

# Webデザイン実習2C

2017/11/08

Kazuma Sekiguchi

class@cieds.jp

# 告知

- Adobe XD Meeting #10
- 11月14日（火） 19:00～
- 参加費無料、懇親会は実費
  - 学生さんの参加歓迎です
- サイボウズ株式会社東京オフィス
  - 東京日本橋タワー 27階
  - 地下鉄日本橋駅B6出口直結
- <https://xd-study.connpass.com/event/71332/>  
にて受付中



# ブラウザの仕事



- ブラウザーが表示するのはHTMLデータなど
  - HTMLはファイルじゃ無くてデータでも良い
- サーバーからデータを送ってきてそれを展開するのが本来のブラウザの仕事

# クライアント側での処理

- ブラウザーはHTMLとCSSを表示
  - ユーザーの操作に合わせて表示を変える、HTML、CSSだけでは表現できない、利用できないものがどうしても発生してきた
  - CSSを動的に書き換えることで、アニメーションを実現する
  - HTMLを動的に書き換えることで、マウスオーバー時に画像を変える
  - ユーザーの動きにあわせてCSSを書き換えて、インタラクションを行う
- これらはCSSだけでは不可能なことが多い

# プログラミング言語

- ブラウザー側で利用できるプログラミング言語は JavaScript (JS)
  - 間違えてもJavaとは略さないこと（別な言語になる）
- JavaScriptを利用することで、ブラウザー上でさまざまなことが実現することが可能
  - HTMLの書き換え、CSSの書き換え、ブラウザー上でユーザーが実施した操作の取得などなど
  - 最近ではJSを利用して、デスクトップアプリやスマホアプリを作成することも可能
  - 極めて汎用性が高いが、習得が少し難しい言語

# プログラミング言語とは

- 何らかのデータを与えると処理をして、結果を返す
  - この集合体（だけじゃないけど）
- 例：自販機
  - お金を投入→金額と偽物じゃ無いか確認→問題ない
  - 商品選択→売り切れじゃ無いか確認→問題ない
  - 商品落としまーす
- いくつかの処理を組み合わせて目的を達成する
  - 重要なのは、データという概念、処理という概念

# プログラミング言語とは

- JavaScript

- プロトタイプ指向のオブジェクト指向言語（覚えなくて良い）
- ブラウザー上で動作して、HTMLやCSSを変化させて表示するための処理を行うことに特化したプログラミング言語
- 書き方が結構独特（他の言語を知らなければ気にならない）
- 複雑なことをしようとする、書き方が途端に煩雑になる
- 参考、サンプルが非常に豊富。探せば直ぐに求めているものが見つかる可能性が高い
- 結果が直ぐ分かる

# プログラミング言語とは

- 1文字でも間違えると動作しなくなる
  - HTMLやCSSは間違えても表示はされる
  - JSは1文字でも間違えると動作しない、最悪ブラウザが真っ白になる
    - どのプログラミング言語も同じではあるが、入力時注意
    - 打ち間違いはどうしても出るが、できるだけ減らす工夫をする
    - フォントを変更したり、文字を大きくしたり、誤字と思われるところにアンダーラインが引かれるようなエディタを試してみる
- 1度では大体動作しない
  - 慣れても何度書いても間違いは無くならない
  - エラーがあったときに自分で回復する能力を

# JavaScriptを書いてみる

- `<script></script>`の間に記述する
  - 別ファイルに記述してもOK
  - 別ファイルの場合は、拡張子をjsとして`<script src= "〇〇〇.js" ></script>`として読み込ませる
- 実行されるタイミングは基本的にブラウザでHTMLが表示された後

# 最近のJS

- ブラウザーでリッチな表現（アニメーションしたり、ドラッグして操作したりなど）を行うために欠かせない
  - インタラクシオンのほとんどがJSを利用する
  - HTML5タグのいくつかはJSを用いないと意味の無いものもある
- あちこちで書かれるようになってきた
  - JSの書きづらさが問題になってきている

# 最近のJS

## ネイティブ

- ECMAScriptの仕様に従って記述していく
- 一番汎用性が高いがJSの深い知識も必要

## AltJS

- 別の書き方をして、JSにトランスパイル(変換)する
- JSには無い言語上の仕組みが使えたりする
- JS以外に新しい言語を覚える必要がある

## ライブラリ併用

- ネイティブでは面倒な処理を簡単に書けるようにしたもの
- 最近はずっと少しく人気下がってきている

# jQuery

- 最近のウェブ上での表現はDOMを利用して見た目を変化させるのが多い
  - CSSでできるだけ表現を行う
  - アニメーションもCSSTransitionなどを利用する
- JSをネイティブで記述するのが一番良いが、記述量が多い
  - できるだけ記述量を減らしたりDOMへのアクセスを簡単にできた方が楽
  - 良く使用するような機能を予め用意しておく
- 代表的なJSのライブラリがjQuery
  - ライブラリ＝誰かが作成したある機能を利用するために簡単に利用できるように機能をまとめたもの

# jQuery

- 表現はDOMを通じたCSSの改変などが多い
  - バックエンドシステムと連動してユーザビリティを向上する目的でJSが利用されるのが最近
  - SPA (SinglePageApplication) など
    - GoogleMap、Gmail、Twitter、Facebookなどのようにリロード無しで機能が利用できるようなもの
    - ほとんどがJSを上手く利用することで実現している
- SPAを実現するライブラリー群とjQueryはイマイチ相性が良くない
  - JSの進化と共にjQueryの機能に近いものがJS自体に搭載され始めている

# jQuery

- jQuery不要論の登場
  - ”*You Don't need jQuery*”  
<https://github.com/oneuijs/You-Dont-Need-jQuery>
  - jQueryを使うと速度が落ちる（事実）
  - 返って開発効率が落ちる、代替できる手段が提供されてきている
  - jQueryを使わないで、表現を実現したりする
- 現在のウェブは速度重視に移行しつつある
  - 1秒でも早く描画される、利用できる、ユーザビリティが向上できるというところに力点
  - コーダー（フロントエンドエンジニア）に求められる能力

# jQuery

- jQueryは比較的枯れた技術
  - 最先端では無いが、未だに利用されるケースは多い
    - ES6, Babel, webpack, Angular4, React, Riot.jsとかが先端かな
  - 利用される = それなりに便利。恐らく消えて無くなる可能性は低い
  - jQueryベースで書いたものをJSネイティブに移行するのはさほど難しくは無い
- 現在も開発は続いている
- プラグインなどの豊富な資源
  - jQueryをベースにさまざまなプラグインが登場
  - 例：モーダルウィンドウ、スライドショー、パララックス表現などなど
  - 現在もある程度プラグインは利用できる

# jQueryの利用

- jQueryはダウンロードして利用
  - <http://jquery.com/download/>からダウンロード可能



The screenshot shows the jQuery website's download page. The navigation bar includes links for Download, API Documentation, Blog, Plugins, and Browser Support, along with a search box. The main content area is titled "Downloading jQuery" and provides instructions on how to download and use the files. A callout box points to the link "Download the compressed, production jQuery 3.2.1", which is highlighted with a blue border. The callout box contains the text "通常はこれだけダウンロード" (Usually, just download this).

**Download** API Documentation Blog Plugins Browser Support Search

## Downloading jQuery

Compressed and uncompressed copies of jQuery files are available. The uncompressed file is best used during development or debugging; the compressed file saves bandwidth and improves performance in production. You can also download a [sourcemap file](#) for use when debugging with a compressed file. The map file is *not* required for users to run jQuery, it just improves the developer's debugger experience. As of jQuery 1.11.0/2.1.0 the `//# sourceMappingURL` comment is [not included](#) in the compressed file.

To locally download these files, right-click the link and select "Save as..." from the menu.

### jQuery

For help when upgrading jQuery, please see the [upgrade guide](#) most relevant to your version. We also recommend using the [jQuery Migrate plugin](#).

[Download the compressed, production jQuery 3.2.1](#)

[Download the uncompressed, development jQuery 3.2.1](#)

[Download the map file for jQuery 3.2.1](#)

You can also use the slim version:

[Download the compressed, production jQuery 3.2.1 slim build](#)

[Download the uncompressed, development jQuery 3.2.1 slim build](#)

[Download the map file for the jQuery 3.2.1 slim build](#)

[jQuery 3.2.1 release notes](#)

通常はこれだけダウンロード

# jQueryの利用

- `<script src= “jquery.min.js” ></script>`  
として自分が書くスクリプトの前にjQueryを読み込ませる
- 自分のスクリプトは別途`<script>`を記述してその中に記述する

```
<script src=“jquery.min.js”></script>
<script>
$(function(){
  //ここに自分の実現したいことを書く
});
</script>
```

# jQuery

- `$(操作する対象).操作(引数);`
  - 基本的な文法としてはこれだけ
  - ドットで連結していく場合もある
- 操作する対象には、CSSのセレクタと同じ記述でOK

`$("#text")` → id="text" に対して何か行う

`$(".photo")` → class="photo" に対して何か行う

`$("a")` → aタグ に対して何か行う

`$("#contents li")` → id="contents" 内にあるliタグ に対して何か行う

## classを当てる、外す

- jQueryで要素に対し、動的にclassを適用（追加）したり外すことが可能

```
$("#div").addClass("text");//textクラスを追加
```

```
$(".blueButton").addClass("move");//moveクラスを追加
```

```
$(".blueButton").removeClass("move");//moveクラスを除去
```

```
$("#h1").removeClass("redText");//redTextクラスを除去
```

## classを当てる、外す

- classを適用する場合は、セレクターを指定し、addClassメソッドを使って、適用したいクラスを指定する
  - 適用したいclassを記述するときは先頭の「.」は取る
- classを外したい場合は、セレクターを指定し、removeClassメソッドを使って外したいクラスを指定する

# イベント

- ブラウザー上でユーザーが何か行う操作がイベント
  - クリック、スクロールバーの移動、リサイズ・・・・。
  - ブラウザーは常にイベントを監視している
  - ユーザーの動きに合わせてブラウザーの表示を変化させている
- JSではこのイベントを受け取ることが可能
  - クリックした時に何かする、スクロールバーが移動したときに何かする、ということが可能
  - CSSにはイベントを受け取る機能がほぼない（唯一hoverだけ取得できる）

# イベントの取得と活用

```
$(“a”).on(“click”,function(){  
    $(“p”).addClass(“active”);  
});
```

```
$(“a”).on(“click”,function(){  
    $(this).addClass(“move”);  
});
```

- クリックの場合、クリックされたらイベントを取得したいセレクターを指定し、onメソッドで繋ぐ
  - “click” などのようにイベント名を記述
  - function() { }を記述し、中で行いたいことを記述する
- クリックされた要素自体に何かしたい場合は、thisが利用可能
  - thisの場合は、” ” が付かないので注意

# jQueryでのイベントの書き方

- クリック→click,mouseDown,mouseUp
- ダブルクリック→dblClick
- マウスを乗せた→mouseover/mouseEnter
- マウスを離れた→mouseout/mouseLeave
- ページが読み込まれた→load
- ページから離れた→unload
  
- 他にもいろいろ
- 一番良く利用されるのはクリックイベント