準備体操　Google Chromeを使ってみよう

Step1：Google Chromeでファイルを開く

　各種テキストエディタと同じようにファイルをGoogle　Chromeドラッグ＆ドロップすることで、任意のファイルを開くことが出来ます。

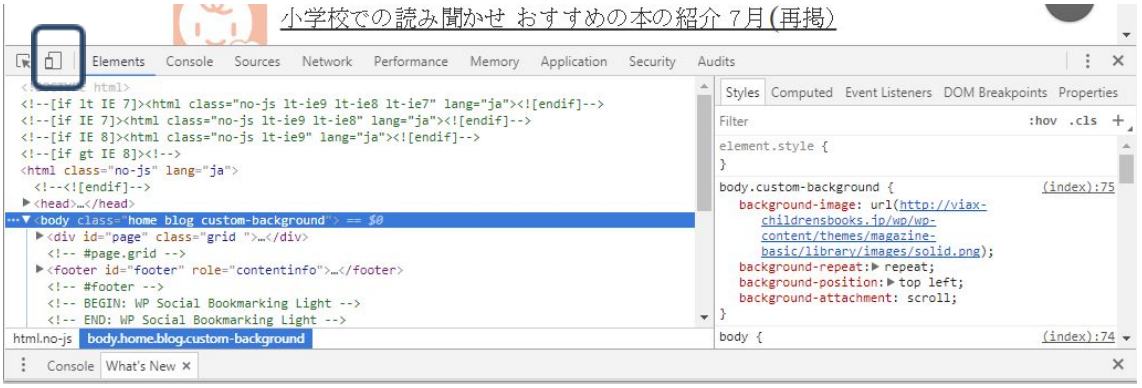
Step2：ディベロッパーモード（開発者モード）にする

　まず、アドレスバーに「本のこまど」と入力しEnterを押します。

　「本のこまど | ヴィアックステクニカルサポート室が運営する児童サービス支援 ...」が一番最初の検索結果になると思いますので、これをクリックします。

　「本のこまど」のトップページに来たら、「F12」を押してください。

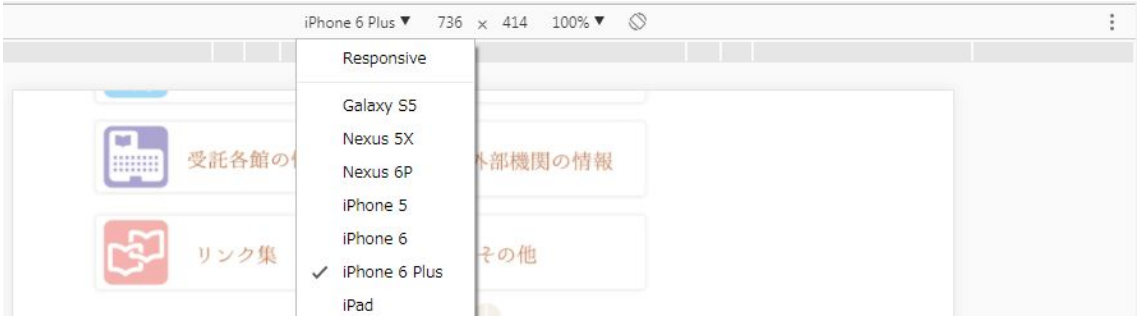
　ディベロッパーモードになり、ウィンドウの下側に以下のような表示がされます。



　まずは「スマホモード」に切り替えてみましょう。

　このボタンを押すと、スマホモードになります。

　もう一度押すとPCモードに代わります。



画面上部からさまざまなデバイスでの見え方を確認できます。

Step3：ページを書き換えてみよう

　まず、スマホモードになっている場合はPCモードに戻してください。

次に、適当なリンクにマウスカーソルを重ねて、右クリックし、「検証(I)」を選択します。

まずは書かれている文書を変更してみます。
この例では「４月のおすすめ本リスト」となっているので、「４月のおすすめ本リスト」の部分を下の画面の「反転部分」から消してみましょう。
Webページの方も文字が消えるはずです。

次に文字の大きさや色を変えてみましょう。
開発者モードの右側のelement.style{　とある部分に指定することで、文字の大きさや色を変えられます。
まず、文字を大きくしてみましょう。
font-size:150%;　と入力してください。
fを入力すると、入力補助が出ますので、それを使うと入力しやすいです。
指定した文字の大きさが1.5倍になったはずです。数字を変えることで、さらに文字を大きくしたり、小さくすることが可能です。

次に文字の色を変えてみましょう。
color:#f66; と入力すると文字が赤っぽくなります。
colorでは#赤緑青の順に16進法（0123456789abcdef）で色を指定することが出来ます。fが一番強い発色で、すべてをfにすると白に、すべてを0にすると黒になります。

font-sizeや、colorといった指定をして見栄えを変えるものを（カスケード）スタイルシートといい、単に「スタイル」とかCSSなどとも言われます。

Step4：再読み込み（リロード）

これからのワークショップでは、各種テキストエディタでファイルを上書き保存した後、再読み込み（リロード）をすることが多いです。

Google　Chromeでは[F5]キーを押すか、更新ボタンを押すことで、再読み込み（リロード）することが出来ます。

今回は、再読み込みをすると、手を入れる前の画面へ戻ります。

この後の実習の間、あとで説明する各種テキストエディタで上書き保存した後、Google Chromeを再読み込み（リロード）して、保存内容を反映させる、という動作を頻繁に行います。

できるだけ早く、上書き保存＆リロードに慣れてください。

書誌・書影系APIを使ってみよう

書影や書誌情報の提供サービスを使ってみましょう。
[ex_04]フォルダ内のbooklist.htmlをGoogle　Chromeへドラッグ＆ドロップしてみましょう。

上のような画面が開きます。

Step1-1：まずは適当にISBNを入力してみるAPIを使ってみる
1）まずは例として、次の3つのISBNで試してみましょう。
Ex: 9784818823556,　9784408451749,　9784794967213
ISBNの区切りには「カンマ」を使用して下さい。空白は不要です。
入力が終わったら「書誌データ取得」ボタンを押してください。
3件のデータが表示されます。書影は2件のみの表示になるはずです。
リンクになっているURLをクリックすると、カーリルを使って蔵書検索ができるようになっています。

2）各種テキストエディタを使って検索対象を修正する。

◎各種テキストエディタでファイルを開く
＊今回はドラッグ＆ドロップの方法を使います。
開きたいファイルにマウスカーソルを重ねて、マウスの左ボタンを押しっぱなしにし、マウスカーソルを各種テキストエディタのアイコン（もし各種テキストエディタがすでに開いている場合にはそのウィンドウ）に重ねてください。

まず、booklist.htmlを各種テキストエディタへ「ドラッグ＆ドロップ」して下さい。

```
<!doctype html>
<html lang="ja">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <!-- Bootstrap core CSS -->
  <link rel="stylesheet" type="text/css" href="https://stackpath.bootstrapcdn.com/bootstrap

  <title>API Workshop</title>

  <style>
  h1 {;} /*見出しのデザイン*/
  .comment {;} /*解説のデザイン*/
  .book_title{;}/*本のタイトル*/
  </style>
```

こんな感じにテキストファイルが開けます。

テキストエディタについて

Wordなどの文章を作るソフトの場合、文章を記述するのと、デザイン（文字の大きさや色を変更したり）が一緒に出来ますが、テキストエディタの場合は、文章を記述する事しかできません。その変わり、メモ帳など、他のテキストエディタでもファイルを編集することが出来ます。

■カーリルのシステムIDの調べ方

カーリルの検索対象をシステムIDを使って指定します。

手順1）Chromeで新しいタブを開き、アドレスバーに「カーリル」と入力して検索をします。

検索結果の先頭に出るはずの「カーリル ｜ 日本最大の図書館蔵書検索サイト」をクリックし、カーリルのサイトへ移動します。

手順2）「図書館マップ」から都道府県を選び、自分の図書館を選びます。

図書館マップから都道府県を選び、自分の図書館を選ぶ。

自分の図書館の情報が表示されたページきたら、IDを調べます。

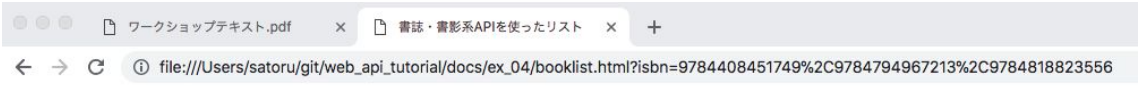
IDは、図書館‥‥というところに書かれているもので、野田市の場合は、Chiba_Nodaとなっています。

同じ自治体（同じシステム）の図書館であれば、本館でも分館でも変わらない表示です。

Step1-2：本を変更してみる。

　好きな本のISBNを入力してもらっても、研修室にあるものを使っても構ません。
　さきほど入力したリストを消すために、再度、booklist.htmlをGoogle　Chromeへドラッグ＆ドロップするところから初めて下さい。
　再読み込みをしても、アドレスバーに入力したISBNが残っているのでリストは消えません。（ちなみにブックマーク登録すると、そのブックリストを呼び出すことが出来ます。）

ここにISBNが残っています。



＊お願い
　今回は、1部屋から何人ものひとが同時にAPIにアクセスするので、念のため、一度に表示するのは多くても5冊程度にしてください。

それぞれのAPIの特徴

openBD：クレジット表示義務もなく、ほぼ自由に使ってよいことになっているAPI。2017年1月から新しく始まったAPIなので、新刊に強く旧い本に弱い。一度に1000件までのデータを取得できる（他のAPIは1冊づつ取得）。ISBNさえ分かれば、刊行前の書誌も取得可能とのこと。書影はGoogleBooksAPIより綺麗なぶん、少し重い。

GoogleBooks：アクセス制限（1000件/1日）があるので、Web-OPACの一覧画面などでうっかり使うと、表示されなくなることも。しかし速度が速く安定している。また、英語多読などの外国の資料も表示できる。新刊の登録はやや遅い。

CiNii：利用には登録が必要（しなくても使える）。プログラムを書く側からすると使いやすい仕組みになっている。主に大学図書館のデータで成り立っているため、公共図書館で使う場合は、思わぬ書誌が無いことがある。新刊の登録はやや遅い。

NDL：新刊のデータが入ってくるのは遅いが、国内の書籍であれば、ほとんどヒットする。ただし、動作が遅いので、データがあってもエラーになることがある。利用にはクレジット表示が必要。営利目的の場合は、事前に申請が必要。

Amazon：事前に登録が必要。クレジット表示義務があり、情報源へのリンクを張る必要がある。アカウント情報を隠さないといけないので、今回使用するJavaScriptのようなプログラムでは利用が難しい。

このほかに有料ですが日外アソシエーツから提供されている「BOOKデータ」もあります。

Step2：ブックリストを作って見出しをつけよう

```
55
56  <script>
57
58  var isbn_list = get_isbn_list()
59  //var isbn_list = ["9784408451749","9784794967213","9784818823556"]
60
61  if(isbn_list.length > 0){
62    $("#form").hide();
63    search(isbn_list)
64  }
65
```

今までは、入力欄からISBNを入力していましたが、ブックリストを作る用にファイルにISBNを指定しておきます。こうすると、このページを見た人は全員、同じブックリストを見ることが出来ます。

booklist.htmlで、以下のように記述されているものを入力例として参考にしながらISBN

を入力してください。
　入力が終わったら、行頭の[//]スラッシュ2つを消します。

例　**//var isbn_list = [“9784408451749”,”9784794967213”,”9784818823556”];**
　　↓
　var isbn_list = [“9784408451749”,”9784794967213”,”9784818823556”];

　また、その上にある以下の行頭に[//]スラッシュ２つを追記してください。
　//var isbn_list = get_isbn_list()

＊あんまり考えすぎないこと
　簡単にテーマが決まれば、それに沿ってISBNを調べてもいいですし、思いつかないなら、本を適当に選んでもでもかまいません。実際のサービスとしても「本日のオススメ：ブックポストに返却された本から」などでまったく構わないと思います。

　テーマが決まったら、見出しを変更します。

　任意の<h1>から</h1>で囲まれた部分（<h1>は消さない）を書き換えます。
　<h1>はそのページの主題を示すタグになります。

　次に簡単な説明をつけましょう。
　から で囲まれた部分を書き換えます。
　改行などで行を増やしても大丈夫ですが、 は消さないようにします。
　見た目で改行が必要な場合には、改行する場所に **
** と入力します。
　
は、改行を意味するタグです。
　作業が終わったら、Google Chromeを再読み込みして、確認してみましょう。

Step3：デザインを変更しよう

　最後に準備体操でやったように文字の大きさや色を変えてみましょう。
　まず見出しの色を変えてみましょう。

```
  <title>API Workshop</title>

  <style>
  h1 {;} /*見出しのデザイン*/
  .comment {;} /*解説のデザイン*/
  .book_title{;}/*本のタイトル*/
  </style>

</head>
```

見出し(h1タグ)のデザインは、 h1{} のカッコの中に記述することができます。
h1{font-size:150%; color:green;}
とすると、文字サイズが普通の文字の150%の大きさで、通常の見出しより小さくなり、
緑色で表示されるはずです。
デザインを指定するときは、
プロパティ（font-size やcolorなど）の後に：（コロン）
値（150%や、greenなど）の後に；（セミコロン）
を入力する決まりになっているので、忘れないようにしてください。

同じように本のタイトルの部分も修正してみましょう。
本のタイトルは、 .book_title{} のカッコ内に記述することができます。
.book_title{ font-size:120%; color:#c33;}
とすると、タイトル部分が少し大きく、赤い色になります。

OPAC+を作ってみよう

最初の講義で事例として紹介した、中野区などで公開されているOPAC+を参考にしながら、OPAC+を作ってみます。
まずは利用者への公開は考えず、自分が仕事で便利な組み合わせは、何だろうか、ということを考えてサイトを組み合わせてみてください。
まずは、1コマ目と同じように各種テキストエディタをChromeで、［ex_05］フォルダにあるopacplus.htmlを開いてください。

Step1：まずは動作を確認してみよう
OPAC+は複数のサイトを簡単に検索できるようにする仕組みです。
まずはChrome上でその動作を確認してみましょう。
表示された検索窓に任意の検索語を入力します。

画面では「外来生物」で検索しています。
ウィンドウの大きさによって、検索対象を指定するタブは画面上部に表示されます。
また（別タブ）と表示されているものについては、フレーム（このようにページ内に別のページを表示する機能）で表示することが出来ないため、別タブで開くようになっています。
「東京都立」「野田市立」をそれぞれ選択して挙動を確認しましょう。（「カーリルローカル」「Amazon」は現在、動きません。動かないことも確認しましょう）。

Step2：カーリルの検索対象の修正

1コマ目と同じく、カーリルのシステムIDで検索対象を指定します。

（システムIDを忘れた方は、「**■カーリルのシステムIDの調べ方**」で）

```
<script>

var sites = [
  {
    "name": "カーリル",
    "target": "get",
    "url": "https://calil.jp/search?q=OPACWORD&sysid=Chiba_Noda"
  },
  {
    "name": "東京都立",
    "target": "tab",
    "url":
    "https://catalog.library.metro.tokyo.jp/winj/opac/search-standard.do?
    ummy="
  },
```

各種テキストエディタで開いたopacplus.htmlの、Chiba_Nodaと書かれている場所を自館のIDに置き換えて、上書き保存してください。

Chromeで再読み込みした後、もう一度、検索すると、今度は指定の図書館で検索することが出来るようになっているはずです。

Step3：カーリルローカルへリンクしよう

手順1）まず、カーリルのトップページからカーリルローカル（地域資料）に移動し、自分の所属する都道府県を選んでください。

手順2）以下の画面になったら、検索語に半角大文字でOPACWORDと入力し検索します。

手順3）検索結果が表示されたら、アドレスバーを全て選択してコピーしましょう。
（アドレスバーにカーソルをあわせて、Ctrl+A、Ctrl+C）
手順4）コピーが終わったら、各種テキストエディタで開いたopacplus.htmlの

{
"name": "カーリルローカル",
"target": "get",
"url": ""
},

の最後の２つのクォーテーションの間に、コピーした内容を貼り付けます。
東京都の場合は、以下のようになります。

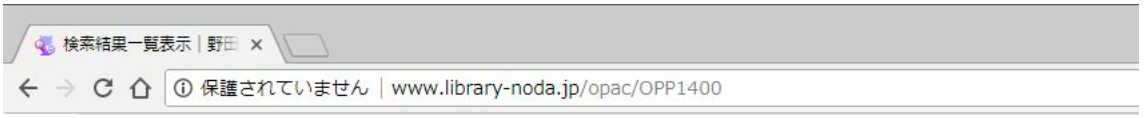
{
"name": "カーリルローカル",
"target": "get",
"url": "https://calil.jp/local/search?csid=tokyo&q=OPACWORD"
},

終わったら、上書き保存して、Chromeをリロードします。
今度は、「カーリルローカル」のタブをクリックすると、フレーム内にカーリルローカルの検索結果が表示されると思います。

Webサービスでのデータの送信について
例えばWeb-OPACの検索結果一覧や、詳細画面にリンクを貼りたい場合には、単に、そのアドレスだけでなく、必要なデータを送信する必要があります。
https://calil.jp/search?q=OPACWORD&sysid=Chiba_Nodaの場合、？はここからはアドレスではなく、データですよ、という区切り記号です。
つまり、https://calil.jp/search　までがアドレスで、その後のq=OPACWORD&sysid=Chiba_Noda　がデータです。
また＆は、ここからは別のデータですよ、という区切り記号です。
つまり、q=OPACWORDと、sysid=Chiba_Nodaの２つのデータが、https://calil.jp/searchのアドレスに送られている、ということになります。
データの名前（パラメータ）はqとsysid、値がOPACWORDと、Chiba_Nodaです。
STEP2の作業で、なんとなく分かるかもしれませんが、sysidは図書館を指定するためのパラメータで、図書館IDを指定することで、図書館の検索結果を取得できます。
qは、実は検索語を指定するパラメータです。
opacplus.htmlでは、[OPACWORD]という文字を、入力された検索語に置換する、という方法で、各サイトの検索結果を取得しています。
したがって、カーリルの場合、
https://calil.jp/search?q=「検索語」&sysid=「図書館ID」
で、指定した図書館の検索結果を表示させることが出来ます。
カーリルは簡単ですが、図書館のWeb-OPACも同じ仕組みで動いています。アドレスとパラメータが分かれば、多くの場合、検索結果や詳細画面を表示させることが出来ます。
ちなみに、検索などをした際、アドレスバーに送信データが表示される方法をget、表示されない方法をpostと言います。

get方式　＊送信するデータが確認できます。

post方式　＊送信するデータが隠れるようになっています。



Step4：Amazonへのリンクを作ろう
　同じようにAmazonでOPACWORDで検索したアドレスバーの内容を
　{
　"name": "Amazon",
　"target": "tab",
　"url": ""
　}
の部分に入力してみましょう。

Amazonが別タブで開くわけ
　試しに、と書かれている部分の、**tab**を、カーリルローカルと同じように**get**に変えてみると、同じタブで開くようになりますが、真っ白い画面が表示されるだけになっています。
　東京都立図書館などもそうですが、セキュリティの設定として、他のサイトからフレームで開くことが出来ないサイトがあります。今回のAmazonやGoogleなどもそうです。
　このように、他のサイトからの操作に対してかかる制限を、Same-Originポリシーと言います。

Step5：以下のサイトから2つ以上選んでリンクを作ろう
○Naxos
○国立国会図書館デジタルコレクション
○レファレンス共同データベース
○CiNii Articles
○リサーチナビ

手順1）まずは余分な野田市立図書館を消す。
　今回使っているjavascriptというプログラムでは、/*［テキスト］*/のように括ると、コメント、といって記述を無視して処理しなくなります。

```
/*
{
  "name": "野田市立",
  "target": "get",
  "url": "http://www.library-noda.jp/opac/..."
},
*/
```

上書き保存＆リロードして、「野田市立」のタブが消えていることを確認してください。

手順2）カーリルの要素をコピーして新しいタブのための行を作る。
　カーリルの以下の要素をコピーして、直後に貼り付けてください。

```
{
  "name": "カーリル",
  "target": "get",
  "url": "https://calil.jp/search?q=OPACWORD&sysid=Chiba_Noda"
```

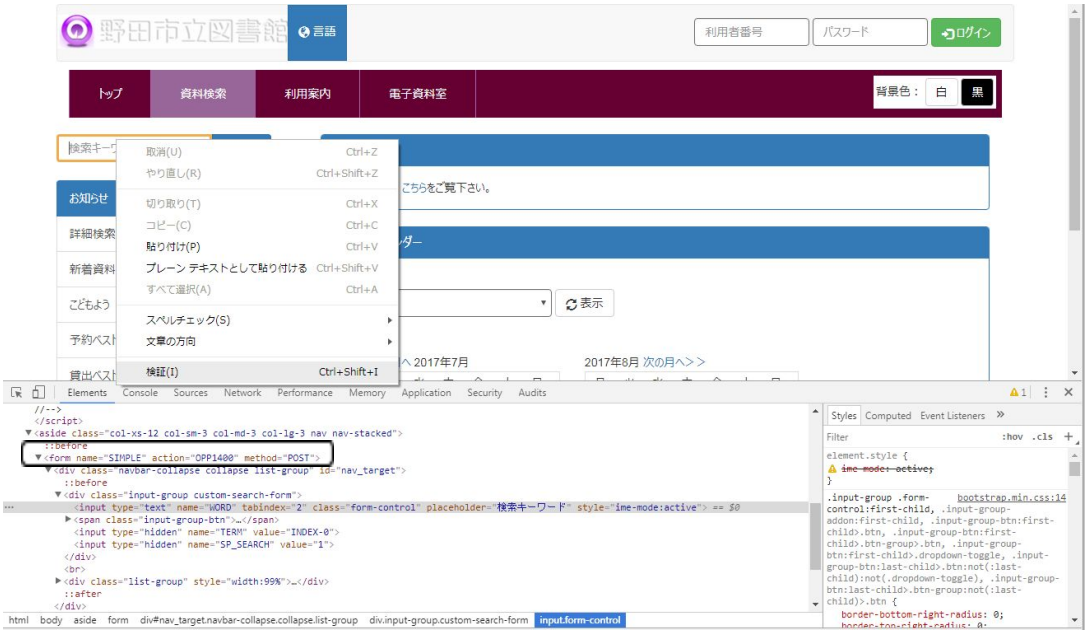
},
’name’の値の部分は、表示されるサイト名になります。
あまり長いと不格好なので、各サイト名は、適当に縮めて記入してください。
‘url’の値の部分が、サービスのアドレスと送信データです。

手順3）アドレスと送信データを記述する。
　さきほどと同じように、加えたいサイトへ移動した後、OPACWORD（半角大文字）で検索し、アドレスバーの内容をコピー＆ペーストしてください。
　入力が終わったら、上書き保存＆リロードして、実際に動くかどうか確認してみてください。

○**変更しなかったものについて**
　今回、"target":"get"と書かれている部分は変更しませんでした。
　"target":"get"の部分は、前述のとおり、フレーム内で開けるか、開けないかによって、指定します。今回は、どのサイトも'get’のままで大丈夫です。

エクストラステージ：任意のサイトを加えてみる

　これまでやってきたとおり、サービスのアドレスと送信データさえわかれば、多くの場合opacplusで使うことが出来ます。
　データの送信方式がgetの場合は、アドレスバーから送信データを確認できますが、そうでない場合には、準備運動でやった「ページの書き換え」が使えるかどうか、試してみましょう。
例：野田市立図書館の場合：



手順1）postをgetに書き替え
　検索窓で右クリック⇒検証（I）を選びます（ディベロッパーモードになります）
　検索窓が含まれる<form>タグ(だいたいちょっと上の行にある)のmethod=”POST“をmethod=”get“に書き換えます。

手順2）検索を実行する。
　いったん、検索が動くか確認するため、OPACWORDではなく、検査結果が表示される単語で検索します。

手順3）検索語をOPACWORDに変えて、アドレスと送信データを記述する。
　アドレスバーにデータ部分が表示されるようになります。入力した検索語もアドレスバーに表示されているはずなので、そこをOPACWORDに変更し、コピー＆ペーストします。
　その後、サイト名を変更したら、上書き＆リロードして下さい。

上手く動作しない場合
　そもそもこの方法では動かせないサービスもありますが、以下を確認してください。

1）フレームで開くことが出来ない
　この場合は、前述のとおり、3つ目の要素を変更します。getで検索していたならtabに変更して別タブで開くようにしてください。

それでも動かない場合は、作動できない可能性が高いです。