

タイマー関数

一定時間ごとに処理を繰り返す

setInterval(listener, delay)

setIntervalは、一定の遅延間隔ごとに指定された関数などを実行します。


setIntervalは、インターバルタイマーを一意に識別するインターバルIDを返します。

返されたインターバルIDを使うことで対象のインターバルタイマーを削除することができます。

setInterval() - Web API | MDN

setInterval() メソッドは Window および Worker メソッドで提供され、一定の遅延間隔を置いて関数やコードスニペットを繰り返し呼び出します。

 <https://developer.mozilla.org/ja/docs/Web/API/setInterval>


 mdn web docs


clearInterval(intervalID)

clearIntervalは、指定されたインターバルIDに対応したインターバルタイマーを削除します。

clearInterval() - Web API | MDN

グローバルの `clearInterval()` メソッドは、以前に `setInterval()` の呼び出しによって確立されたタイマーを利用した繰り返し動作を取り消します。 指定された引数で前回確立されたアクション

 <https://developer.mozilla.org/ja/docs/Web/API/clearInterval>

 mdn web docs

```
<!DOCTYPE html>
<html lang="ja">
<head>
  <meta charset="UTF-8">
  <title>Web演習 2 - サンプル 9</title>
</head>
<body>
  <h1>タイマー関数の利用</h1>
  <h2>一定時間間隔の繰り返し</h2>
  <p id="txt_interval">color name list</p>

  <button type="button" id="btn_interval_start">start</button>
  <button type="button" id="btn_interval_stop">stop</button>

  <script src="sample09_1.js"></script>
</body>
</html>
```

```
{

  //element
  const doc = document;
  const intervalText = doc.querySelector('#txt_interval');
  const intervalStartButton = doc.querySelector('#btn_interval_start');
  const intervalStopButton = doc.querySelector('#btn_interval_stop');

  // setting
  const colors = [ 'red', 'green', 'blue', 'tomato', 'lightgreen', 'skyblue' ];
  let intervalId = null;
  let intervalCount = 0;
  const delay = 1000;

  // #btn_interval_start click event
  intervalStartButton.addEventListener('click', ()=>{
    if(!intervalId) {
      intervalId = setInterval(()=>{
        let index = intervalCount % colors.length;
```

```

        intervalText.innerText = colors[index];
        intervalText.style.color = colors[index];
        intervalCount++;
    }, delay);
}
});

// #btn_interval_stop click event
intervalStopButton.addEventListener('click', ()=>{
    if(intervalId){
        clearInterval(intervalId);
        intervalId = null;
    }
});
}

```

時間経過後に処理を実行

setTimeout(listener, delay)

setTimeoutは、指定された時間を経過した際に、指定された関数などを実行します。


setTimeoutは、タイマーを一意に識別するタイムアウトIDを返します。

返されたタイマーIDを使うことで対象のタイマーを削除することができます。

setTimeout() - Web API | MDN

グローバルの setTimeout() メソッドは、時間切れになると関数または指定されたコードの断片を実行するタイマーを設定します。

 <https://developer.mozilla.org/ja/docs/Web/API/setTimeout>

 mdn web docs


clearTimeout(timeoutID)

clearTimeoutは、指定されたタイマーIDに対応したタイマーを削除します。

clearTimeout() - Web API | MDN

グローバルの clearTimeout() メソッドは、 setTimeout() の呼び出しによって以前に確立されたタイムアウトを解除します。

 <https://developer.mozilla.org/ja/docs/Web/API/clearTimeout>

 mdn web docs

```
<!DOCTYPE html>
<html lang="ja">
<head>
  <meta charset="UTF-8">
  <title>Web演習 2 - サンプル9</title>
</head>
<body>

  <h1>タイマー関数の利用</h1>
  <h2>指定時間経過後の実行</h2>
  <p id="txt_timeout">メッセージ</p>
  <button type="button" id="btn_timeout_start">start</button>
  <button type="button" id="btn_timeout_stop">stop</button>

  <script src="sample09_2.js"></script>
</body>
</html>
```

```
{
  //element
  const doc = document;
  const timeoutText = doc.querySelector('#txt_timeout');
```

```
const timeoutStartButton = doc.querySelector('#btn_timeout_start');
const timeoutStopButton = doc.querySelector('#btn_timeout_stop');

// setting
let timerId = null;
const deley = 2000;

// #btn_timeout_start click event
timeoutStartButton.addEventListener('click', ()=>{
  timeoutText.innerText = 'タイマースタート';
  if(!timerId) {
    timerId = setTimeout(()=>{
      timerId = null;
      timeoutText.innerText = 'timeoutで設定した時間を経過';
    },deley);
  }
});

// #btn_timeout_stop click event
timeoutStopButton.addEventListener('click', ()=>{
  timeoutText.innerText = 'タイマーストップ';
  if(timerId) {
    clearTimeout(timerId);
    timerId = null;
  }
});
}
```