Webプログラミング
第14回

前回まで

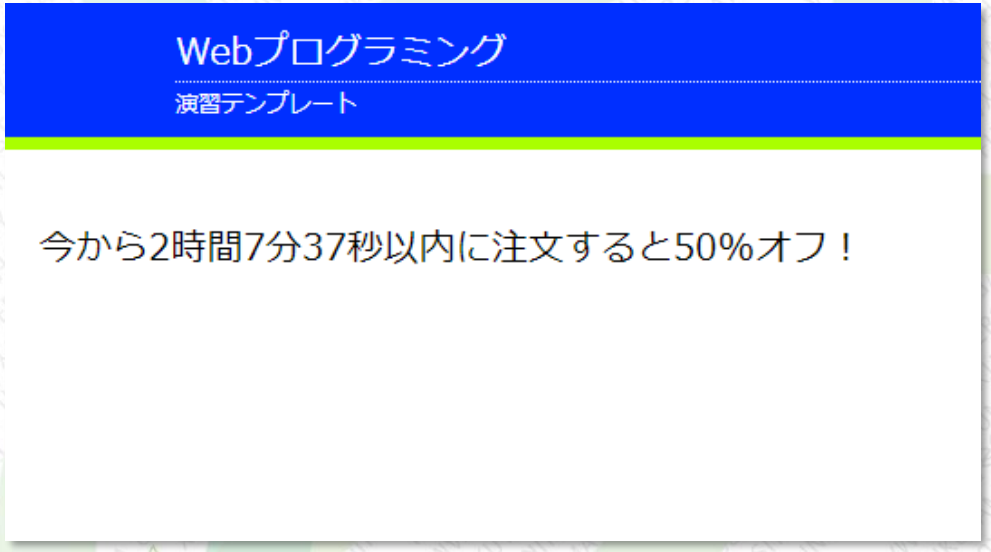
- 日付と時間の処理
  - 年月日と日付をわかりやすく表示する
  - Dateオブジェクト
  - 日数計算
- 四則演算以外の計算
  - 円周率を表示する
  - Mathオブジェクト

今回の内容

- カウントダウンタイマー
  - 残り時間の計算
  - １秒ごとに表示を更新する
- イメージビューワー
  - サムネイルをクリックすると、画像が切り替わる

カウントダウンタイマー

- Dateオブジェクトを使ってカウントダウンタイマー(１秒刻み)を作る
  - 日時の計算、タイマー機能を利用する



カウントダウンタイマー

- 残り時間を計算する関数(function)を作る、以下の処理をする
  - 関数名は countdown()
  - 未来の時刻(ゴール時刻)を引数で受け取る
  - ゴール時刻から現在時刻を引く
  - 計算結果を返す

カウントダウンタイマー

- テンプレートの「index.html」をコピーして、ファイル名を「index1401.html」に変更、下記を追記する
- テンプレートの「style.css」もコピーする

index1401.html

```
27          </footer>
28          <script>
29              'use strict';
30              function countdown(due) {
31                  const now = new Date();
32                  const rest = due.getTime() - now.getTime();
33                  const sec = Math.floor(rest / 1000) % 60;
34                  const min = Math.floor(rest / 1000 / 60) % 60;
35                  const hours = Math.floor(rest / 1000 / 60 / 60) % 24;
36                  const days = Math.floor(rest / 1000 / 60 / 60 / 24);
37                  const count = [days, hours, min, sec];
38                  return count;
39              }
40          </script>
41      </body>
```

カウントダウンタイマー

- function countdown() を呼び出して、コンソールで動作を確認する
- 現在時刻から、同日の23時間:59分:59秒までの結果を表示する
- 「index1401.html」に下記を追記する

index1401.html

```
37              const count = [days, hours, min, sec];
38              return count;
39          }
40          let goal = new Date();
41          goal.setHours(23);
42          goal.setMinutes(59);
43          goal.setSeconds(59);
44          console.log(countdown(goal));
45      </script>
```

今の時刻と、同日の23時59分59秒までの差が表示される

配列[0, 7, 24, 28]は [日, 時, 分, 秒] の順番に並んでいる

カウントダウンタイマー

- function countdown() を呼び出し、コンソールに表示
- 現在時刻から、同日の23時間:59分:59秒までの差
- 関数呼び出し側では以下のような流れになる

```
let goal = new Date();
```
… 現在の日時でオブジェクトを作成

```
goal.setHours(23);
goal.setMinutes(59);
goal.setSeconds(59);
```
… 時間に 23 を設定する
… 分に 59 を設定する
… 秒に 59 を設定する

} 変数 goal の時刻設定を 23:59:59 にする

```
console.log(countdown(goal));
```
… 関数呼び出し、戻り値を表示する

カウントダウンタイマー

- function countdown(due) の処理

```
console.log(countdown(goal));
```

← 呼び出し側

goal の内容が due に代入される(Dateオブジェクト)

```
function countdown(due) {
    const now = new Date();
    const rest = due.getTime() - now.g
```

```
const now = new Date();
```

… 現在の日時でオブジェクトを作成

```
const rest = due.getTime() - now.getTime();
```

… due のミリ秒から、定数 now のミリ秒を引いて、定数 rest に代入

<getTime()メソッド>

メソッド	説明
getTime(ミリ秒)	1970/1/1の0時からの時間をミリ秒で取得する

カウントダウンタイマー

- function countdown(due) の処理(続き)

restにミリ秒単位で数値が入っている場合、1000で割って60で割った余りが
0 ～ 59 の**秒**となる

(2)小数点があれば切り捨てる　(3)60で割った余り

```
const sec = Math.floor(rest / 1000) % 60;
```

(1)ミリ秒→秒に変換

分の計算は、1000で割ってさらに60で割ったものを、60で割った余り

```
const min = Math.floor(rest / 1000 / 60) % 60;
```

秒をさらに60で割る

時、**日**も同様に計算する

```
const hours = Math.floor(rest / 1000 / 60 / 60) % 24;
const days = Math.floor(rest / 1000 / 60 / 60 / 24);
```

カウントダウンタイマー

- function countdown(due) の処理(続き)

日の計算は以下のようになる

```
const days = Math.floor(rest / 1000 / 60 / 60 / 24);
```

(1)ミリ秒 → 秒

(2)秒 → 分

(3)分 → 時

(4)時 → 日

計算した 日、時、分、秒を配列 count に代入、それを戻り値として返す

```
const count = [days, hours, min, sec];
return count;
```

カウントダウンタイマー

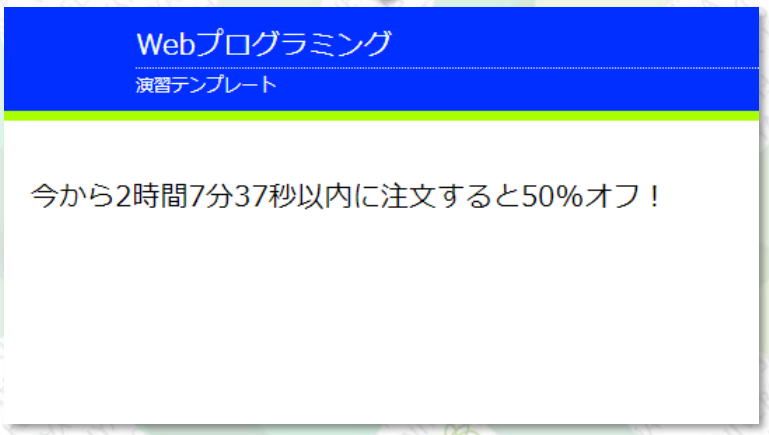
- 次にHTMLに表示する
- 「index1401.html」に下記を追記する

index1401.html

```
18          <section>
19              <p>今から<span id="timer"></span>以内に注文すると50%オフ！</p>
20          </section>
```

:

```
45          console.log(countdown(goal));
46          const counter = countdown(goal);
47          const time = `${counter[1]}時間${counter[2]}分${counter[3]}秒`;
48          document.getElementById('timer').textContent = time;
49      </script>
```



現在時刻から当日夜 23:59:59
までの時間が表示される

カウントダウンタイマー

- HTMLに表示する

関数呼び出し

```
const counter = countdown(goal);
```

```
function countdown(due) {
    const now = new Date();
    const rest = due.getTime() - now.getTim
                    :
    const count = [days, hours, min, sec];
    return count;
}
```

配列の変数を返す

counter は
配列変数となる

変数 counter は配列変数であり
counter[0]は 日
counter[1]は 時
counter[2]は 分
counter[3]は 秒
が入っている

```
const time = `${counter[1]}時間${counter[2]}分${counter[3]}秒`;
```

HTMLに表示

```
document.getElementById('timer').textContent = time;
```

カウントダウンタイマー

- 残り時間が１秒ごとに変化するようにする
- １秒ごとに以下の処理を繰り返す
  - 残り時間を再計算する
  - 残り時間を表すテキストを作成
  - HTMLに出力
  - 「index1401.html」で下記のように修正する

```
45            console.log(countdown(goal));
46            const counter = countdown(goal);
47            const time = `${counter[1]}時間${counter[2]}分${counter[3]}秒`;
48            document.getElementById('timer').textContent = time;
49        </script>
```

index1401.html

```
45        //console.log(countdown(goal));
46        function recalc() {
47            const counter = countdown(goal);
48            const time = `${counter[1]}時間${counter[2]}分${counter[3]}秒`;
49            document.getElementById('timer').textContent = time;
50        }
51    </script>
```

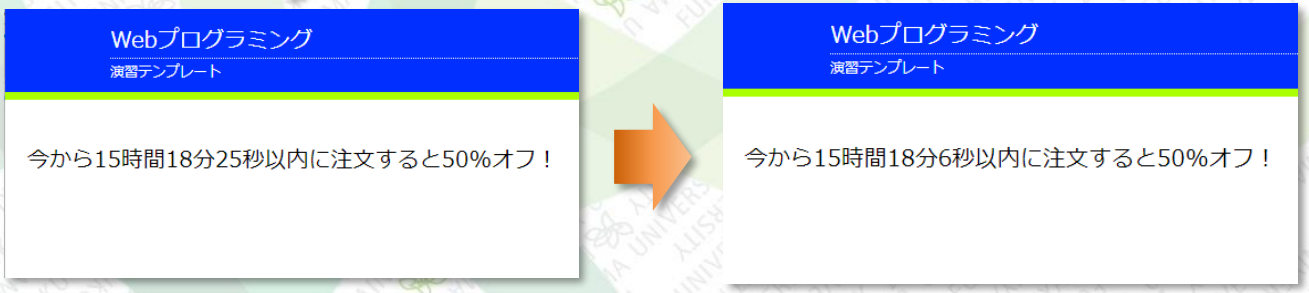
カウントダウンタイマー

- 残り時間が１秒ごとに変化するようにする
- １秒ごとに以下の処理を繰り返す
  - さらに「index1401.html」で下記を追加する

index1401.html

```
46          function recalc() {
47              const counter = countdown(goal);
48              const time = `${counter[1]}時間${counter[2]}
49              document.getElementById('timer').textContent
50              refresh();
51          }
52          function refresh() {
53              setTimeout(recalc, 1000);
54          }
55          recalc();
56      </script>
```

１秒ごとに残り時間が切り替わるようになった



カウントダウンタイマー

- 残り時間が１秒ごとに変化するようにする
- １秒ごとに以下の処理を繰り返す

実行の順序は、以下の(1)～(4)の順番で、１秒ごとに(2)(3)(4)が繰り返される

```
(2) function recalc() {
        const counter = countdown(goal);
        const time = `${counter[1]}時間${counter[2]}分${count
        document.getElementById('timer').textContent = time;
        refresh();
    }
    function refresh() {
        setTimeout(recalc, 1000);
    }
(1) recalc();
```

(3)

(4)

「待ち時間」後に「関数」を１度だけ実行する

```
setTimeout( 関数, 待ち時間 );
```

※「待ち時間」はミリ秒で指定

※「関数」は () を付けないように注意

カウントダウンタイマー

- 残り時間が１秒ごとに変化するようにする
  - 一桁表示のときに十の位に'0'を表示する(秒のところ)
  - 「index1401.html」で下記を追加する

index1401.html

```
46  function recalc() {
47      const counter = countdown(goal);
48      //const time = `${counter[1]}時間${counter[2]}分${counter[3]}秒`;
49      const sec2 = String(counter[3]).padStart(2,'0');
50      const time = `${counter[1]}時間${counter[2]}分${sec2}秒`;
51      document.getElementById('timer').textContent = time;
```

「1秒」や「6秒」が「01秒」や「06秒」となる

今から14時間34分6秒以内に注文すると50%オフ！ → 今から14時間34分06秒以内に注文すると50%オフ！

padStart()メソッド、指定した桁数でそろえる

文字列.padStart(揃える桁数, 埋める文字);

※padStart()メソッドはStringクラスのメソッド

イメージの切り替え

- サムネイルをクリックすると画像が切り替わる…を作ってみる
- テンプレートの「index.html」をコピー、ファイル名を「index1402.html」に変更＆下記を追記する

```
7       <link href="style.css" rel="stylesheet">
8       <style>
9           section img {
10              max-width: 100%;
11          }
12          .center {
13              margin: 0 auto 0 auto;
14              max-width: 90%;
15              width: 500px;
16          }
17          ul {
18              display: flex;
19              margin: 0;
20              padding: 0;
21              list-style-type: none;
22          }
23          li {
24              flex: 1 1 auto;
25              margin-right: 8px;
26          }
27          li:last-of-type {
28              margin-right: 0;
29          }
30      </style>
31  </head>
```

index1402.html

イメージの切り替え

- サムネイルをクリックすると画像が切り替わる…
  - さらに「index1402.html」に下記を追記する

index1402.html

```
41      <section>
42          <div class="center">
43              <div>
44                  <img src="img1.jpg" id="bigimg">
45              </div>
46              <ul>
47                  <li><img src="thumb-img1.jpg" class="thumb" data-image="img1.jpg"></li>
48                  <li><img src="thumb-img2.jpg" class="thumb" data-image="img2.jpg"></li>
49                  <li><img src="thumb-img3.jpg" class="thumb" data-image="img3.jpg"></li>
50                  <li><img src="thumb-img4.jpg" class="thumb" data-image="img4.jpg"></li>
51              </ul>
52          </div>
53      </section>
54      </div><!-- /.container -->
```

- 「index1402.html」と同じフォルダに、画像ファイルをコピーする

img1.jpg, ～ img4.jpg・・・画像サイズ 1000x1000

thumb-img1.jpg ～ thumb-img4.jpg・・・画像サイズ 300x300

イメージの切り替え

- サムネイルをクリックすると画像が切り替わる
  - ブラウザで表示してみると

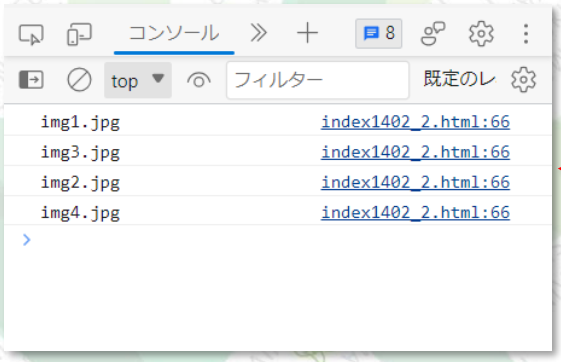
サムネイルをクリックすると、上部の画像が切り替わるようにする

イメージの切り替え

- サムネイルをクリックすると画像が切り替わる…
  - クリックしたサムネイルのdata-image属性の値をコンソールに表示する、「index1402.html」に下記を追記

index1402.html

```
60          </footer>
61          <script>
62              'use strict';
63              const thumbs = document.querySelectorAll('.thumb');
64              thumbs.forEach(function(item, index) {
65                  item.onclick = function() {
66                      console.log(this.dataset.image);
67                  }
68              });
69          </script>
70      </body>
```



サムネイルをクリックすると、ファイル名(data-image属性の値)が表示される

イメージの切り替え

- サムネイルをクリックすると画像が切り替わる…
  - document.querySelectAll() メソッド

querySelectAll()メソッドは、()で指定されたCSSのセレクタにマッチする要素すべてを取得する

```
document.querySelecorAll('CSSセレクタ');
```

```
const thumbs = document.querySelectorAll('.thumb');
```

‘.thumb’ を指定しているので、これにマッチする

```
<img src="thumb-img1.jpg" class="thumb" data-image="img1.jpg">
<img src="thumb-img2.jpg" class="thumb" data-image="img2.jpg">
<img src="thumb-img3.jpg" class="thumb" data-image="img3.jpg">
<img src="thumb-img4.jpg" class="thumb" data-image="img4.jpg">
```

がすべて取得される

※querySelectorAll()で取得した変数は NodeList型 のオブジェクトであり、配列オブジェクトのメソッドは基本的に使えない

イメージの切り替え

- サムネイルをクリックすると画像が切り替わる…
  - forEach()メソッド

配列の各要素の繰り返し処理を行う

```
配列.forEach(function(item, index){
  処理内容をここに書く
});
```

「function(){～}」の ～ の部分が、配列のすべての要素に対して繰り返し実行される

```
thumbs.forEach(function(item, index) {
```

回数	item	index
1回目	<img src="thumb-img1.jpg" … >	0
2回目	<img src="thumb-img2.jpg" … >	1
3回目	<img src="thumb-img3.jpg" … >	2
…	…	…

イメージの切り替え

- サムネイルをクリックすると画像が切り替わる…
  - item.onclickイベント

```
item.onclick = function() {
```

item(ここでは<img src="thumb-img1.jpg" … >など)で、マウスがクリックされたときに = に続くfunctionの{～}内の処理が実行される

  - this

```
console.log(this.dataset.image);
```

thisはイベントが発生した(マウスがクリックされた)要素を指す

イメージの切り替え

- サムネイルをクリックすると画像が切り替わる…
  - data-imageはカスタムデータ属性

data-???? の????は、自由につけてよい

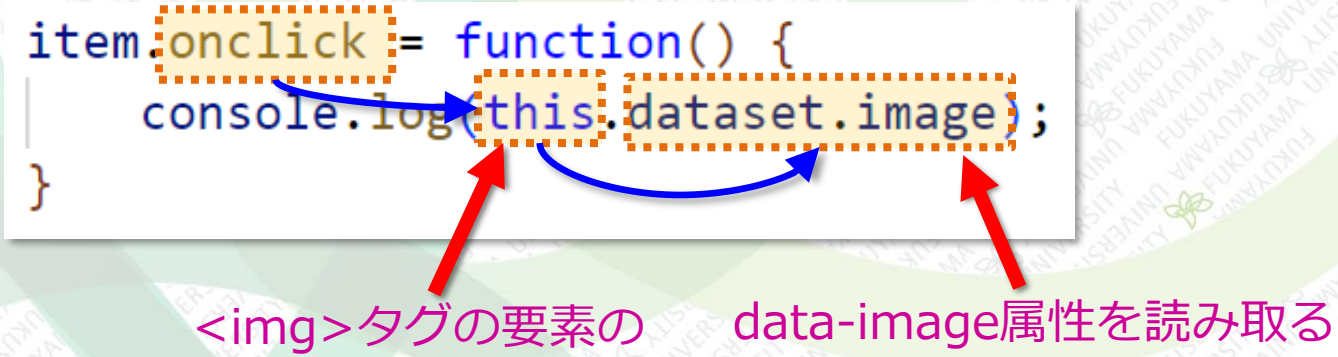
```
<タグ名 data-image="img1.jpg">
```

この部分は自由に決めてよい

  - data-????属性の値を読み取る

```
取得した要素.dataset.????
```

```
item.onclick = function() {
    console.log(this.dataset.image);
}
```



イメージの切り替え

- サムネイルをクリックすると画像が切り替わる…
  - 画像を切り替える、「index1402.html」に以下を追記

index1402.html

```
64          thumbs.forEach(function(item, index) {
65              item.onclick = function() {
66                  //console.log(this.dataset.image);
67                  document.getElementById('bigimg').src = this.dataset.image;
68              }
69          });
```

サムネイルをクリックすると、メインの画像が切り替わるようになる

```
document.getElementById('bigimg').src = this.dataset.image;
```

```
<div>
    <img src="img1.jpg" id="bigimg" data-image="img3.jpg">
</div>
```

<img src="img1.jp" が data-image の"img3.jpg"に書き換えられる

演習１

テンプレートの「index.html」をコピーして、ファイル名「ensyu01.html」に変更しなさい。
講義資料の「index1402.html」の画像を使用し、メインの画像のみ表示(サムネイルは削除)して、さらに１秒間隔で自動的に画像が切り替わるようにしなさい。画像ファイルは、下記のように配列にして利用するとよい。

```
let imgfile = ["img1.jpg", "img2.jpg", "img3.jpg", "img4.jpg"];
```

＜実行例＞

はじめに戻る