Javaによるネットワークプログラミング」
- 11月15日 第7回「Javaによるネットワークプログラミング」 **最終課題(1)**
- 11月17日** 第8回「ウェブプログラミングについて」

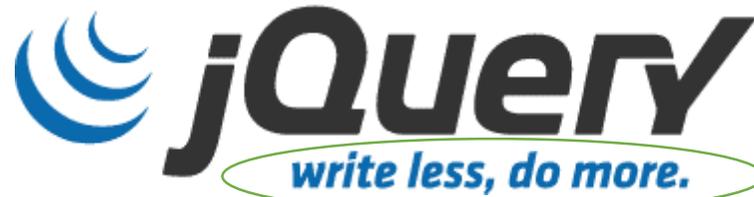
# スケジュール

- 11月22日 第9回「JavaScriptによるクライアントサイドウェブプログラミング」
- 11月29日 第10回「JavaScriptによるクライアントサイドウェブプログラミング」
- 12月6日 第11回「JavaScriptによるクライアントサイドウェブプログラミング」
- 12月13日 第12回「JavaScriptによるクライアントサイドウェブプログラミング」
- 12月20日 第13回「JavaScriptによるクライアントサイド...」 **最終課題(2)**
- 1月10日 第14回「まとめと演習」 **課題の演習**
- 1月17日 第15回「まとめと演習」 **課題の演習**

- **演習課題を提出してください**

# jQueryとは？

- JavaScriptを簡単に使うための仕組みが詰まったライブラリ



write less, do more.

# プログラムの比較例

- htmlのすべての<div>要素の文字列を赤くする

- 通常のJavaScript

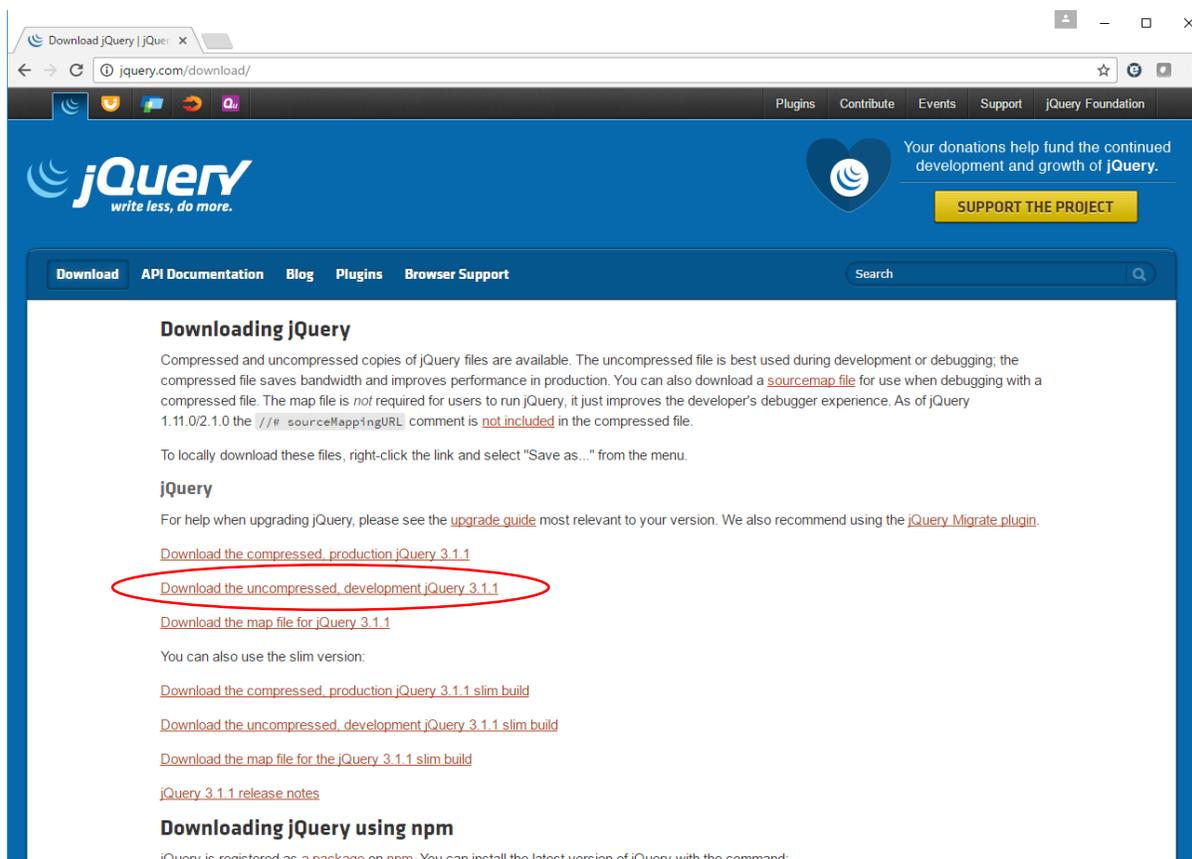
```
var divs = document.getElementsByTagName('div');  
for (var i=0; i<divs.length; i++) {  
    divs[i].style.color='red';  
}
```

- jQueryを使うと

```
$('#div').css('color', 'red');
```

# jQueryの使いかた

- jQueryをダウンロードする
  - <http://jquery.com/download/>



# jQueryの使いかた

- `<script>`タグでダウンロードしたファイルを読み込む

```
<html>
<head>
<meta charset="utf-8">
<title>タイトルはここに</title>
</head>
<body>
...本文...
<script type="text/javascript" src="jquery-3.1.1.js"></script>
<script>
  ここに JavaScript (jQuery含む) を書く
</script>
</body>
</html>
```

jQueryを読み込み

jQueryのプログラムを書く

# jQueryの使いかた

- jQueryのプログラムは「`$ (function () {…} )`」の「…」部分に記述する

```
<script type="text/javascript" src="jquery-3.1.1.js"></script>
<script>
  ここに(あれば)通常の JavaScript のプログラムを書く
  $(function() {
    ここに jQuery のプログラムを書く
  })
</script>
```

# jQueryの使いかた

- 以下の2ステップ
  - STEP1: 要素を選択して
  - STEP2: 操作を行う

```
$('#li').css('color', 'red');
```

\$('#セレクタ')

メソッド()

# セレクタ

- **基本的なセレクタ(名称で指定)**
  - `$('h2')` // タグ名セレクタ
    - 特定のタグ(この場合は<h2>タグ)の要素
  - `$('#m1')` // IDセレクタ
    - 特定のid属性(この場合はm1)のついた要素
  - `$('.rice')` // クラスセレクタ
    - 特定のclass属性(この場合はrice)のついた要素

# セレクタ

- **階層構造をたどって要素を特定するセレクタ**
  - `$ ('#today li')`
    - idがtodayの要素の下の階層にあるli要素
  - `$ ('#today > li')`
    - idがtodayの要素の直下の階層にあるli要素
  - `$ ('#m1 + li')`
    - idがm1の要素と同じ階層ですぐ次のli要素
      - #m1とliの間に別の要素がある場合は選択されない
  - `$ ('#m1 ~ li')`
    - idがm1の要素と同じ階層で以降に出現するli要素
      - #m1とliの間に別の要素がある場合も(複数)選択される

# セレクタ

- **対象を絞り込むフィルタ系セレクタ**

- `$('ul:first')`, `$('ul:last')`
  - 最初の要素, 最後の要素
- `$('ul:even')`, `$('ul:odd')`
  - 偶数番目の要素, 奇数番目の要素
- `$('li:eq(3)')`
  - 3番目(0オリジン)の要素
- `$('li:contains('ラーメン'))')`
  - 特定の文字列を含む要素

- **その他にも数多くのセレクタが用意されている**

- <http://api.jquery.com/category/selectors/> (本家)
- <http://semoooh.jp/jquery/api/selectors/> (日本語)

# jQueryの使いかた

- 以下の2ステップ
  - STEP1: 要素を選択して
  - STEP2: 操作を行う

```
$('#li').css('color', 'red');
```

\$('#セレクタ')

メソッド()

# メソッド

- メソッドは1つのセレクタに対して複数書ける(メソッドチェーン)

3番目の<li>要素を

文字色をredにして,

背景色をlimeにして,

- `$('li:eq(2)').css('color', 'red')`  
`.css('backgroundColor', 'lime')`  
`.fadeOut();`

フェードアウトする

# メソッド

- **多様な種類が存在**
  - スタイル操作: CSS
  - 属性操作: Attributes
  - 要素間の移動・検索: Traversing
  - 要素の追加・削除: Manipulation
  - イベント: Events
  - エフェクト: Effects
  - Ajax(非同期通信): Ajax
- **全ての使い方を覚える必要はなく、リファレンスを見ながら使えればOK**
  - <http://api.jquery.com> (本家)
  - <http://semoooh.jp/jquery/> (日本語)

# スタイル操作: CSS

- **CSSによる操作**
  - **スタイルの設定**
    - `$('#today').css('color', 'red')`
  - **スタイルの取得**
    - `var color = $('#today').css('color')`
- **CSSで操作できるスタイルを指定可能**
  - 色
  - フォント・サイズ
  - テキストスタイル(行間など)
  - 幅・高さ
  - 境界線
  - 表示・配置

# 練習: セレクタとスタイル操作

- サンプルファイルをダウンロードしてください

<http://lss.oit.ac.jp/~t2015039/NetworkApplication-JavaScript/jquery.html>

- 下記の操作を考えて試してください
  - <h2>タグの文字を赤 (red) に変える
  - idがm1の要素の文字を赤 (red) に変える
  - classがriceの要素の文字を青 (blue) に変える
  - idがtodayの要素の下の階層にあるliの文字を赤 (red) に変える
  - idがm1の要素と同じ階層で以降に出現するliの背景を黄緑 (lime) に変える
  - 最後のliの文字を青 (blue) に変える
  - 2番目のliの背景を黄緑 (lime) に変える
  - 「ラーメン」を含むliの背景をオレンジ (orange) に変える

# 属性操作: Attributes

- **<a>要素にtarget属性を追加**
  - `$('a').attr('target', '_blank');`

```
<a href="http://www.example.com" target="_blank"></a>
```

取得

```
.attr('href')  
.attr('src')  
.width()  
.html()  
.text()
```

設定

```
.attr('href', 'index.html')  
.attr('src', 'apple.jpg')  
.width('80px')  
.html('<p>今日は学校です. </p>')  
.text('こんにちは')
```

# 要素間の移動・検索: Traversing

```
<html>  
  <body>  
    <h1></h1>  
    <ul>  
      <li></li>  
      <li></li>  
      <li></li>  
    </ul>  
  </body>  
</html>
```

# 要素間の移動・検索: Traversing

- \$('h1')

```
<html>  
  <body>  
    <h1></h1>  
    <ul>  
      <li></li>  
      <li></li>  
      <li></li>  
    </ul>  
  </body>  
</html>
```

# 要素間の移動・検索: Traversing

- `$('.h1').next()`

```
<html>
  <body>
    <h1></h1>
    <ul>
      <li></li>
      <li></li>
      <li></li>
    </ul>
  </body>
</html>
```

# 要素間の移動・検索: Traversing

- `$('.h1').next().find('li')`

```
<html>
  <body>
    <h1></h1>
    <ul>
      <li></li>
      <li></li>
      <li></li>
    </ul>
  </body>
</html>
```

# 要素間の移動・検索: Traversing

- `$('.h1').next().find('li').eq(1)`

```
<html>
  <body>
    <h1></h1>
    <ul>
      <li></li>
      <li></li>
      <li></li>
    </ul>
  </body>
</html>
```

# 要素間の移動・検索: Traversing

- `$('#h1').next().find('li').eq(1).text('ここでは！')`

```
<html>
  <body>
    <h1></h1>
    <ul>
      <li></li>
      <li>ここでは！</li>
      <li></li>
    </ul>
  </body>
</html>
```

# 要素の追加・削除: Manipulation

- `$('#h1').append('大阪府').after('<h2>枚方市</h2>')`

```
<html>
  <body>
    <h1>大阪府</h1>
    <h2>枚方市</h2>
    <ul>
      <li></li>
      <li>ここです! </li>
      <li></li>
    </ul>
  </body>
</html>
```

# 要素の追加・削除: Manipulation

- **.prepend ('追加文')**
  - **内部挿入: 要素内部の先頭にコンテンツを挿入**
  - **\$ ('div').prepend ('追加文')**
    - `<div>あああ</div> ⇒ <div>追加文あああ/div>`
- **.before ('追加文')**
  - **外部挿入: 要素の前にコンテンツを挿入**
  - **\$ ('div').before ('追加文')**
    - `<div>あああ</div> ⇒ 追加文<div>あああ/div>`

# エフェクト: Effects

- `$('.h1').fadeOut(5000)`

```
<html>
  <body>
    <h1>東京都</h1>
    <h2>多摩市</h2>
    <ul>
      <li></li>
      <li>ここです！</li>
      <li></li>
    </ul>
  </body>
</html>
```

ゆっくり消える

# エフェクト: Effects

- `.fadeIn ()`
  - フェードイン
- `.show ()`
  - 表示
- `.hide ()`
  - 非表示

# jQueryの使いかた

- 以下の2ステップ
  - STEP1: 要素を選択して
  - STEP2: 操作を行う

```
$('#li').css('color', 'red');
```

\$('#セレクタ')

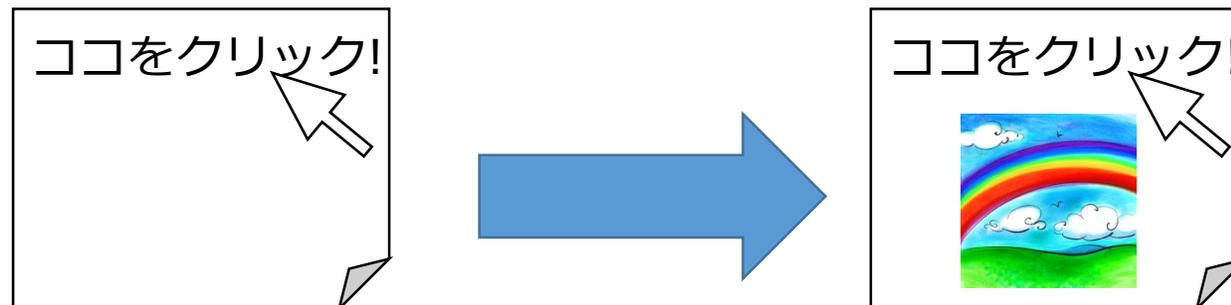
メソッド()

# メソッド

- **多様な種類が存在**
  - スタイル操作: CSS
  - 属性操作: Attributes
  - 要素間の移動・検索: Traversing
  - 要素の追加・削除: Manipulation
  - イベント: Events
  - エフェクト: Effects
  - Ajax(非同期通信): Ajax
- **全ての使い方を覚える必要はなく、リファレンスを見ながら使えばOK**
  - <http://api.jquery.com> (本家)
  - <http://semoooh.jp/jquery/> (日本語)

# イベントを利用するプログラム

- ある文字列がクリックされた時に,
  - 画像を表示する
- ある画像の上にマウスカーソルをのせた時に,
  - 画像の説明を表示する
- ある文字列がダブルクリックされた時に,
  - 文字列を赤色にする

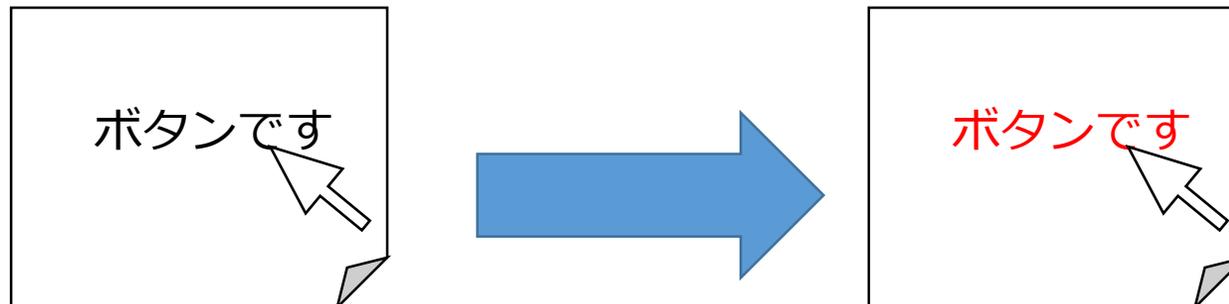


# イベント: Events

```
<div id="button">ボタンです</div>
```

```
$('#button').click(function() {  
    // ここにクリックされた時の動作を記述  
    $(this).css('color', 'red');  
});
```

イベントが設定された  
要素自身を選択する場合  
\$(this)を用いる



# イベント: Events

- ```
$( 'div' ).hover (
    function () { $('#image').fadeOut () /* カーソルが乗った時 */ },
    function () { $('#image').fadeIn () /* カーソルが外れた時 */ }
);
```

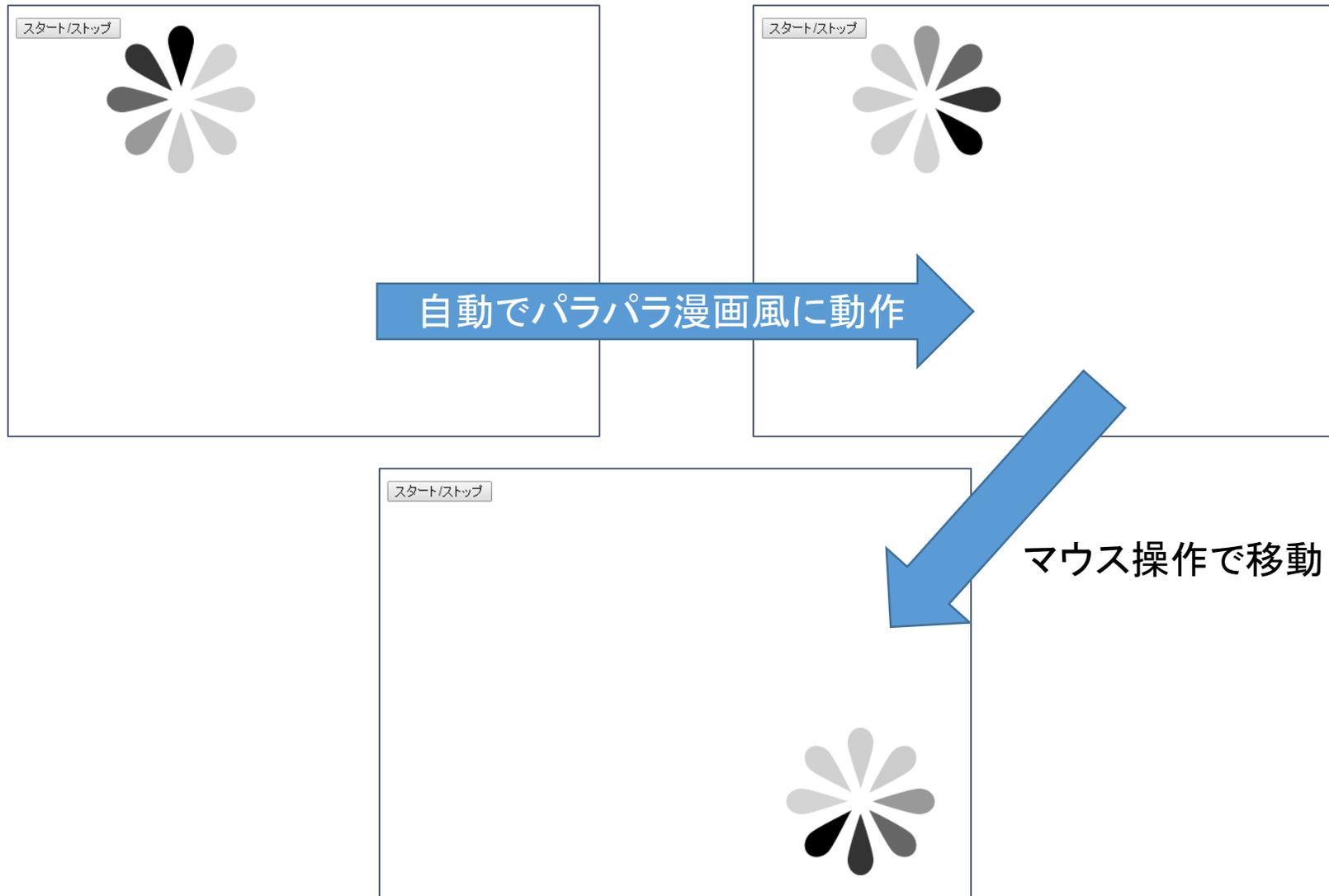
- ```
$( document ).ready (function () {
    //要素の準備が出来たら実行
});
```

- ```
$(function () {
    //jQueryを使うときにいつも書くこれは上記の省略記法
});
```

ブラウザの処理の流れ

1. HTMLファイルの読み込み完了
2. `$(document).ready()`の実行
3. 画像読み込み・ブラウザ表示

# アニメーション



# 一定時間後に繰り返す

- `timerID = setTimeout ()`
  - 指定した時間の経過後に関数を呼び出す
  - 関数の最後で自分自身を呼び出すように使うと繰り返し動作する関数を実現できる

```
function tokei() {  
  $('#target').append("ほ");  
  setTimeout(tokei, 200);  
}
```

- `clearTimeout (timerID)`
  - タイマーを解除する
    - 繰り返し動作する関数もストップする

# 画像のプリロード

- 画像の読み込みには時間がかかる場合があるので、タイマー関数で画像を表示する前にブラウザに読み込んでおくとよい

```
$("#<img>").attr("src", "image0.png");  
$("#<img>").attr("src", "image1.png");  
$("#<img>").attr("src", "image2.png");  
$("#<img>").attr("src", "image3.png");  
$("#<img>").attr("src", "image4.png");
```

```
for (var i = 0; i < 5; i++) {  
    $("#<img>").attr("src", "image" + i + ".png");  
}
```

# 練習: ぱらぱらマンガ

- 配布フォルダの画像を用いて,  
ぱらぱらマンガでアニメーションを作ってみよう
  - ヒント: `setTimeout()` を関数の最後に使う



- アニメーションのスピードを変えるとどうなるか試してみましょう
- 画像のプリロードにもトライしてみましょう

# animate () メソッド

- CSSのさまざまなプロパティ(色・サイズ・場所など)を徐々に変化させるメソッド

```
$('#image').animate(CSSプロパティ,  
                    変化時間,  
                    変化スピード,  
                    終了後の処理);
```

# animate () メソッド

- CSSのさまざまなプロパティ(色・サイズ・場所など)を徐々に変化させるメソッド

```
$('#image').animate(CSSプロパティ,  
    変化時間,  
    変化スピード,  
    終了後の処理);
```

連想配列で複数指定が可能:

```
{  
    プロパティ1: "変化後の値"  
    ...  
    プロパティn: "変化後の値"  
}
```

```
#image {  
    position: absolute;  
}
```

- **注意!**

- animate () メソッドを用いて「移動」を行う場合には、動かす対象のCSSの**position**プロパティをデフォルトから**relative** (相対位置で指定) もしくは**absolute** (絶対位置で指定) に変更する必要がある

# animate () メソッド

- CSSのさまざまなプロパティ(色・サイズ・場所など)を徐々に変化させるメソッド

```
$('#image').animate(CSSプロパティ,  
    変化時間,  
    変化スピード,  
    終了後の処理);
```

アニメーション全体にかかる時間:  
"slow", "normal", "fast" か  
ミリ秒単位の時間

アニメーションの変化スピード:  
"linear": 一定のスピードで変化  
"swing": 最初速くのちにゆっくり  
など

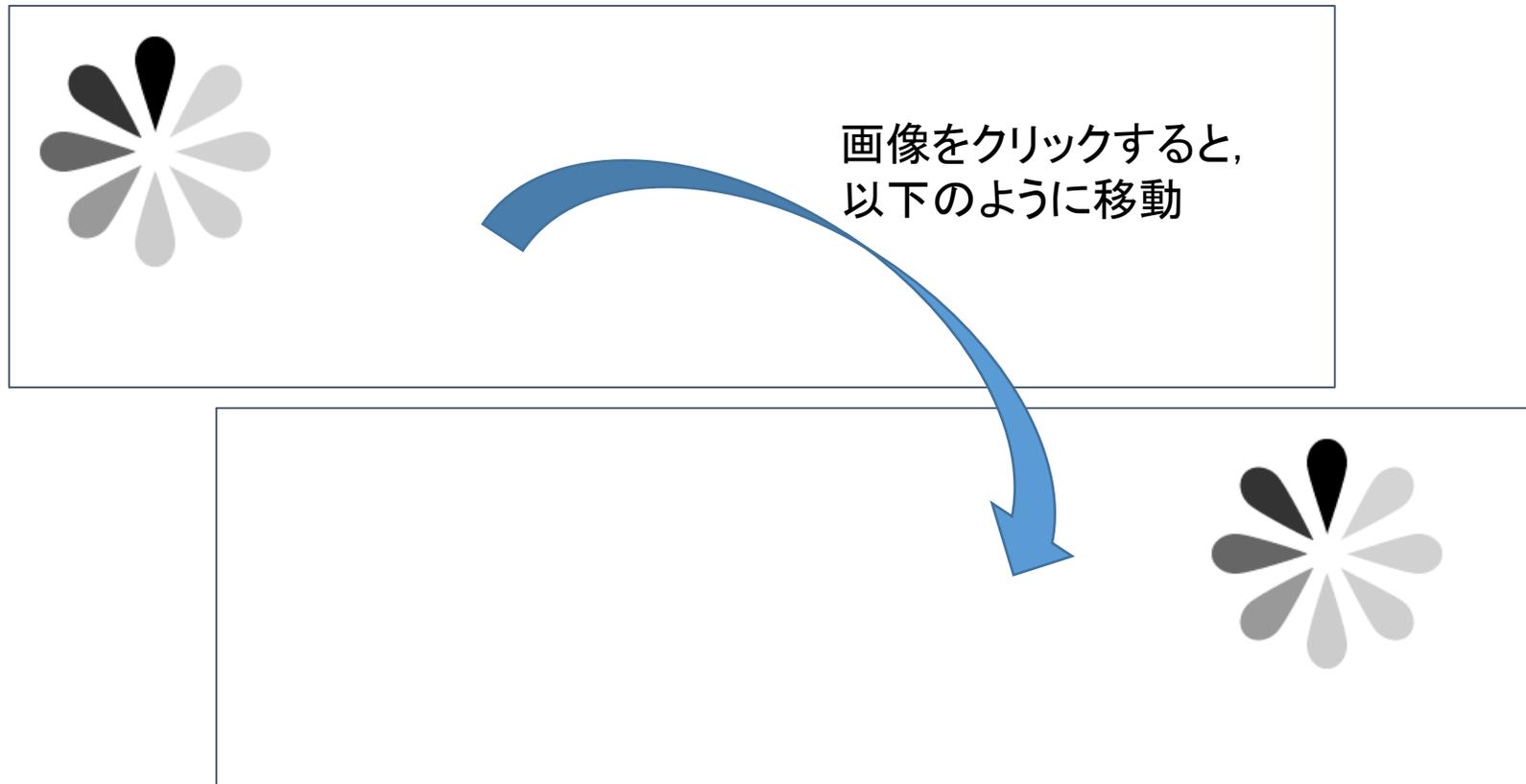
# stop () メソッド

- **animate ()** メソッドによるアニメーションは **stop ()** メソッドで中止できる

```
$('#image').animate({'left':'500px'}, 'slow', 'swing');  
$('#image').stop();
```

# 練習: animate () メソッド

- 画像をクリックすると  
決まった位置に動くプログラムを作ってみましょう



# Eventオブジェクト

- **click () メソッド や dblclick () メソッド に指定する関数に引数をとると Eventオブジェクト を取得できる**

```
$(document).dblclick(function(event) {  
  ...  
  var x = event.pageX; // 文章内のx座標  
  var y = event.pageY; // 文章内のy座標  
  ...  
}
```

- **より詳細はリファレンスを参照のこと**
  - <http://api.jquery.com/category/events/event-object/>
  - <http://semoooh.jp/jquery/cont/doc/event/>

# 演習

- アニメーションのあるページを作成して公開する
  - 基本課題
    - ボタンをクリックでばらばらマンガのスタートストップ
    - 画面上をダブルクリックで画像の移動
  - 発展課題
    - 画像の移動中にダブルクリックが行われたらどうするか
    - 画像の移動中はばらばらマンガを一時停止する
    - ボタンをクリックしたら「スタート」「ストップ」と文字を
    - クリックした位置に画像の左上がきている場合は、クリックした位置に移動するように変更する
    - 画像やデザインやアニメーションスピードの変更などさまざま
- どうしてもわからない人へのヒント  
<http://lss.oit.ac.jp/~t2015039/NetworkApplication/>
- ウェブページを**作成・公開**するとともに、その**認証情報(ユーザ名 & パスワード)**を**テキストファイルに記載**して、**次回の講義開始時まで**に、提出フォルダ(Xドライブ)に**ファイルで提出**
  - X:\N科専門\石井講師\ネットワークアプリケーション\第11回
  - ファイル名は「<学生番号>.txt」とする(ハイフンなし) 例: N14999.txt