コンテンツメディア
プログラミング実習 Ⅱ
JQuery

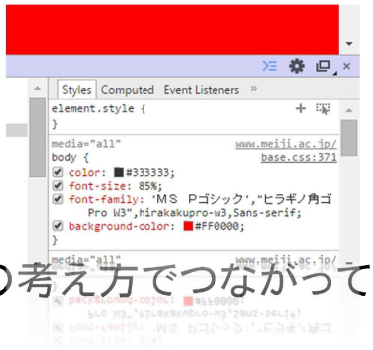
中村　宮下　斉藤　菊池

Chromeでウェブサイトに
いたずらしていたのを思い出してw

「HTMLやCSSの要素を書き換える」

これをプログラム（JavaScript)から行うことによってより動的な体験を提供していく

【今日のおしながき】



・JQuery

・Chrome拡張入門

両方ともに「JavaScriptによる書き換え」の考え方でつながっている

前回および前々回とのつながりやオーバーラップも多めにしてあるので、ぜひ過去の資料も見返しながら進もう！

Web上で動く簡単なシンセ（FMS1年　小渕 豊さん制作　@FMS_Cat ）

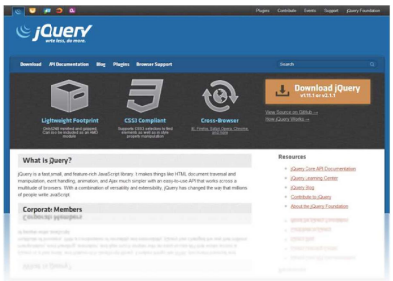
http://fms-cat.github.io/synth

jQueryやJQuery Knobを駆使、
独学で1週間ほどで制作されている

CMP実習２は「課題」という呼び方をしているが、
実装に対するモチベーションや姿勢はCE概論やプログラム実習１・２と同じでOK！

jQuery（復習）

- アプリケーションフレームワークのひとつ
- 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いため，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の基本（復習）

「$()」は「jQuery()」の略

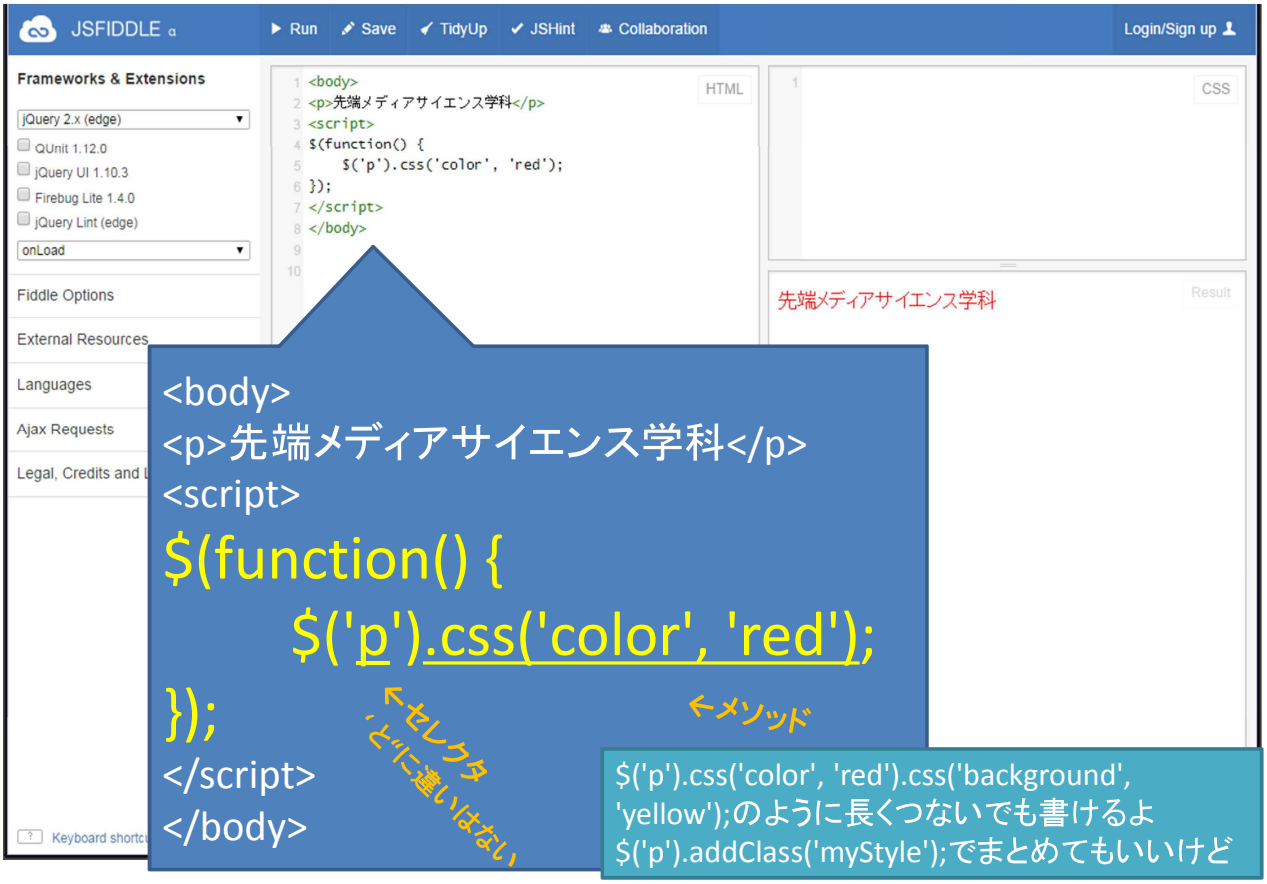
– 本来は

- jQuery(function(){ ... });
- jQuery('#hoge').on(...);
- jQuery('#hoge').css(...);

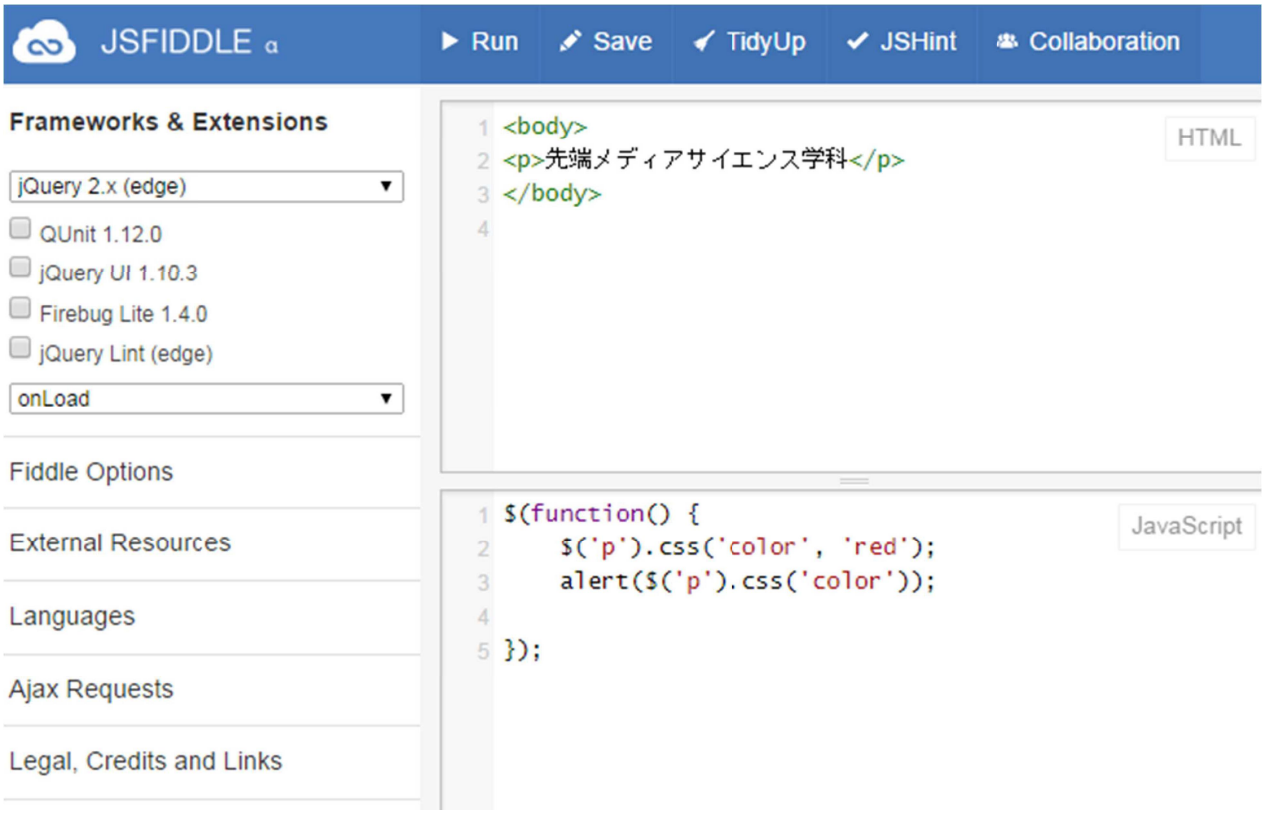
```
$(function(){
    プログラム
});
```

でjQuery に関するプログラムを書く

http://jsfiddle.net/ で軽く復習（1ファイルにまとめる編）



こう書いたほうがわかりやすいかも



セレクタ（復習）

elementを選択するには

- tag : $("tagname")
- id : $("#idname")
- class : $(".classname")

カンマで複数選択もできる

<p id="one">, <p class="one"> などと書いたタグに適用して実験してみよう

エフェクト！(括弧内はミリ秒)

- $('p').show(1000);
- $('p').hide(1000);

- $('p').fadeIn(1000);
- $('p').fadeOut(1000);

- $('p').toggle(1000);
- 表示状態にあるものを非表示にし、非表示状態にあるものは表示状態に

ほかにもいろいろ指定方法がある

```
$("#idname > .classname").css('color', 'red');
```

→ idnameの直下のclassnameだけを指定

半角スペースだとそれ以下の要素を全部指定になる

```
$("#idname").children().css('color', 'red');
```

→ idnameの子要素すべてを赤くする

属性セレクタ

リンク
があるとき、

```
$('a[href="http://www.meiji.ac.jp/"]').css('color', 'red');
```

で適用できる。

このときの=は等号なので、!=で≠の意味になったりする。
*=は「含まれる」という意味になるので、「.ac.jp」ドメインへのリンクだけまとめて赤くするとかできる！

値の取得

```
alert($('p').css('color'));
```

pタグの色を取得して出力
jsfiddleのプログラムに追加して確認してみよう

※出力の仕方はいろいろあったよね…

- console.log("出力したい内容");
- alert("出力したい内容");
- document.write("出力したい内容");

属性の取得・変更

リンク
があるとき、

```
alert($('a').attr('href'));
```

のように取り出したり、

```
$('a').attr('href', 'http://miyashita.com/');
```

のように変更したりできる

タグの中身の変更

```
$('p').text('中身のテキスト変更');
```

```
$('p').html('<HTMLタグ>');
```

.empty()、.remove() で、
要素を空にしたり削除することもできる

コールバック関数

第二引数として関数を書くと、処理が終わったときにそれを実行してくれる

```
$('p').fadeOut(1000, function() {
        alert("Goodbye");
});
```

イベント

```
$('p').click(function() {
        alert("hello!");
});
```

のように、

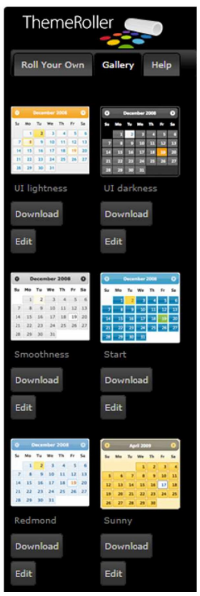
イベントが起こったら実行してほしいことをfunctionとして括弧内に書く

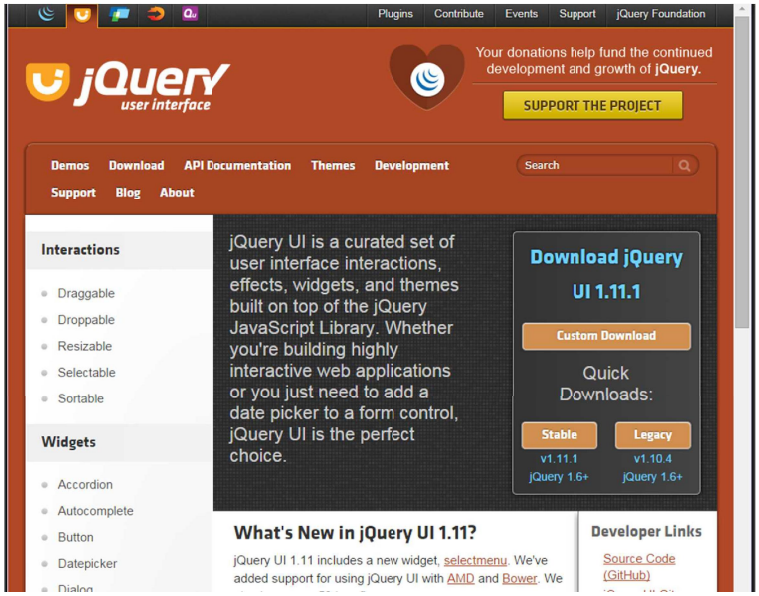
イベント一覧（復習）

- 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あたった時
- 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

jQuery UI (jqueryui.com)

- ダウンロード時にテーマを選ぶ

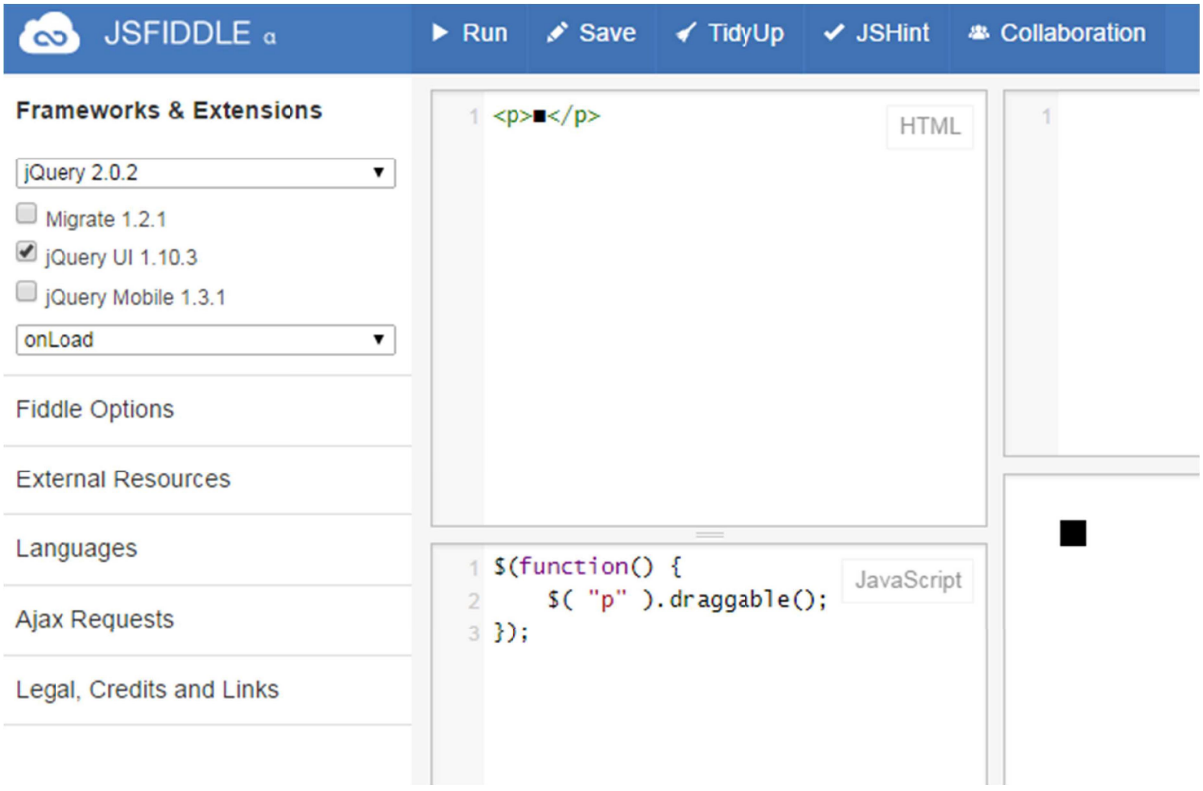




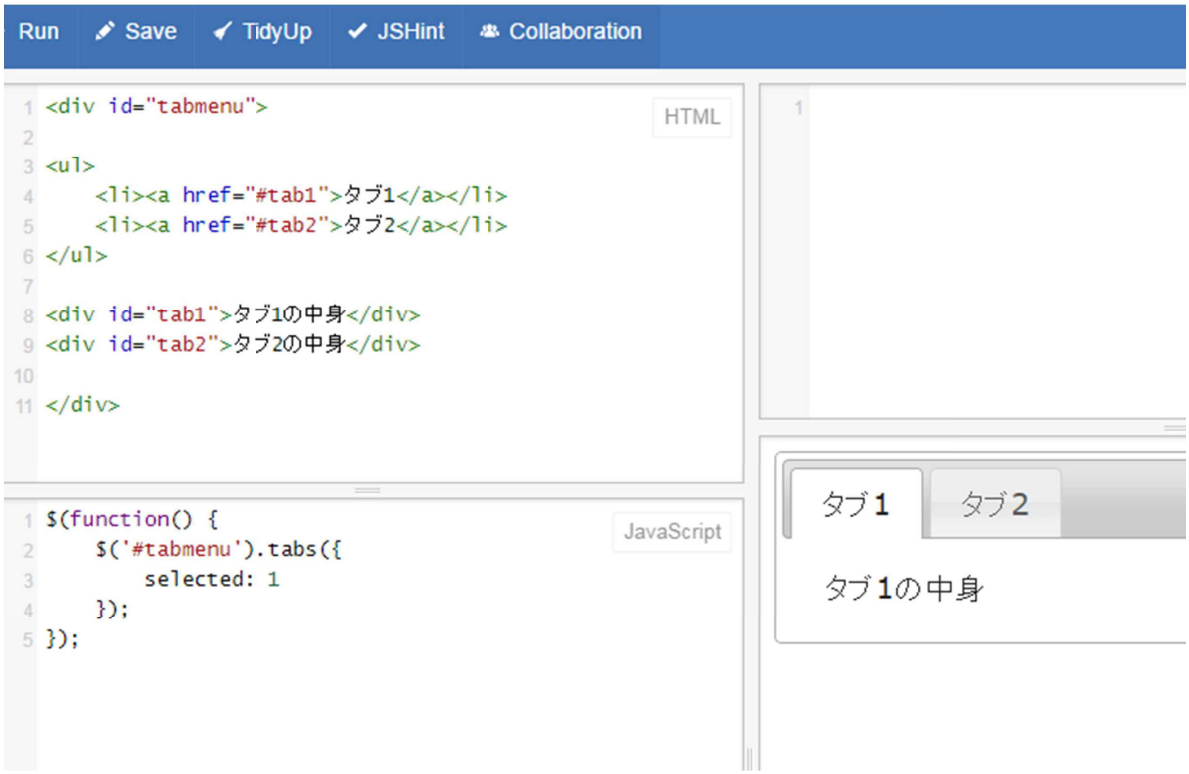
弾むエフェクト

（ http://jqueryui.com/effect/ ）

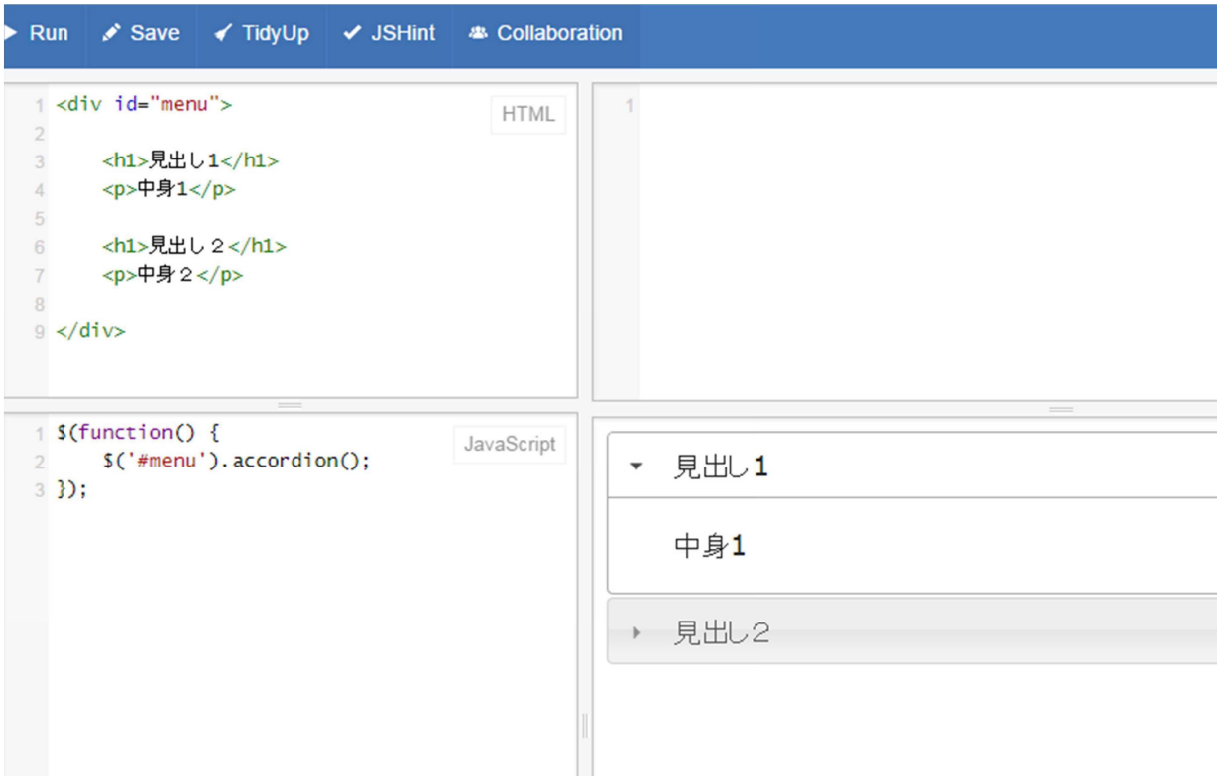
段落をドラッグ可能にするw



簡単に作れちゃうタブメニュー



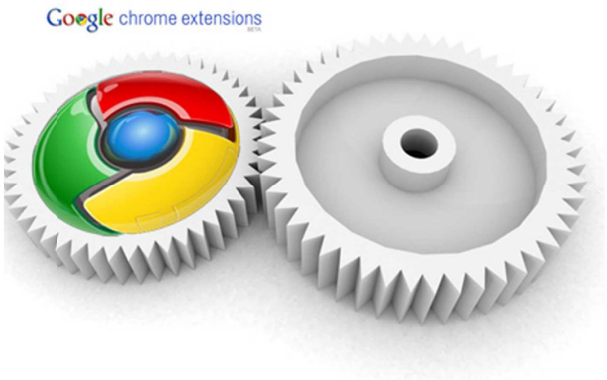
簡単に作れちゃうアコーディオン

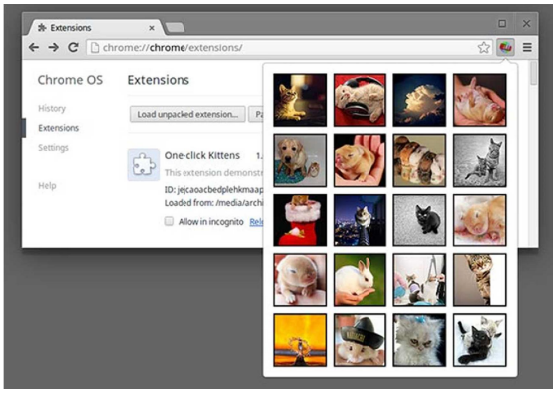


課題

- jQueryおよびjQuery UIのAPI Documentationを見て、テキストに載っていない仕様を試してください！！

- jQueryおよびjQuery UIを組み合わせて、派手でインタラクティブなウェブサイトを作ってください！！





Chrome拡張　入門

https://developer.chrome.com/extensions/getstarted

コンテンツ・メディアプログラミング実習II
第５回（2）デバッグの実際

3組　宮下　中村

4組　菊池　斉藤

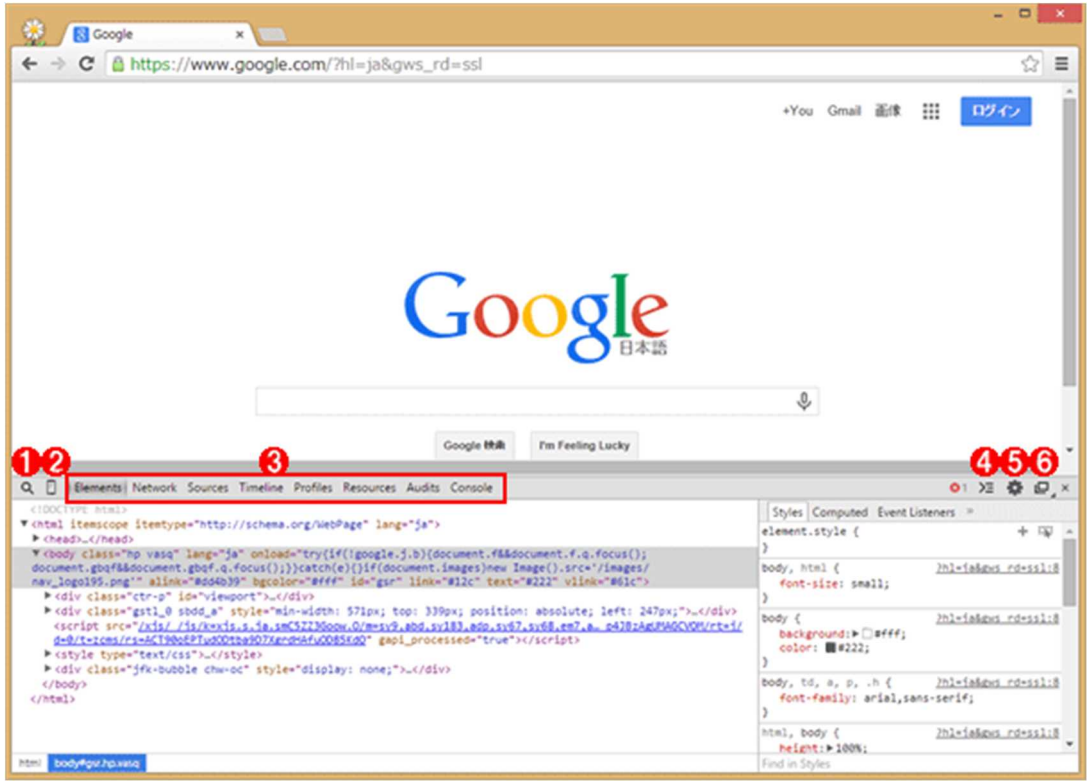
「**Google Chrome Developer Tools（DevTools）入門**
Web開発でよく使う、特に使えるChromeデベロッパー・ツールの機能」を参考にしています
http://www.buildinsider.net/web/chromedevtools/01

他にも　http://gihyo.jp/dev/feature/01/devtools
　http://shim0mura.hatenadiary.jp/entry/20111231/1325357395　などたくさんあります

ショートカットもちょくちょく覚えるといいです！
https://developer.chrome.com/devtools/docs/shortcuts

Chromeのデベロッパーツールに もっと慣れよう

- F12での起動　「要素の検証」での起動
- ESCで閉じる　右上の×クリックでも可



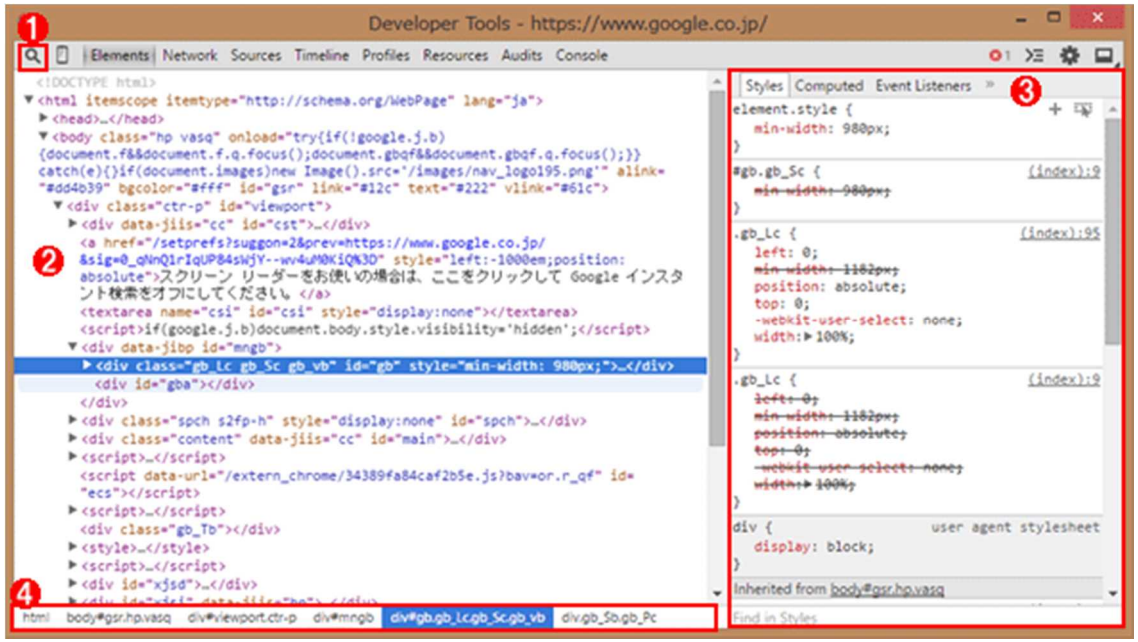
「**Google Chrome Developer Tools（DevTools）入門 Web開発でよく使う、特に使えるChromeデベロッパー・ツールの機能**」より
http://www.buildinsider.net/web/chromedevtools/01

1. Webページ上の要素を、マウスを使って選択できる。虫眼鏡アイコン。
2. スマートフォンなどの表示を確認するためのエミュレーション・ウィンドウの表示／非表示を切り替える。デバイスモード・アイコン。
3. 各タブは「パネル」と呼ばれており、このパネルで機能を切り替えることができる。
4. デベロッパーツールの下部へのドロワー（※その中にはConsole／Search／Emulation／Rendering タブがある）の表示／非表示を切り替える。ドロワー（＝引き出しを意味する「drawer」）は、実際に引き出しのように下からスライドして表示されたり、隠されたりする。
5. デベロッパーツールの詳細な設定をする設定ダイアログの表示／非表示を切り替える。
6. デベロッパーツールを別ウィンドウにするか、下辺か右辺のいずれかにドックするかを切り替える。

［Elements］パネル

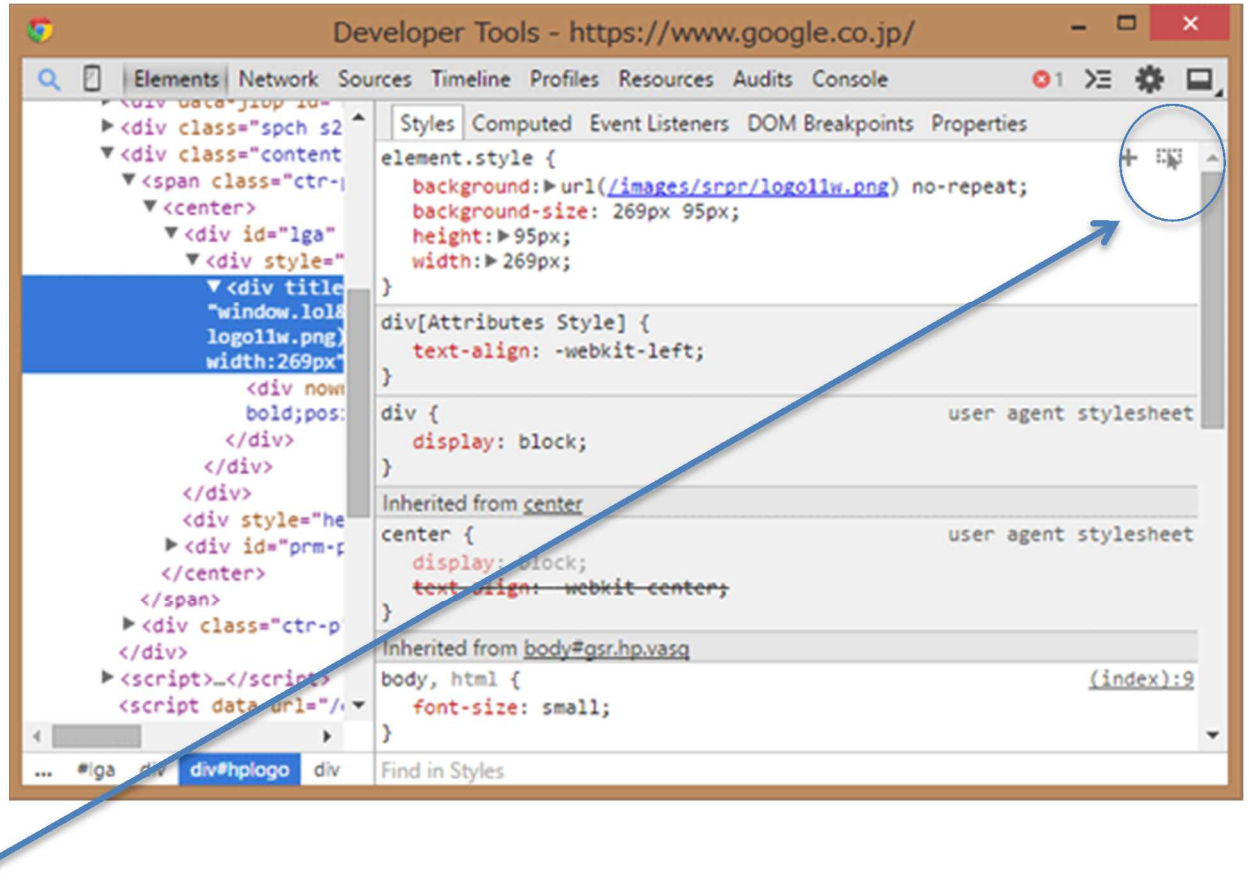
すでにこのパネルは慣れていると思いますが復習

右クリックから「Edit as HTML」を選ぶと要素ごと編集できて便利

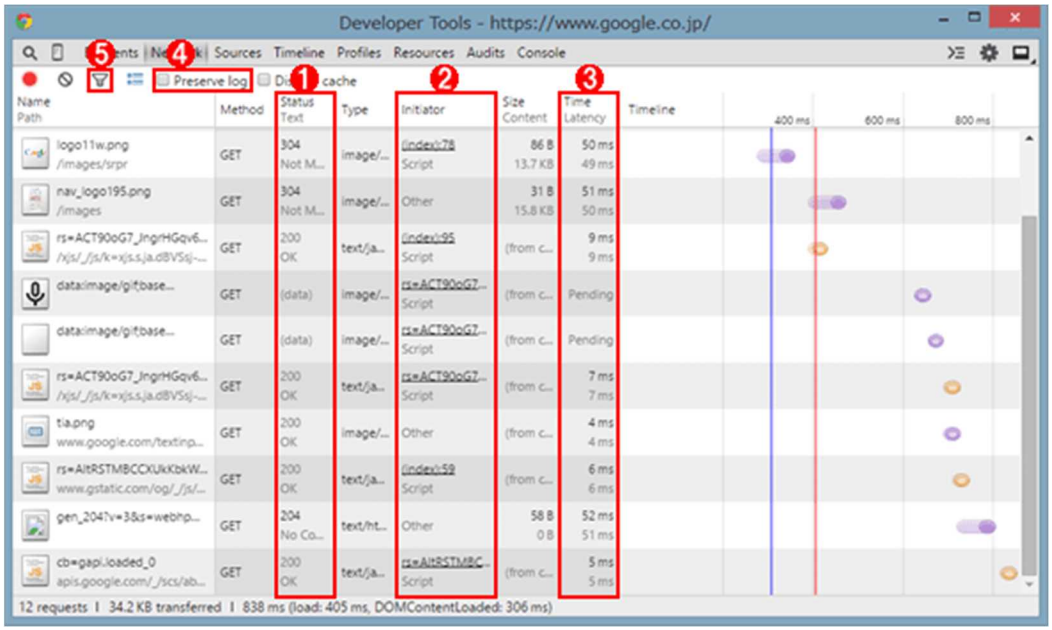


要素に設定されている属性やスタイルを確認可能

- 「user agent stylesheet」というのはブラウザに設定されているスタイルのことね！
- 継承されたスタイルは「Inherited from …」って書いてあります。
- 数値を矢印キーで上下させられたりカラーパレットを出せたり色々便利
- a:hoverなど、a要素におけるCSS　(疑似クラス)をいじるときはこれを押す



［Network］パネル



ページのリクエストとダウンロードされるまでの時間が表示される。

①［**Status**］**列：** HTTPステータス。

②［**Initiator**］**列：** ファイルを呼び出す起点。例えばJavaScriptコードから読み込まれたときには、そのJavaScriptファイル名が表示される。

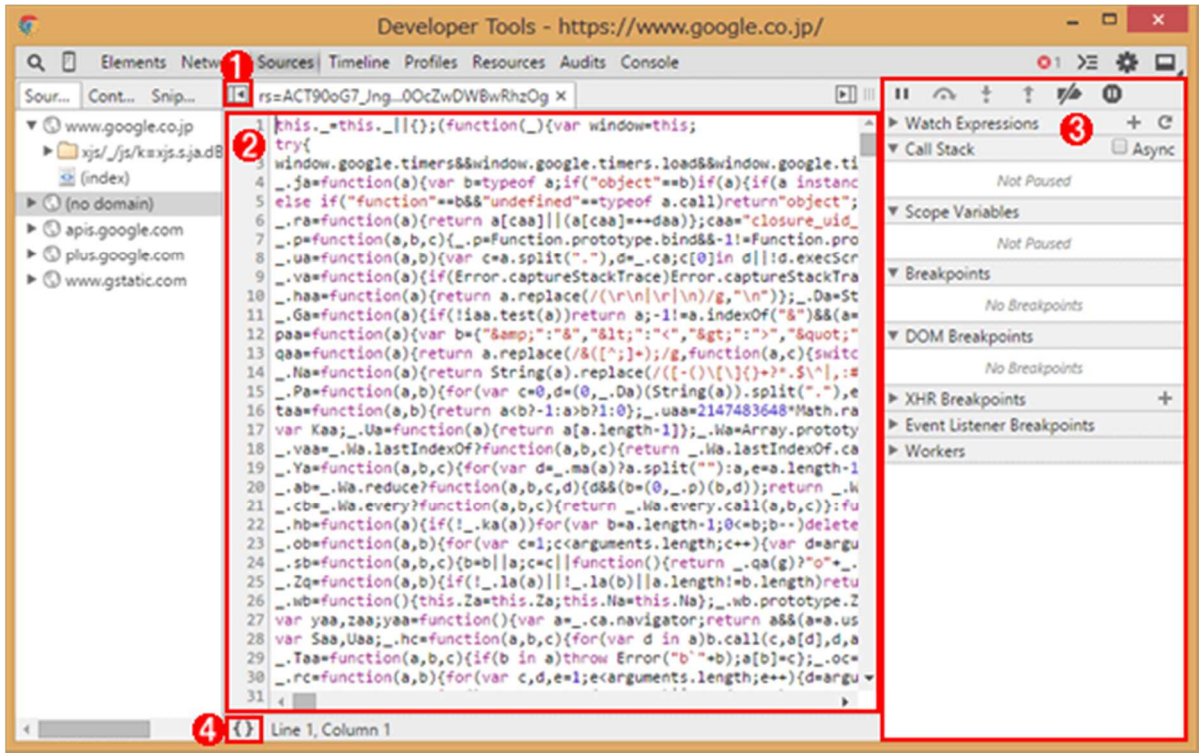
③［**Time**］**列：** ダウンロードにかかった時間。上がリクエストから受信が完了するまでの時間。下がリクエストから受信開始するまでの時間。

④［**Preserve log**］**（ログの保持）：** ページを遷移してもログを残しておくように設定する。

⑤**フィルター：** フィルターアイコン（）をクリックすると（のようにアイコンの色が変わり）、リクエストの種類をフィルターして特定の項目だけの表示に切り替えられる

[Sources]パネル

CSSやJavaScriptをデバッグできる！！



① ページのソースファイルを一覧表示するnavigatorを表示

② ソースビュー

③ サイドバー

④ コード整形ボタン（便利！）

Ctrl+Sの保存はブラウザ内の保存にすぎないので注意！

（ファイル名を右クリック「Local Modifications...」で見られる）

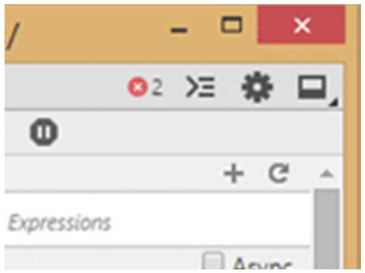
ほんとに保存したいときは「Save As...」を選んでください

JSのエラーをデバッグ

- JSの文法エラーがある場合は右上に印が出る
- コンソールにエラーメッセージが出る

  右にどこでエラーがあるかも出ているので、

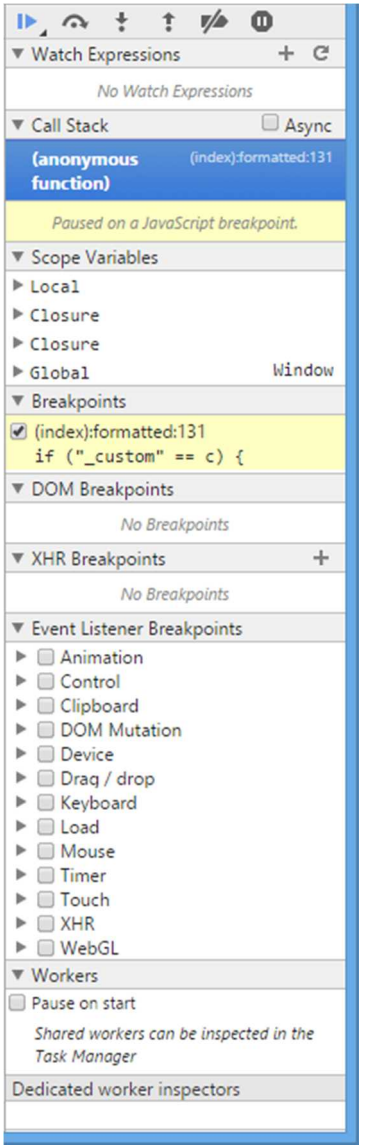
  クリックしてジャンプ！



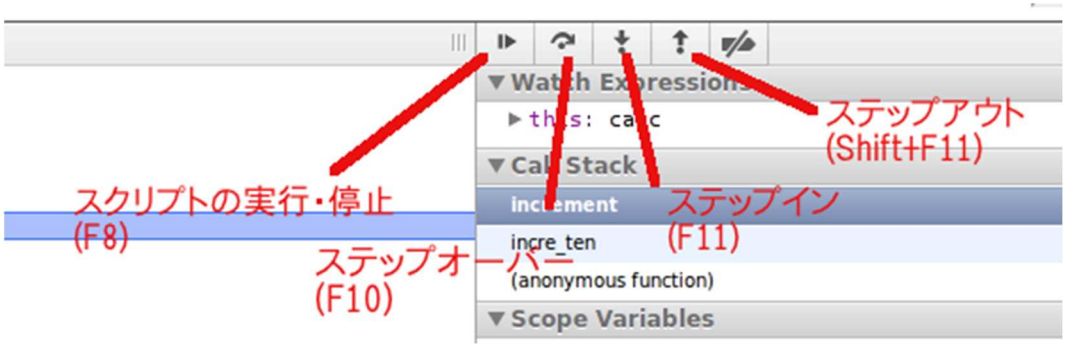
正しくJSプログラムを用意し、それを少し壊して実行してみて、どういうエラーが出るか試してみよう！

デバッガー機能のブレークポイントを使おう

- 行番号をクリックするとブレークポイントが設定できる
  そこで処理が止まる
- さらにマウスオーバーすると、それぞれの変数の中身とか
  みんな見れちゃう！！



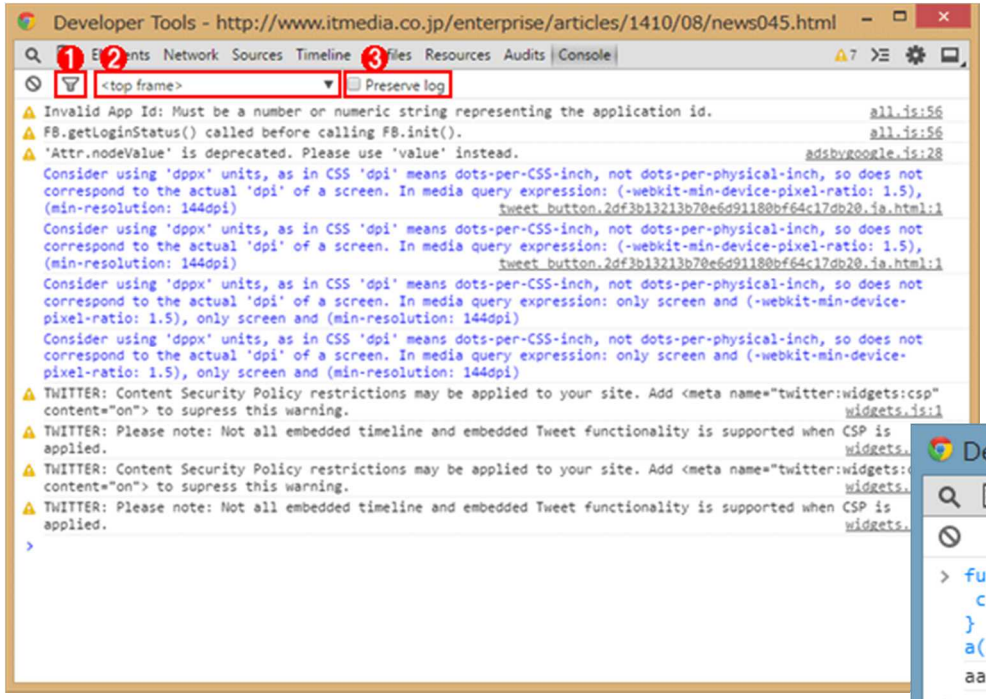
- 関数の中に入る／入らないステップ実行がある

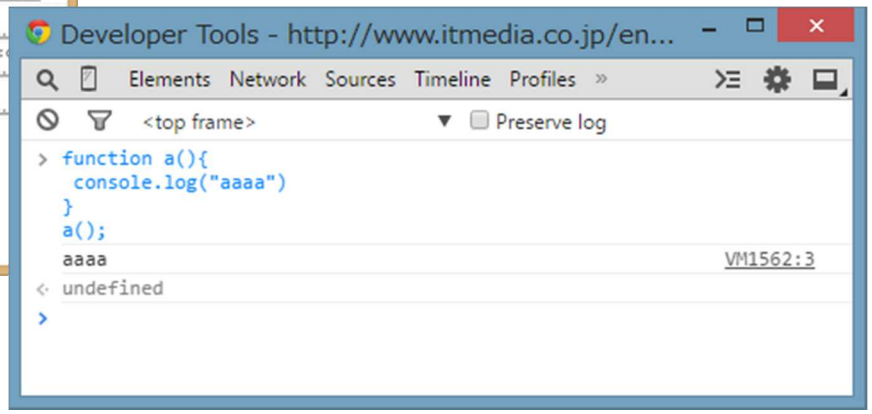


JSのプログラムを用意して試してみよう

［Console］パネル

ページのエラー等の情報表示　およびコマンド入力が可能
オートコンプリートも便利
Shift+Enterで複数行の入力も可能(Enterで実行ね)





① フィルターの表示／非表示を切り替える。
② コンソールログを表示するフレームを選択する。
③ チェックを付けると、ページを遷移してもコンソールログを残すことができる

コンソール上で使える特殊な関数

$	document.getElementByIdのショートカット
$$	document.querySelectorAllのショートカット
$x	document.evaluate(XPath)のショートカット
$1, $2, $3, $4	$1から$4にはコンソールでの実行結果が4回分だけ記憶されています。
copy(text)	引数に渡した文字列をクリップボードにコピー
dir(object)	引数に渡したオブジェクトを解析（DOM要素を渡したときもオブジェクトとして扱う）
dirxml	引数に渡したノードをツリー表示
inspect	引数に渡したオブジェクトに応じて適切に解析を行う（localStorageを渡すとStorageパネルに切り替えるなどの処理も行う）。
keys	オブジェクトのプロパティを配列で返す
monitorEvents	引数に渡したDOMについて各種イベントを監視する。第2引数で監視するイベントの種類を制御できる。
unmonitorEvents	monitorEventsを解除
profile	JavaScriptのプロファイリングを開始する
profileEnd	JavaScriptのプロファイリングを終了する
values	オブジェクトが持つ値を配列で返す

Webアプリケーション構築ってしんどいよね

- いろんな言語を使う
- いろんなAPIを使う
- 複合的に使う
- 連携して使う
- よくわからないまま使う（詳細な仕様なんて知らん）

付け焼き刃の理解のまま「勘でいじって」組み合わせていくのは、はっきりいって、ストレスが大きいかもしれない。

でも、スタンドアロンアプリでは絶対できないことができる！

ネット上の「大きな力」を手に入れることができる！

心構え的なこと

- 実は知識量の差ではない！
- プログラミングが得意な人がプログラミングしてるのを観察してみよう！
  - バグっても平静を保っている
  - なかなかデバッグできなくても全然焦らない
    「きっといつかバグはとれる」という確信があるのかも
    「最初は動かなくて当たり前」と思ってるのかも
    物事を怖がらない

もし親にパソコンやスマホについて質問されたとしたら？
しかもそれが慣れ親しんでいないOSの、未知の質問だったとしたら？
…でも、けっこう解決できる。なぜか？

サーバーサイドプログラミングを学んだ後の「応用実習」に向けて

- 配られた資料が全部理解出来なくてもできるかも
- 配られた資料だけで十分ではないかも

→ **「調べながらプログラミング」する姿勢が大事**

英語のドキュメントでもビビらない

自信がない人は本屋に行ってみよう！

JavaScriptやJQueryなどを冠した本が山ほどある

課題（レポート）

クライアントサイドアプリケーション制作において、
どのようなバグをどのようなデバッグ手法で直したか、
複数の事例とともにレポートとして提出してください

※単純な文法エラーではなく、「エラーがないのに意図した動作にならない」バグを対象とします