第 2 回

JavaScript入門

第1回でHTMLに入門しました。

さらにJavaScriptを使ってWebアプリを作ります。

見るだけの「ページ」から、動く「アプリ」へ！

フィジカルコンピューティング演習
愛知淑徳大学 人間情報学部　　國分 三輝

✈ 2.0 JavaScript 入門

第 1 回で HTML を使って Web ページを作りました。これだけでも見るだけの（動かない）コンテンツは作ることができますが…

HTML に加えて、JavaScript というプログラミング言語※2 を組み合わせると、ページがダイナミックに変化する Web アプリを作ることができます。近ごろでは、JavaScript で PC やスマホのパワフルな機能をフルに使えます。

簡単なアプリを作りながら、JavaScript によるプログラミングに入門しましょう。

2.1 JavaScript で Web アプリ

➔ 3 ページ

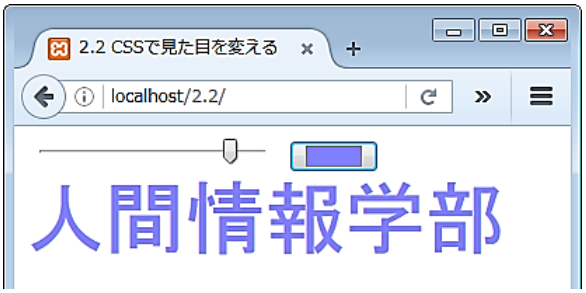
html ファイルに JavaScript を書くと、Web アプリを作れます。HTML の部品をプログラムで取得したり、操作したり、イベント処理、条件分岐など、JavaScript の基礎を紹介します。ブラウザを使ったデバッグ方法も紹介します。

2.2 CSS で見た目を変える

➔ 9 ページ

ページの見た目（部品の大きさ、色、フォントなどの）は、HTML や JavaScript とは異なる CSS という言語で設定します。CSS で設定した見た目を JavaScript から変更する方法を紹介します。

（CSS はこの授業ではあまり用いません）



2.3 時計アプリ

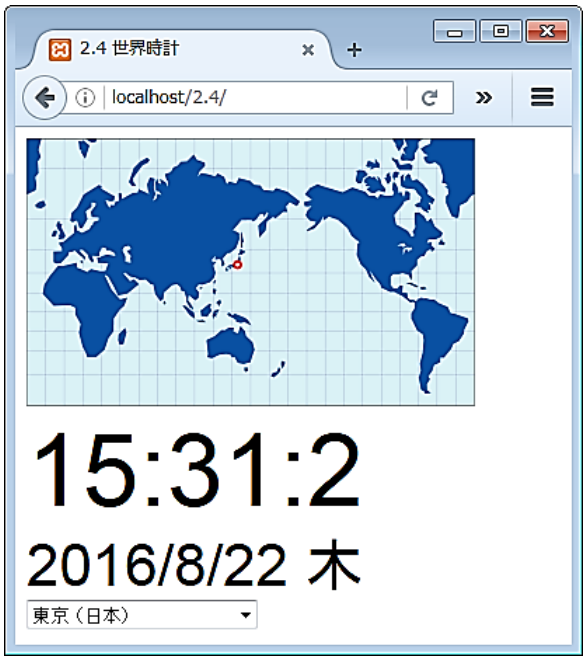
➔ 13 ページ

少し進んだサンプルとして、時計アプリを作ります。タイマー機能を使って一定間隔で時刻を調べてページを更新する方法を紹介します。これはアニメーションなどにも使えます。配列変数の使い方も紹介します。

2.4　世界時計

➔ 17 ページ

時計アプリを改造して世界時計にします。リストから世界の都市を選ぶと、その都市の日時と、都市の位置を示した地図が表示されます。



本日の課題

➔ 20 ページ

✈ 2.1 JavaScript で Web アプリ

	PC	And	iOS
Firefox	○	○	○
Chrome	○	○	○

html ファイルに JavaScript を書くと、Web アプリを作れます。HTML の部品をプログラムで取得したり、操作したり、イベント処理、条件分岐など、JavaScript の基礎を紹介します。ブラウザを使ったデバッグ方法も紹介します。

Input JavaScript を入力

- **C:￥xampp￥htdocs￥2.1** という階層を作り、その中に、1.2 で作った index.html をコピーします。
- この index.html を Brackets で開き、以下のコードを入力しましょう。網掛けが新しい部分です。

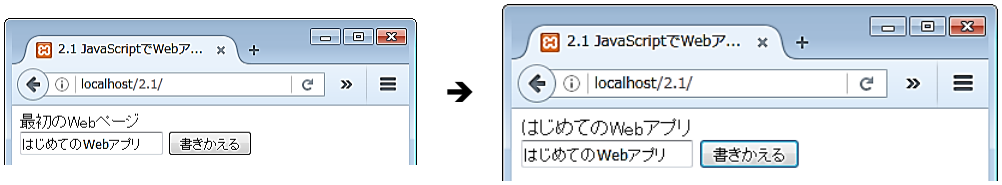
※ ①や②などの丸数字は打ち込まないこと

C:￥xampp￥htdocs￥**2.1**￥ **index.html**

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset='utf-8'>
        <meta name='viewport' content='width=device-width,initial-scale=1'>
        <title>2.1 JavaScript で Web アプリ</title>
    </head>
    <body>
        <div id='msg'>最初の Web ページ</div>
        <!-- テキストボックスとボタンを配置 -->
        <input type='text' value='' id='txt'>
        <input type='button' value='書きかえる' id='btn'>

        <script>                                                    ①
            // 変数 btn を宣言し、id が btn の要素<input>を入れる  ②
            const btn = document.getElementById('btn');             ③
            // イベント設定：btn に click が起きたら関数 changeMsg を実行
            btn.addEventListener('click', changeMsg);               ④

            // msg を書きかえる関数 changeMsg の定義
            function changeMsg() {                                  ⑤
                const msg = document.getElementById('msg');         ⑥
                const txt = document.getElementById('txt');         ⑦
                // msg の中の文字列を txt の value 属性の値にする
                msg.innerHTML = txt.value;                          ⑧
                }
        </script>                                                   ①
    </body>
</html>
```

Check 動作確認

- Web サーバをスタートさせ（第 1 回 P.5 参照）Firefox に以下のように入力して Enter
  - 例： **10.11.123.45/2.1/**
  - テキストボックスに何か文字を入力してから 書きかえる ボタンを押してみましょう。
  - 「最初の Web ページ」の部分が、入力した文字列に書き換わったら、成功です！



  - うまく動かない場合は、index.html をよーく見返して、間違いを修正しましょう。
  - スマホでも試してみましょう。

- ユーザが入力した文字列とユーザが押したボタンによって Web ページが変化しました。この時点でこの index.html は、見るだけの Web ページからインタラクティブな Web アプリになりました！

Learn 解　説

① `<script> ～ </script>`

<script>タグで囲んだ内側に JavaScript のプログラムを書きます。<script>タグは、<body>タグの内側に、しかも終了タグ</body>の直前に書くのが一般的です。

② `// 変数 btn を宣言し～` … コメント行

JavaScript のコメントは `//`（スラッシュ 2 個）の後に書きます。積極的にコメントを書いておきましょう。HTML の場合の <!-- ～ --> と異なっていてややこしいので注意しましょう。

③ `const btn = document.getElementById('btn');`

初めて出ましたね。これが JavaScript の命令です。JavaScript で HTML の部品を操作する場合、この命令のパターンはとても頻繁に使います。

まず `const btn` という部分で、btn という名前の変数を定義しています。

次の `document.getElementById( )` は、「html ドキュメントの中から（）内の id の部品を取ってこい」という意味です。今回の場合、btn という id を付けた部品は 書きかえる ボタンの <input>要素です。このボタンを取ってこい、という意味です。

これら二つの命令を `=` でつなぐことで「変数 btn に、btn という id の部品を入れよ」という意味になります。この後プログラム中で「btn」と書くと、書きかえる ボタンを指すことになります。

行末のセミコロン `;` も重要です。JavaScript では命令の終わりに必ずセミコロンを付けます。

④ `btn.addEventListener('click', changeMsg);`

`xxx.addEventListener(event, function)` は、部品に対してイベント（何か起きたら、何か処理をさせる機能）を設定します。今回の場合、btn [書きかえる] に対して、「`click` というイベントが起きたら changeMsg という処理（⑤）を実行する」機能を設定しています。

click 以外にも様々なイベントを設定できます（例えば以下）。部品によっても設定できるイベントが異なります。今後少しずつ覚えていきましょう。

イベント名	意味
click	マウスがクリックされた時
dblclick	マウスがダブルクリックされた時
mousedown	マウスのボタンが押された時
mouseup	マウスのボタンが離された時
mousemove	マウスが動いた時
mouseover	部品にマウスポインタが乗った時

※ 他にも様々なイベントがあります。必要に応じて調べましょう（コラム③参照）。

⑤ `function changeMsg() { … }`

`function` キーワードで関数を定義しています。関数とは「いくつかの命令をまとめたもの」です。function の後に好きな関数の名前(今回の場合「changeMsg()」)を書き、さらに中カッコ `{ }` の中に命令を書きます。

今回は、changeMsg 関数の中に、⑥～⑧の三つの命令が書かれています。④のイベントが発生した時（ボタンがクリックされた時）に、これらの命令が実行されます。

⑥ `const msg = document.getElementById('msg');`

③と同じパターンです。id を msg とした 要素（つまり、「初めての Web ページ」と書いた部分）を、変数 msg に入れています。

⑦ `const txt = document.getElementById('txt');`

③⑥と同じパターンです。id を txt とした<input>要素（つまり、テキストボックス）を、変数 txt に入れています。

⑧ `msg.innerHTML = txt.value;`

`innerHTML` は「要素の中身」という意味です。この場合、msg（つまり、 要素）の中身は「初めての Web ページ」という文字列です。ここでは、その文字列に対して、txt（つまり、テキストボックス）の value（値）を代入しています。まさにここで、「初めての Web ページ」をテキストボックスの文字列で書きかえています。

Input JavaScript を追加

- 上の例だと、テキストボックスが空の状態でボタンを押すとページの文字列が消えてしまいます。これを、テキストボックスが空の時は警告ダイアログを出して文字列を書きかえず、空でなければ書きかえる、というように、条件に応じて処理を変えるように changeMsg 関数を改造します。

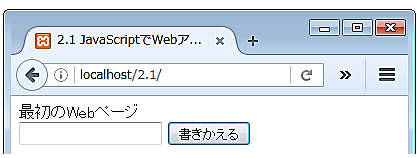
※ ①や②などの丸数字は打ち込まないこと

C:￥xampp￥htdocs￥2.1￥ **index.html**

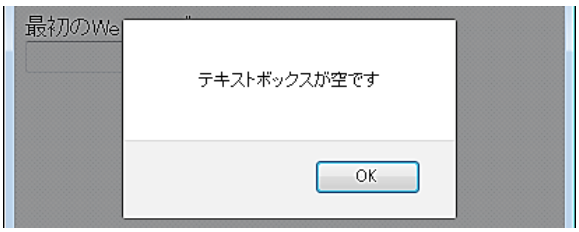
```
<!DOCTYPE html>
  :
（略）
  :
        <script>
            // 変数 btn を宣言し、id が btn の要素<input>を入れる
            const btn = document.getElementById('btn');
            // イベント設定：btn に click が起きたら関数 changeMsg を実行
            btn.addEventListener('click', changeMsg);

            // msg を書きかえる関数 changeMsg の定義
            function changeMsg() {
                const msg = document.getElementById('msg');
                const txt = document.getElementById('txt');
                if (txt.value == '') {  // txt の value 属性が空なら      ①
                    // 警告ダイアログを表示
                    window.alert('テキストボックスが空です');          ②
                } else {                // 空でなければ                  ③
                    // msg の中の文字列を txt の value 属性の値にする
                    msg.innerHTML = txt.value;
                }
            }
        </script>
（以下略）
```

Check 動作確認

- Firefox でチェック
  - テキストボックスが空の状態でボタンを押すと、警告ダイアログが出ましたか？
  - テキストボックスに何か文字を入力してボタンを押すと、前回同様に文字列が書き換わりましたか？



➔



Learn 解　説

①　`if (txt.value == '') { … }`

条件分岐の書き方です。( ) 内の条件式が真(true)の時、中カッコ { } 内の処理を実行します。この場合、txt.value (テキストボックスの値) が空の時に中カッコ { } 内を実行します。

== は、左右の値が等しいかどうかを調べる比較演算子で、等しいなら真(true)、等しくなければ偽(false)を返します。比較演算子には以下のような種類があります。

比較演算子	比較内容	算術記号では
==	等しい	=
!=	等しくない	≠
>	右の値より大きい	>
>=	右の値以上	≧
<	右の値より小さい	<
<=	右の値以下	≦
===	厳密に等しい	対応なし
!==	厳密には等しくない	対応なし

②　`window.alert('テキストボックスが空です');`

警告ダイアログを出す命令です。( ) 内の文字列をダイアログに表示します。

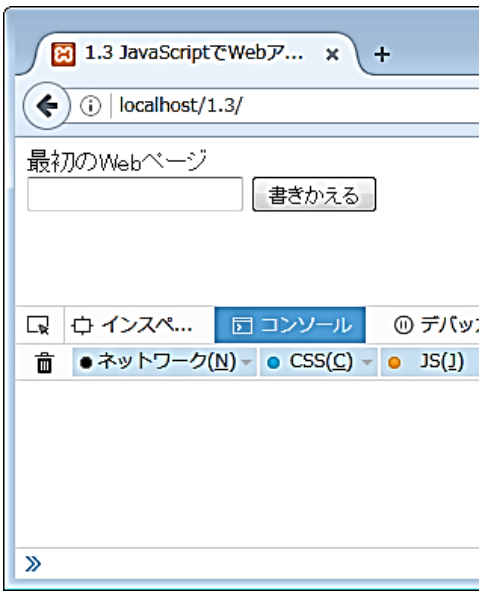
ダイアログには他にも、OK ボタンとキャンセルボタンが付いた window.confirm()や、文字列を入力できるテキストボックスが付いた window.prompt()などがあります。

③　`if (txt.value == '') { … } else { … }`

else { } は、if の条件式が偽(false)の時に中カッコ { } 内の処理を実行します。if()以降、ここまでワンセットです。

Check HTML・JavaScript をブラウザでデバッグ

- コードの打ち間違いが原因でうまく動かない時、普通のプログラミングの場合、開発ソフトが間違いを見つけてくれます (デバッグ)。HTML や JavaScript の場合ブラウザを使ってデバッグします。
- 設定 ≡ メニュー → 開発ツール → 開発ツールを表示 (Ctrl+Shift+I) を選ぶとコンソール画面が出ます。ここに表示されたエラーの内容と行番号をもとに、その周辺を重点的にチェックしてコードを直していきます。また、プログラムからこのコンソールに文字列を出力できます。処理が思った通りに進んでいるかどうかのチェックにも使えます。



Input デバッグ用の JavaScript を追加

- **C:￥xampp￥htdocs￥2.1** の index.html に、Console に文字列を出力するコードを追加します。
- changeMsg 関数の冒頭にこのコードを追加することで、ボタンの click イベントできちんと chengeMsg 関数が呼び出されているかを確認することができます。

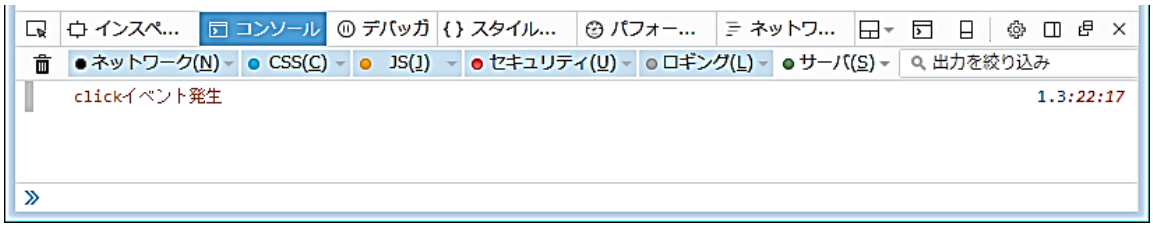
※ ①や②などの丸数字は打ち込まないこと

C:￥xampp￥htdocs￥2.1￥ index.html

```
（略）
            // msg を書きかえる関数 changeMsg の定義
            function changeMsg() {
                // click イベントが発生したことをデバッグ画面に表示
                console.log('click イベント発生');                    ①
                const msg = document.getElementById('msg');
                const txt = document.getElementById('txt');
                if (txt.value == '') {  // txt の value 属性が空なら
                    // 警告ダイアログを表示
                    window.alert('テキストボックスが空です');
                } else {                // 空でなければ
                    // msg の中の文字列を txt の value 属性の値にする
                    msg.innerHTML = txt.value;
                }
            }
（以下略）
```

Check 動作確認

- ボタンをクリックすると、コンソールに「click イベント発生」と表示されましたか？



Learn 解　説

① console.log('click イベント発生');

ブラウザの Console に文字列を出力する命令です。( ) 内には、文字列だけでなく、計算式も入れられます。計算式を入れると、その計算結果が出力されます。

✈ 2.2 CSS で見た目を変える

	PC	And	iOS
Firefox	○	○	○
Chrome	○	○	○

ページの見た目（部品の大きさ、色、フォントなどの）は、HTML や JavaScript とは異なる CSS という言語で設定します。CSS で設定した見た目を JavaScript から変更する方法を紹介します。

Input HTML を入力

- **C:￥xampp￥htdocs￥2.2** の中に、1.1 で作った index.html（テンプレート）をコピーします。
- この index.html を Brackets で開き、以下のコードを入力しましょう。

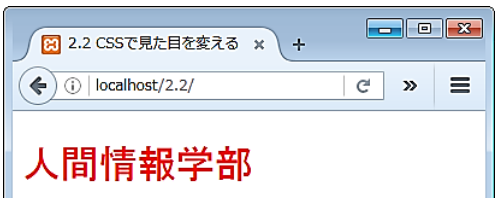
※ ①や②などの丸数字は打ち込まないこと

C:￥xampp￥htdocs￥2.2￥ **index.html**

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset='utf-8'>
        <meta name='viewport' content='width=device-width,initial-scale=1'>
        <title>2.2 CSS で見た目を変える</title>
        <style>                                          ①
            /* id が msg の要素についてスタイルを設定 */  ②
            #msg {                                       ③
                font-family: arial,sans-serif;           ④
                font-size: 32px;                         ⑤
                color: #FF0000;                          ⑥
            }
        </style>                                         ①
    </head>
    <body>
        <div id='msg'>人間情報学部</div>
    </body>
</html>
```

Check 動作確認

- Firefox に、1.1 でメモしたアドレス（★）に続けて「/2.2/」と入力して Enter
  - 例： **10.11.123.45/2.2/**
  - 大きめのサイズで、赤色で「人間情報学部」と表示されていれば、成功です。



Learn 解　説

① **<style> ～ </style>**

ページの見た目（スタイル）は、<style>タグの内側で、CSS[※1] という言語を使って設定します。<style>タグは<head>の内側に書きます。

※1 Cascading Style Sheets の略。

② **/* ～ */　… コメント行**

CSS では /* ～ */ の間がコメントです。HTML とも JavaScript とも違い、ややこしいですが…

③ **#msg { … }**

#msg は「id が msg の要素について」という意味です（セレクタと呼びます）。id が msg の要素について中カッコ { } 内のスタイルを設定せよ、という命令です。つまり、⑦の「人間情報学部」という 要素に対してスタイルを設定しています。

④ **font-family: arial,sans-serif;**

フォントの指定です。ここでは、Windows に必ず入っている arial と、Web の標準フォントの sans-serif（普通のゴシック体）を指定しています。

行末のセミコロン（ ; ）を忘れないようにしましょう。

⑤ **font-size: 32px;**

フォントサイズの指定です。単位はピクセル（px）です。特にサイズを指定しない場合 16px になります。px 以外にも様々な単位があります。必要に応じて各自で調べてください。

⑥ **color: #FF0000;**

表示色の指定です。#CC99FF のような 6 桁の 16 進数の値で指定します。#の後の 2 桁が赤(R)、次の 2 桁が緑(G)、最後の 2 桁が青(B)の強さです。値を覚えるのは困難ですから、「Web カラー」などのキーワードでネット検索し、色見本から値をコピペしましょう。

CSS による Web ページの見た目の初期設定については、ここまでです。

これ以降、CSS で行ったスタイルの初期設定を JavaScript で動的に変える Web アプリに改造します。ここからが本題です。

Input HTML と JavaScript を追加

- **C:￥xampp￥htdocs￥2.2** の中の index.html に以下のようにコードを追加しましょう。

※ ①や②などの丸数字は打ち込まないこと

C:￥xampp￥htdocs￥2.2￥ **index.html**

```
（略）
    <body>
        <input type='range' min='8' max='64' step='1' value='32' id='sld'>      ①
        <input type='color' value='#FF0000' id='col'>                           ②
        <br>                                                                    ③
        <div id='msg'>人間情報学部</div>
        <script>
            const msg = document.getElementById('msg');                        ④

            const sld = document.getElementById('sld');                        ④
            // イベント設定：sld に change が起きたら関数 changeSize を実行
            sld.addEventListener('change', changeSize);                        ⑤
            // サイズを変える関数 changeSize の定義
            function changeSize() {                                            ⑥
                // msg に設定されている style の、fontSize を変更
                msg.style.fontSize = sld.value + 'px';                         ⑦
            }

            const col = document.getElementById('col');                        ④
            // イベント設定：col に change が起きたら関数 changeColor を実行
            col.addEventListener('change', changeColor);                       ⑤
            // 色を変える関数 changeColor の定義
            function changeColor() {                                           ⑥
                // msg に設定されている style の、color を変更
                msg.style.color = col.value;                                   ⑧
            }
        </script>
    </body>
（略）
```

Check 動作確認

- Firefox でチェックしましょう。
  - スライダーで文字サイズが、色選択ダイアログで色を選ぶと文字の色が、それぞれ変われば成功です！

Learn 解　説

① **`<input type='range' min='8' max='64' step='1' value='32' id='sld'>`**

<input>タグに type='range'を指定してスライダーを置きました（p.8 参照）。`min` はスライダーの最小値、`max` は最大値、`step` は 1 目盛りの増減幅、value は初期値です。

② **`<input type='color' value='#FF0000' id='col'>`**

<input>タグで type='color'を指定して、色入力欄（色選択ダイアログボックス）を置きました。（p.8 参照）。value は色の初期値です（CSS の場合と同じ指定方法、p.16 参照）。

③ **`<br>`**

「改行せよ」というタグです。ここでは、部品の配置を調整しているだけです。

④ **`const xxx = document.getElementById(id);`**

よく使うパターンでしたね（1.3 参照）。HTML の要素を変数に入れています。

⑤ **`xxx.addEventListener(event, function);`**

イベントの設定でしたね（1.3 参照）。今回は change というイベントを設定しています。これは「値が変化した時」というイベントです。つまりここでは、スライダーの値が変化した時と、色入力欄の値が変化した時、二つのイベントを設定しています。

⑥ **`function changeSize() { … }`** と **`function changeColor() { … }`**

関数の定義でしたね（1.3 参照）。⑤で設定した二つのイベントに応じた、二つの関数です。

⑦ **`msg.style.fontSize = sld.value + 'px';`**

ここがこの節の本題です。CSS で初期設定したスタイルを変更しています。

xxx.`style`.○○ のように、部品 xxx の style のうち、○○の値を変える、という意味です。ここでは `fontSize` を、ユーザがスライダーで設定した値にしています。CSS の font-size の指定と同様、'px'という単位の文字列も忘れず加えておきましょう。

⑧ **`msg.style.color = col.value;`**

⑦と同様、style のうち、color を、ユーザが色入力欄で指定した値にしています。

✈ 2.3 時計アプリ

	PC	And	iOS
Firefox	○	○	○
Chrome	○	○	○

少し進んだサンプルとして、時計アプリを作ります。タイマー機能を使って一定間隔で時刻を調べてページを更新する方法を紹介します。これはアニメーションなどにも使えます。配列変数の使い方も紹介します。

Input HTML と JavaScript を入力

- **C:￥xampp￥htdocs￥2.3** の中に、1.1 で作った index.html (テンプレート) をコピーします。
- この index.html を Brackets で開き、以下のコードを入力しましょう。長めですが頑張りましょう。

※ ①や②などの丸数字は打ち込まないこと

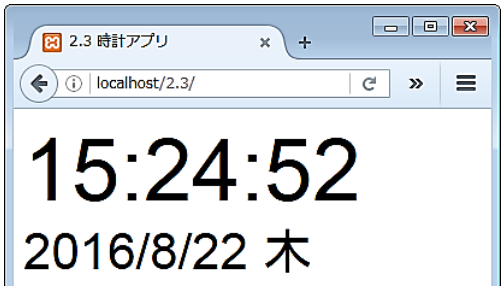
C:￥xampp￥htdocs￥**2.3**￥ **index.html**

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset='utf-8'>
        <meta name='viewport' content='width=device-width,initial-scale=1'>
        <title>2.3 時計アプリ</title>
        <style>                                                              ①
            #time {
                font-family: arial, sans-serif;
                font-size: 80px;
            }
            #date {
                font-family: arial, sans-serif;
                font-size: 48px;
            }
        </style>
    </head>
    <body>
        <!-- 時刻と日付を表示する div 要素 -->
        <div id='time'>--:--:--</div>                                        ②
        <div id='date'>----/--/-- ---</div>

        <script>
            // 時計を表示する関数 drawClock の定義
            function drawClock() {
                const now = new Date(); // 現在のシステム日時を取得して変数 now に入れる ③
                // 時刻を表示
                const hour = now.getHours();    // 時 ( Hours ) を取得して hour に入れる   ④
                const min = now.getMinutes();   // 分 ( Minutes ) を取得して min に入れる
                const sec = now.getSeconds();   // 秒 ( Seconds ) を取得して sec に入れる
                const time = document.getElementById('time');
                time.innerHTML = hour + ':' + min + ':' + sec;               ⑤
```

```
            // 日付を表示
            const year = now.getFullYear(); // 年（FullYear）を取得して year に
            const mon = now.getMonth();     // 月（Month）を取得して mon に
            const day = now.getDate();      // 日（Date）を取得して day に
            const dayNo = now.getDay();     // 曜日番号（Day、日 0～土 6）を dayNo に ⑥
            // 曜日名の配列を定義
            const dayArray = new Array('日', '月', '火', '水', '木', '金', '土');   ⑦
            const dayName = dayArray[dayNo];  // 日番号から曜日名を決める           ⑧
            const date = document.getElementById('date');
            date.innerHTML = year + '/' + mon + '/' + day + ' ' + dayName;
        }

        // 関数 drawClock を実行（最初に 1 回）
        drawClock();                                                              ⑨
    </script>
</body>
</html>
```

Check 動作確認

- Firefox に、1.1 でメモしたアドレス（★）に続けて「/2.3/」と入力して Enter
  - 例：**10.11.123.45/2.3/**
  - 現在時刻と日付が表示されれば成功です！　なお、この段階では時刻は動きません。



Learn 解　説

① **<style> ～ </style>**

時刻と日付の文字列に対するスタイルの設定です。

② ** **

時刻と日付を表示する二つの タグです。

③ **const now = new Date();**

日時を扱う場合 `new Date()` というオブジェクトを宣言します。これで、変数 now の中に現在日時の情報が一括で入ります。得られる日時は、Web サーバ（この授業の場合は開発している PC）の日時です。

④　`const hour = now.getHours();`　など

日時情報の中から、時、分、秒、年、月、日などを得る行です。月は 0～11 で得られます。

⑤　`time.innerHTML = hour + ':' + min + ':' + sec;`

id が time の 要素内に、時刻の文字列（時：分：秒）を表示しています。文字列変数同士を `+ 演算子`でつなげることができます。

⑥　`const dayNo = now.getDay();`

「曜日」については、日曜が 0～土曜が 6 の数値（`曜日番号`）で得られます。この数値をもとに、次の⑦と⑧で曜日の名前に変換しています。

⑦　`const dayArray = new Array('日', '月', '火', '水', '木', '金', '土');`

dayArray という名前の変数宣言ですが、`new Array( … )` とすることで、その変数は`配列`になります。（ ）内に配列の要素を設定します。この 1 行で、以下の 7 個の要素ができました。

```
dayArray[0] = '日'  dayArray[1] = '月'  dayArray[2] = '火'  dayArray[3] = '水'
dayArray[4] = '木'  dayArray[5] = '金'  dayArray[6] = '土'
```

⑧　`const dayName = dayArray[dayNo];`

⑦で作った曜日の配列の要素番号に、⑥で取得した曜日番号を入れて、曜日名を得ています。配列の要素は、変数名の後に`大カッコ [ ]` を付け、その中に要素番号を指定して取得します。

⑨　`drawClock();`

関数は定義しただけでは何も起こりません。今回の場合、`function drawClock() { … }` で定義した関数を、ここではじめて実行しています。`関数名();` と書くと関数を実行できます。

Input JavaScript を追加

- 時計としてちゃんと動くように改造しましょう。といっても、`1 文追加`するだけです。

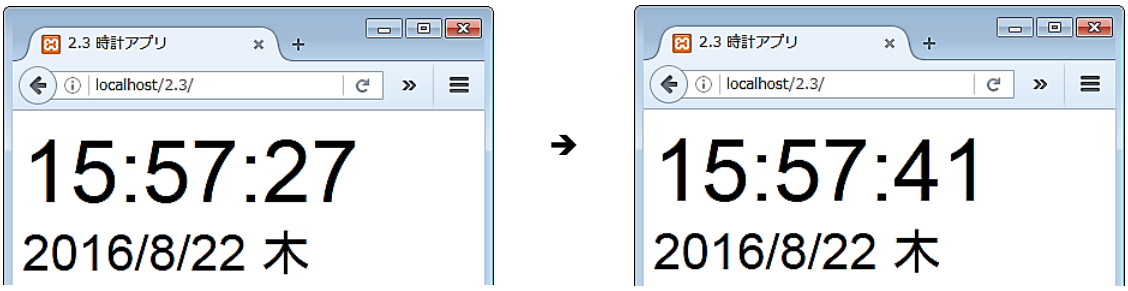
※ ①や②などの丸数字は打ち込まないこと

C:￥xampp￥htdocs￥**2.3**￥ **index.html**

```
（略）
            // 関数 drawClock を実行（最初に 1 回）
            drawClock();
            // 一定間隔で関数を繰り返して実行する setInterval を実行
            window.setInterval(drawClock, 1000);   //(関数, 実行間隔[ミリ秒])   ①
（略）
```

Check 動作確認

- Firefox でチェック。時刻が動いていれば成功です！



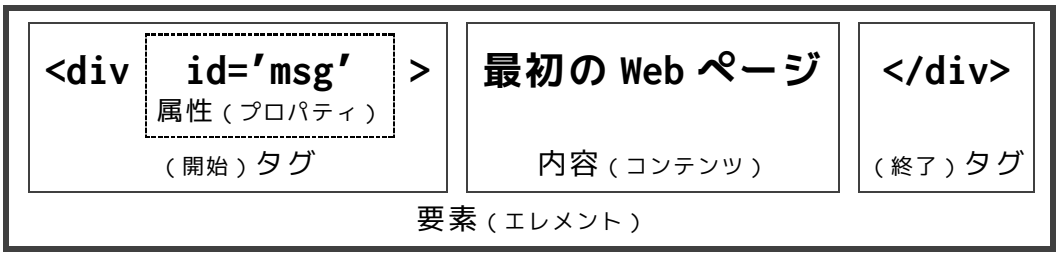
Learn 解　説

① **window.setInterval(drawClock, 1000);**

window.setInterval(関数, 実行間隔[ミリ秒]) は、一定間隔で繰り返して関数を実行します。一般的に「タイマー」と呼ばれます。今回の場合、1000 ミリ秒（つまり 1 秒）ごとに drawClock 関数を実行します。これによって、時刻の表示が毎秒更新されるようになりました。

Break コラム④ タグ、要素、属性、コンテンツ…

HTML では「タグ」「属性（プロパティ）」「内容（コンテンツ）」「要素（エレメント）」などの言葉を使います。整理しておきましょう。



- const x = document.getElementById('msg'); ➔ 「要素（エレメント）」が得られます。つまり…
  変数 x には「 最初の Web ページ 」が入ります。

- const y = x.id; ➔ 「属性（プロパティ）」が得られます。つまり…
  変数 y には「msg」が入ります。スライダーの例（p.11）の sld.value なども同じです。

- const z = x.innerHTML; ➔ 「内容（コンテンツ）」が得られます。つまり…
  変数 z には「最初の Web ページ」が入ります。

✈ 2.4 世界時計

	PC	And	iOS
Firefox	○	○	○
Chrome	○	○	○

時計アプリを改造して世界時計にします。リストから世界の都市を選ぶと、その都市の日時と、都市の位置を示した地図が表示されます。

Input HTML と JavaScript を追加

- **C:￥xampp￥htdocs￥2.4** の中に、2.3 で作った index.html（時計アプリ）をコピーします。
- キャンパススクエアから「PHC_02_素材.zip」をダウンロードし、「2.4 素材」フォルダの中にある 10 個の画像を 2.4 フォルダに入れます。
- index.html を Brackets で開き、以下のコードを入力しましょう。

※ ①や②などの丸数字は打ち込まないこと

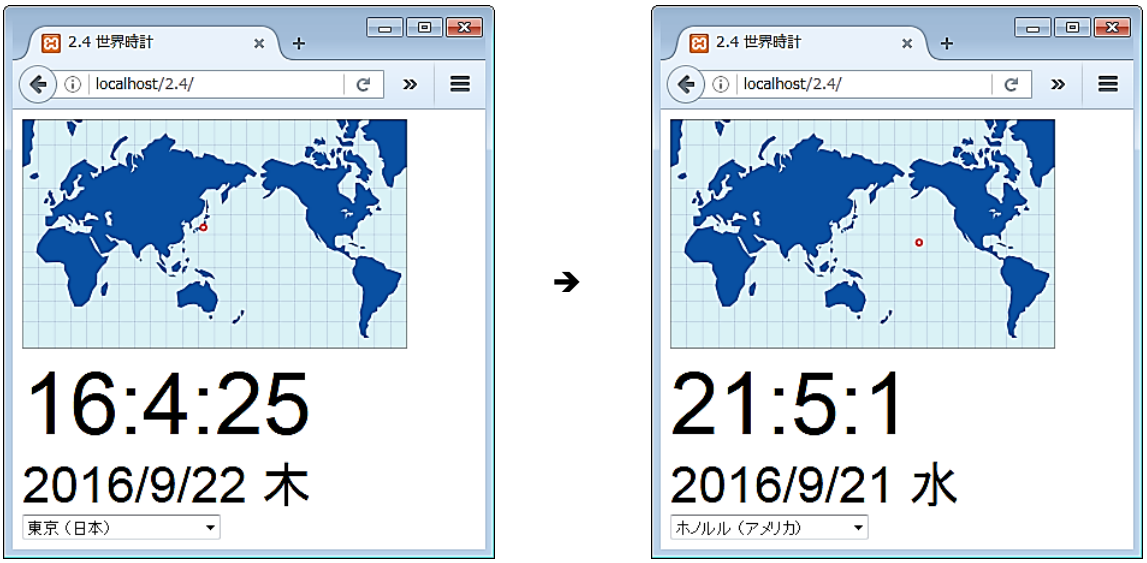
C:￥xampp￥htdocs￥**2.4**￥ **index.html**

```
（略）
  :
   <body>
       <!-- 世界地図を表示する img 要素 -->
       <img src='tokyo.png' id='map'>                                    ①
       <!-- 時刻と日付を表示する div 要素 -->
       <div id='time'>--:--:--</div>
       <div id='date'>----/--/-- ---</div>
       <!-- 都市を選択するプルダウンリスト -->
       <select id='city'>                                                ②
           <option value='-10'>ホノルル （アメリカ）</option>             ③
           <option value='-8'>ロサンゼルス （アメリカ）</option>
           <option value='-5'>ニューヨーク （アメリカ）</option>
           <option value='-3'>サンパウロ （ブラジル）</option>
           <option value='0'>ロンドン （イギリス）</option>
           <option value='3'>モスクワ （ロシア）</option>
           <option value='5.5'>ニューデリー （インド）</option>
           <option value='8'>北京 （中国）</option>
           <option value='9' selected>東京 （日本）</option>             ④
           <option value='10'>シドニー （オーストラリア）</option>
       </select>
       <script>
           // 時計を表示する関数 drawClock の定義
           function drawClock() {
               // 現在のシステム日時情報（Date）を取得して変数 now に入れる
               let now = new Date();                                     ⑤
               // UTC 時刻（協定世界時）を求める                          ⑥
               const offset = now.getTimezoneOffset(); // UTC 時刻との時差（分単位）
               const offsetMs = offset * 60 * 1000;    // 時差をミリ秒単位に変換
               const UTC = now.getTime() + offsetMs    // UTC 時刻（ミリ秒単位）
```

```
            // 選択された都市の時差を求める
            const city = document.getElementById('city');
            const cityNo = city.selectedIndex;      // 選択されている番号を調べる
            const cityOffset = city.options[cityNo].value      // その都市の時差を得る
            const cityOffsetMs = cityOffset * 60 * 60 * 1000;  // 時差をミリ秒に変換
            // UTC 時刻とその都市の時差から、その都市の現在時刻を求める
            const cityTime = UTC + cityOffsetMs;    // 選択された都市の時刻
            // 選択された都市の日時情報を取得して変数 now に入れる
            now = new Date(cityTime);                                          ⑦
            // 時刻を表示
            const hour = now.getHours();      // 時 ( Hours ) を取得して hour に入れる
                :
            ( 中略 )
                :
            date.innerHTML = year + '/' + mon + '/' + day + ' ' + dayName;
            // 地図の表示
            const map = document.getElementById('map');
            const mapFile = new Array('honolulu.png', 'losangeles.png',        ⑧
                    'newyork.png', 'saopaulo.png', 'london.png',
                    'moscow.png', 'newdelhi.png', 'beijing.png',
                    'tokyo.png', 'sydney.png'); // 画像ファイル名の配列を定義
            map.src = mapFile[cityNo];   // map の src に都市の画像ファイルを設定
        }
        // 関数 drawClock を実行
        drawClock();
( 以下略 )
```

Check 動作確認

- Firefox でチェック。都市を選ぶと、地図の画像と日時が変わりましたか？



Learn 解　説

① **<img src='tokyo.png' id='map'>**

ページに画像を置く <img>タグです。src 属性に画像ファイル名を指定します（jpg、gif、png）。今回は東京の位置を記した画像を初期値にしました。<img>は終了タグを使いません。

② **<select> 〜 </select>**

ページにプルダウンリストを置く <select>タグです。③の<option>タグとともに使います。

③ **<option value='-10'>ホノルル （アメリカ）</option>**

プルダウンリストの選択肢は<option>タグで追加します。今回は 10 個の選択肢（都市）を用意しました。value には時差を入れました（協定世界時（UTC）と各都市の時差（時間単位））。

④ **let now = new Date();**

ここまで変数の定義には const を使いましたが、ここでは let を使っています。const 変数は中身を変更（再代入）できませんが、let 変数は再代入できます。2.3 では now に再代入がないので const で問題ありませんでしたが、ここでは⑦で now に再代入があるため let を使いました。

⑤ **<option value='9' selected>東京 （日本）</option>**

<option>タグ内に selected という属性を書くと、そこが選択された状態で始まります。

⑥ **選択された都市の日時を求める処理たち** … **ややこしいので理解できなくても構いません**

A)　Date() で得たシステム日時 now（日本時間）をもとに、協定世界時（UTC）を逆算

a)　now.getTimezoneOffset() でシステム時刻と UTC との時差を得る（分単位で得られる）

b)　分単位の時差をミリ秒単位に変換　（分 x 60[秒] x 1000[ミリ秒]）

c)　now.getTime() で得たシステム時刻（ミリ秒）に b)を足して UTC（ミリ秒）を得る

B)　UTC と選択された都市の時差を計算

a)　city.selectedIndex で、選択された都市の番号を得る（上から 0〜9 で返ってくる）

b)　city.options[cityNo].value で、選択された都市の時差（value、時間単位）を得る

c)　時間単位の時差をミリ秒単位に変換　（時間 x 60[分] x 60[秒] x 1000[ミリ秒]）

d)　A)で得ている UTC に c)で得た都市の時差を足して、その都市の時刻（ミリ秒）を得る

C)　now = new Date(cityTime) で、その都市の日時情報を now に入れる（⑥参照）

⑦ **now = new Date(cityTime);**

Date()は、（)内が空だとシステム日時が、（)内に任意の時刻（ミリ秒単位）を入れるとその時刻にあった日時が得られます。今回は⑤で求めた都市の時刻を（)内に入れました。

⑧ **const mapFile = new Array('honolulu.png', 'losangeles.png', 〜);**

各都市の画像ファイル名の配列です。（)内の画像ファイル名の順序は、<option>に書いた都市の順にします。

データを持ち帰る

- 本日作成したデータは全て USB メモリなどにコピーして持ち帰りましょう
  - C:￥xampp￥htdocs 内の以下のフォルダ（4 個）
  - 2.1　2.2　2.3　2.4

本日の課題

① 2.1、2.3 のコードを完成させる（PC で動作確認もする）

② readme.txt というテキストファイルを作り、以下を書く

- 学籍番号 と 氏名
- 所感　（考えたこと、感想、応用アイデアなど。字数不問だが数行は書いてほしい。）
- 質問　（無ければ不要）

【提出方法】

- ①のフォルダ 2 個と、②の「readme.txt」1 個、全てをまとめて zip 圧縮
- zip のファイル名は「学籍番号.zip」とする　（例：12345nhu.zip）
- CampusSquare のレポート「フィジカルコンピューティング 02」から提出

【提出締切】　10 月 13 日（木）　15:00　（遅れてしまった場合は担当教員に相談のこと）

コラム⑤ 広く公開するならレンタルサーバ

この授業では XAMPP（Apache）を使い、開発している PC を Web サーバにしてアプリをテストしますが、この方法だと LAN（Local Area Network）の中にしか公開できません。大学の PC なら学内だけ、自宅の PC なら自宅内だけです。演習や卒業制作ならこれでも十分ですが、良いアプリができたら、世界に広く公開したくなるかもしれません。

そんな場合は「レンタルサーバ」や「ホスティングサービス」の契約を検討しましょう。数多くのサービスがあり、一概に「○○がおすすめ」とは言えませんが、HTML と JavaScript のみの Web アプリなら、基本的にどこでも大丈夫です。無料で始められるものもあります。

ただ、この授業の後半では、Node.js というツールを使って、サーバ側のプログラムも書きます。Node.js に対応したサービスはまだ多くありません。サービス契約の際には、その他にも、自分の使いたい機能・仕様を満たしているか、じっくり検討しましょう。