JavaScript基礎実習

05. イベント処理

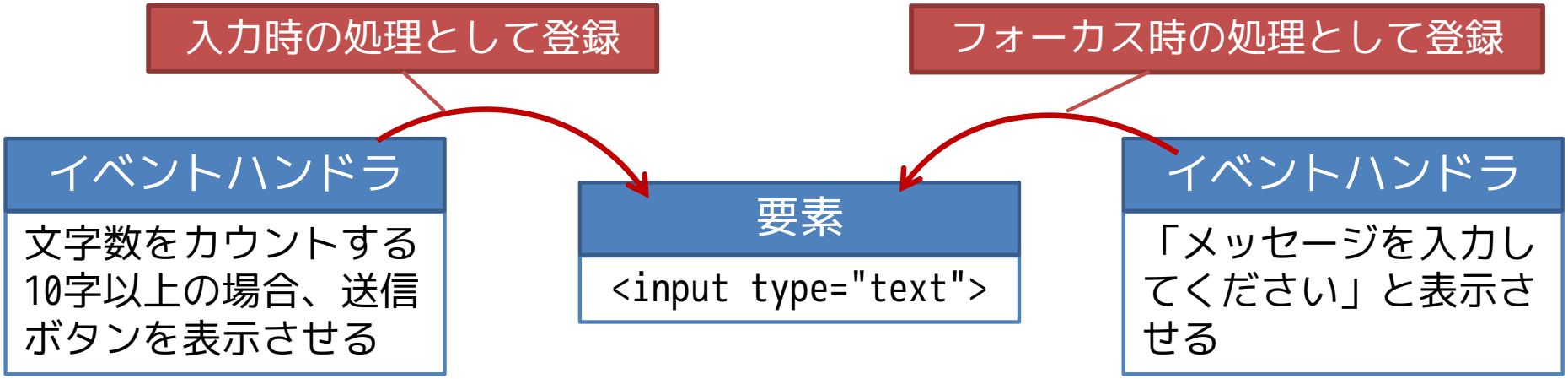
株式会社ジードライブ

今回学ぶこと

- ユーザーが何かしらの操作をしたタイミングや一定時間ごとに処理を実行する方法
  - マウスイベント
  - イベントオブジェクトの利用
  - フォームイベント
  - デフォルトの処理のキャンセル
  - タイマー処理

イベントハンドラ

- Webページ上では、フォーム入力、ボタンクリック…といった様々な出来事(イベント)が発生する
- これらのイベントが発生した際の対応処理をイベントハンドラと呼び、関数として定義する
- イベントハンドラ関数は、イベントの種類と共に、要素に紐づけることができる
  - **addEventListener(イベントの種類，イベントハンドラ関数)** というメソッドを利用する



addEventListener

- 要素にイベントハンドラ関数を紐づけるには、下記のメソッドを使用する

```
要素.addEventListener('イベントの種類', イベントハンドラ関数)
```

  - イベントの種類には、click, focus, input, … などがある
  - イベントハンドラ関数は無名関数(アロー関数)で記述可能

記述例: #btnをクリックしたら、「こんにちは」と表示する

```
const btn = document.getElementById('btn');
btn.addEventListener('click', () => alert('こんにちは'));
```

イベントの種類

イベントハンドラ(イベント発生時の処理内容)

練習

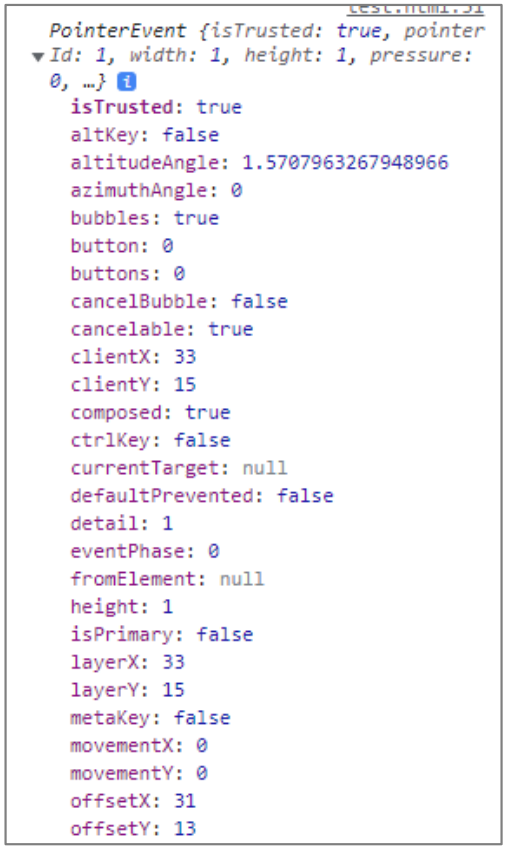
- 練習05-1

Eventオブジェクト

- イベントハンドラ関数では、引数を設定できる
  - この引数はEvent型のオブジェクトで、イベントの種類やそのイベントが発生した要素の情報などを保持している

```
const btn = document.getElementById('btn');
btn.addEventListener('click', event => {
  console.log(event);
});
```

任意の引数名でよい
クリック時のイベントなので、
クリックされた位置の座標など
も保持している



targetプロパティの利用

- イベントを呼び出した対象を参照する場合は、Eventオブジェクトのtargetプロパティを利用する

```
<button>佐藤</button>
<button>鈴木</button>
<button>山田</button>

<script>
// ボタンの名前を取得して、挨拶を表示する
const buttons = document.querySelectorAll('button');
buttons.forEach(button => {
  button.addEventListener('click', event => {
    const name = event.target.innerText;
    alert(`こんにちは、${name}さん`);
  });
});
</script>
```

クリックされたボタンを指す

カスタムデータ属性

- 開発者は独自の属性をHTMLタグに設定することができる
  - 属性名は必ず data- で始める必要がある
  - JavaScriptからは、getAttribute()メソッドでアクセスできる

```
<p>画像をクリックすると鳴き声がします。</p>
<img src="images/cat.jpg" data-sound="audio/meow.mp3">
<img src="images/dog.jpg" data-sound="audio/bowwow.mp3">

<script>
const elements = document.querySelectorAll('img');
elements.forEach(element => {
  element.addEventListener('click', event => {
    const audio = new Audio();
    audio.src = event.target.getAttribute('data-sound');
    audio.play();
  });
});
```

targetを基準にした要素の取得

- イベントを呼び出した対象を基準に、他の要素を選択して操作する場合には、以下のプロパティを利用する

プロパティ	説明
target**.parentNode**	親要素を取得する
target**.previousElementSibling**	同階層の１つ前の要素を取得
target**.nextElementSibling**	同階層の１つ次の要素を取得
target**.children**	子要素のリストを取得する 各子要素にはインデックス番号を使いアクセスする。また、forEachは使えないので、ループ処理はfor文で対応する

targetを基準にした要素の取得

記述例: クリックされたボタンの次の要素を非表示にする

```
<button>猫を非表示にする</button>
<img src="images/cat.jpg" width="300">

<button>犬を非表示にする</button>
<img src="images/dog.jpg" width="300">

<script>
document
.querySelectorAll('button')
.forEach(button => button.addEventListener('click', event => {
  event.target.nextElementSibling.style.display = 'none';
}));
</script>
```

練習

- 練習05-2
- 練習05-3

マウスイベント

- addEventListenerの引数として渡すことができるマウスイベントの種類としては、以下のようなものが存在する

イベント	用途
click	クリックした時の処理の登録
dblclick	ダブルクリックした時の処理の登録
mouseover	マウスカーソルが要素の上に移動した時の処理の登録
mouseout	マウスカーソルが要素から外れた時の処理の登録
mousedown	マウスボタンが下がった時の処理の登録
mouseup	マウスボタンが上がった時の処理の登録
mousemove	マウスカーソルが移動した時の処理の登録

マウスイベントの例

例：画像にマウスカーソルを乗せると、画像のalt属性値が
h1要素として表示される

```
<h1>UNTITLED</h1>
<img src="images/cat.jpg" alt="空を見上げる猫" width="300">
<img src="images/dog.jpg" alt="尻尾を振る子犬" width="300">

<script>
const h1 = document.querySelector('h1');
document.querySelectorAll('img')
.forEach(img => {
  img.addEventListener('mouseover', event => {
    h1.textContent = event.target.getAttribute('alt');
  });

  img.addEventListener('mouseout', event => {
    h1.textContent = 'UNTITLED'
  });
});
</script>
```

練習

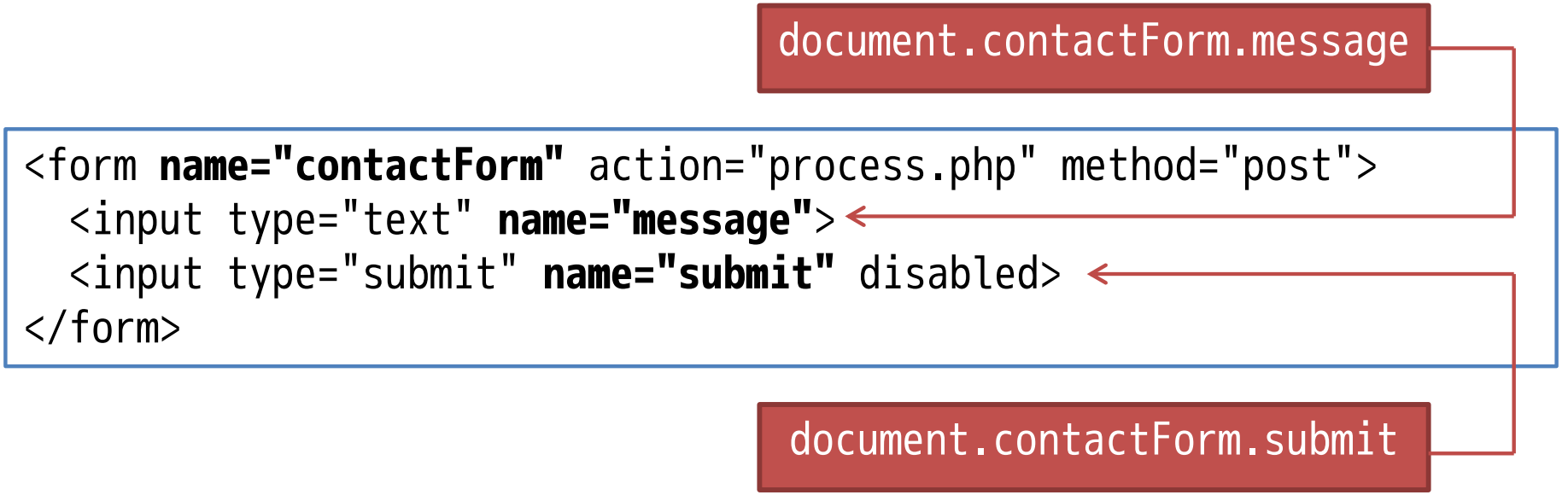
- 練習05-4

フォームへのアクセス

- フォームの部品へは、name属性を使いアクセスできる
  - これまでに学習してきたgetElementById()等を使ってもよい

書式

```
document.formタグのname属性値.フォーム部品のname属性値
```



フォーム値の取得

- テキストボックスの入力値へは、valueプロパティでアクセスすることができる
  - 入力値は「文字列」になるので必要に応じて数値に変換する

```
<form name="keisan">
  <input type="submit" value="計算">
  <input type="number" name="num1" value="0"> +
  <input type="number" name="num2" value="0"> =
  <input type="number" name="answer" value="0" readonly>
</form>

<script>
document.keisan.addEventListener('submit', event => {
  event.preventDefault(); // 送信しない(後述)
  const num1 = Number(document.keisan.num1.value);
  const num2 = Number(document.keisan.num2.value);
  document.keisan.answer.value = num1 + num2;
});
</script>
```

入力不可

input type="number" であっても、数値に変換する必要がある
⇒ Number('文字列') で数値に変換

フォーム値の取得

- ラジオボタンやチェックボックスのように、同じname属性を設定するものは、forやforEachを使い処理する必要がある

```
<form name="myform">
  <input type="checkbox" name="pet" value="dog">犬
  <input type="checkbox" name="pet" value="cat">猫
  <input type="checkbox" name="pet" value="other">その他
  <input type="submit">
</form>

<script>
document.myform.addEventListener('submit', event => {
  event.preventDefault(); // 送信しない(後述)
  document.myform.pet.forEach(p => {
    if(p.checked) console.log(p.value);
  });
});
</script>
```

チェックが入っている場合は、true

フォームイベント

- addEventListenerの引数として渡すことができるフォームイベントの種類としては、以下のようなものが存在する

イベント	用途
change	テキスト入力が確定した時、ラジオボタン等の選択で変更があった時の処理の登録
input	テキスト入力中の処理の登録
submit	フォーム送信時の処理の登録
focus	フォームの部品にフォーカスが当たった時の処理の登録
blur	フォーカスが外れた時の処理の登録

フォームイベントの例

例: 入力中の文字列が10字以上の場合に、送信ボタンを活性化する

```
<form name="messageForm" action="process.php" method="post">
  <input type="text" name="messageInput">
  <input type="submit" name="submit" disabled>
</form>
<script>
document. messageForm.messageInput.addEventListener('input', event => {
  if(event.target.value.length >= 10) {
    messageForm.submit.removeAttribute('disabled');
  } else {
    messageForm.submit.setAttribute('disabled', true);
  }
});
</script>
```

ボタンを非活性化

valueプロパティで入力値を取得
lengthプロパティで文字数を取得

デフォルトの処理のキャンセル

- イベント内の処理で **preventDefault()** を使うと、そのイベントの**デフォルトの処理**をキャンセルすることができる
  - リンク先への遷移、フォームの送信など

記述例：送信前に確認を行う

```
<form name="registerForm" action="process.php" action="post">
  メールアドレス: <input type="email" name="email" required>
  <input type="submit" value="登録">
</form>
<script>
document.registerForm.addEventListener('submit', event => {
  if(!confirm('登録を確定してよろしいですか？')) {
    event.preventDefault(); // 送信をキャンセル
  }
});
</script>
```

confirmは確認用のウィンドウを表示させるメソッド
⇒ OKを選択するとtrueを返す

練習

- 練習05-5

setTimeout()

- 一定時間後に実行したい処理を記述するためのメソッド
  - 時間はミリ秒で指定する(1000ミリ秒 = 1秒)
  - 戻り値はタイマーIDで、setTimeout()のキャンセルに利用する

書式

```
setTimeout(関数, 時間)
```

記述例：ボタンをクリックして３秒後にメッセージが表示される

```
<button id="start">スタート</button>
<script>
document.getElementById('start').addEventListener('click', () => {
   setTimeout(() => alert('３秒経過しました'), 3000);
});
</script>
```

clearTimeout()

- タイマーIDを元に、setTimeout()をキャンセルする

書式

clearTimeout(タイマーID)

記述例：ストップボタンを押下するとsetTimeout()がキャンセルされる

```
<button id="start">スタート</button>
<button id="stop">ストップ</button>
<script>
let timerId;
document.getElementById('start').addEventListener('click', () => {
  timerId = setTimeout(() => alert('３秒経過しました'), 3000);
});
document.getElementById('stop').addEventListener('click', () => {
  clearTimeout(timerId);
  alert('キャンセルしました');
});
```

setInterval()

- 一定時間後ごとに処理を繰り返すためのメソッド
  - 時間はミリ秒で指定する(1000ミリ秒 = 1秒)
  - 戻り値はタイマーIDで、setInterval()のキャンセルに利用する

書式

```
setInterval(関数, 時間)
```

記述例：経過秒数を表示する

```
<h1>0</h1>
<script>
let counter = 1;
setInterval(() => {
  document.querySelector('h1').textContent = counter;
  counter++;
}, 1000);
</script>
```

clearInterval()

- タイマーIDを元に、setInterval()をキャンセルする

書式

```
clearInterval(タイマーID)
```

記述例：ストップボタンを押下するとsetInterval()がキャンセルされる

```
<h1>0</h1>
<button id="stop">ストップ</button>
<script>
let counter = 1;
const timerId = setInterval(() => {
  document.querySelector('h1').textContent = counter;
  counter++;
}, 1000);
document.getElementById('stop').addEventListener('click', () => {
  clearInterval(timerId);
});
</script>
```

練習

- 練習05-6