JavaScript応用実習

01. DOMの操作(応用)

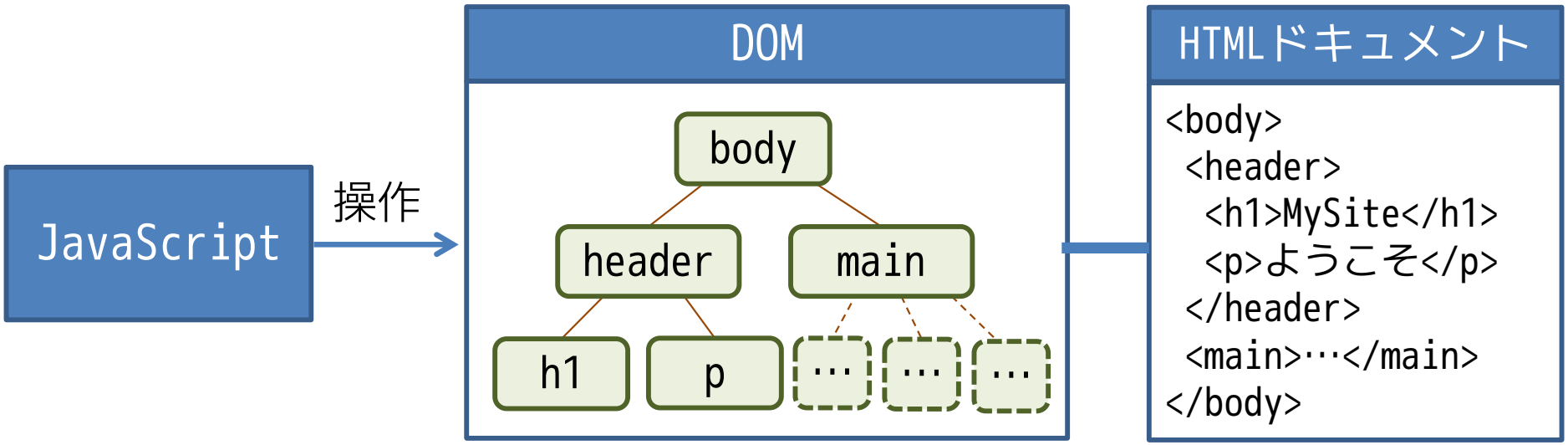
株式会社ジードライブ

今回学ぶこと

- DOMの構造
- HTML要素(Elementオブジェクト)の生成
- templateタグの利用

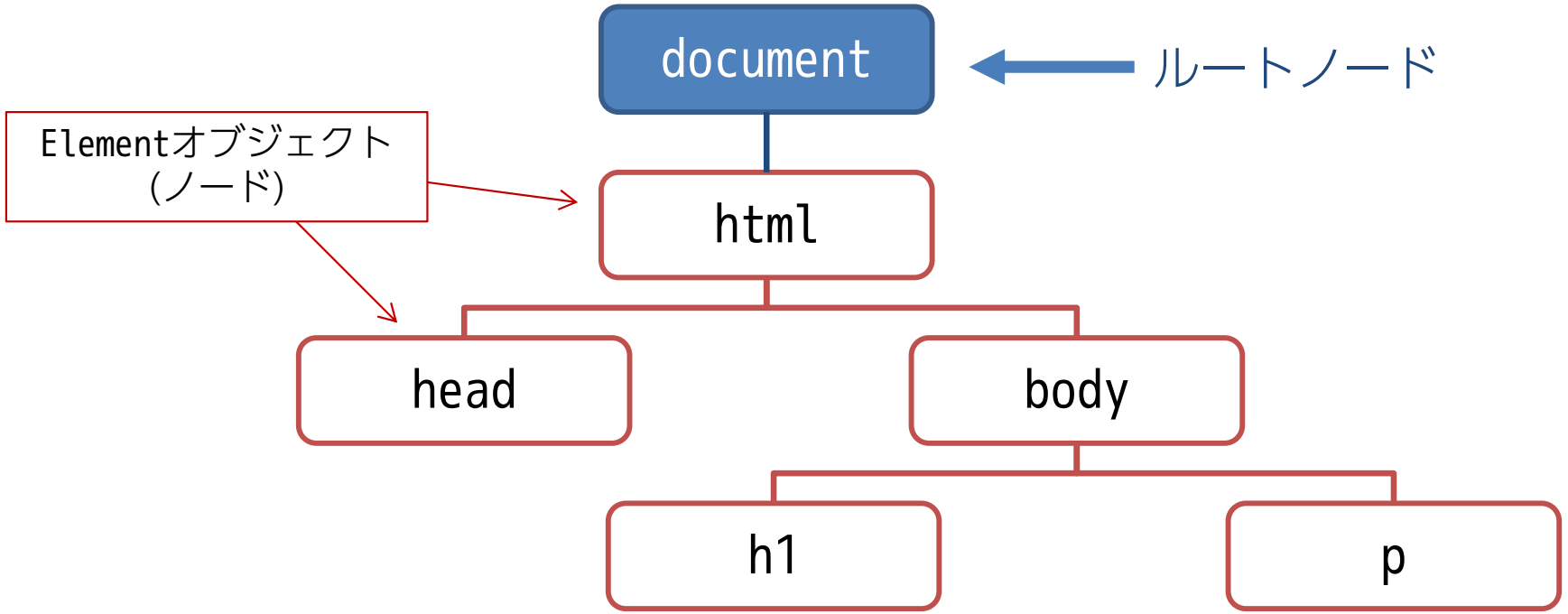
DOM

- DOM(Document Object Model)はHTMLドキュメントを操作するためのインターフェース
  - JavaScriptは、DOMを通じてWebページ内の**要素を動的**に操作することができる



DOMの構造

- DOMは、HTML要素(Elementオブジェクト)のツリー構造として表現される
  - ツリー上の各Elementオブジェクトをノードと呼ぶ
    (節 という意味)



Elementオブジェクトの生成

- DOMを形成するElementオブジェクトは、document.createElement()メソッドで生成することができる
  - 生成しただけでは画面には表示されない
  - 画面に表示させるにはDOMツリーに加わる必要がある
    ⇒ **append()**などのメソッドを使用する

記述例: p要素をbody内に追加する

```
// p要素を作成(この時点では画面に表示されない)
const paragraph = document.createElement('p');
paragraph.textContent = 'こんにちは';

// p要素をbody内に追加(画面に表示される)
document.body.append(paragraph);
```

DOMツリーへの追加①

- ある要素内に、別要素を追加するには、prepend(), append()メソッドを使用する
  - prepend(): ある要素内の最初に別要素を追加
  - append(): ある要素内の末尾に別要素を追加

書式

引数は通常の文字列でもよい

```
親になる要素.prepend(子要素1,子要素2, …);
親になる要素.append(子要素1,子要素2, …);
```

例

```
// div#box内の先頭にp要素を追加
const paragraph = document.createElement('p');
paragraph.textContent = 'こんにちは';
document.getElementById('box').prepend(paragraph);
```

例：DOMツリーへの追加①

元のHTML

```
<div id="info">
  <h2>会社概要</h2>
</div>
```

JavaScript

```
// DOM要素の作成
const h1 = document.createElement('h1');
h1.textContent = '株式会社ABC';
const p = document.createElement('p');
p.textContent = '英会話スクールです';

// DOMツリーへの追加
document.getElementById('info').prepend(h1);
document.getElementById('info').append(p);
```

最終的なDOMツリー

```
<div id="info">
  <h1>株式会社ABC</h1>
  <h2>会社概要</h2>
  <p>英会話学校スクールです</p>
</div>
```

DOMツリーへの追加②

- ある要素の前後に、別要素を追加するには、before(), after()メソッドを使用する
  - before(): ある要素の前に別要素を追加
  - After(): ある要素の後に別要素を追加

書式

引数は通常の文字列でもよい

```
基準となる要素.before(追加要素1, 追加要素2, …);
基準となる要素.after(追加要素1, 追加要素2, …);
```

例

```
// div#box内の次の要素としてp要素を追加
const paragraph = document.createElement('p');
paragraph.textContent = 'こんにちは';
document.getElementById('box').after(paragraph);
```

例：DOMツリーへの追加②

元のHTML

```
<div id="info">
  <h2>会社概要</h2>
</div>
```

JavaScript

```
// DOM要素の作成
const h1 = document.createElement('h1');
h1.textContent = '株式会社ABC';
const p = document.createElement('p');
p.textContent = '英会話スクールです';

// DOMツリーへの追加
document.getElementById('info').before(h1);
document.getElementById('info').after(p);
```

最終的なDOMツリー

```
<h1>株式会社ABC</h1>
<div id="info">
  <h2>会社概要</h2>
</div>
<p>英会話学校スクールです</p>
```

DOMツリーからの削除

- 要素を削除するには、remove()メソッドを使用する
  - remove(): 指定した要素を削除

書式

```
削除対象の要素.remove();
```

例

```
// #infoを削除
document.getElementById('info').remove();
```

練習

- 練習01-1
- 練習01-2

template要素の利用

- templateは、HTMLの雛形(フラグメント)を格納するための要素
  - 画面上には表示されない
  - JavaScriptでの複製利用を想定している

記述例

```
<template id="item-box">
  <div class="item">
    <h3><!-- 商品名 --></h3>
    <img width="300" height="200">
    <p>価格: <span class="price"><!-- 値段 --></span>円</p>
  </div>
</template>
```

template要素の利用

- template要素内の要素を複製する場合は、importNode()メソッドを使用する

記述例

```
// テンプレート内の要素を複製
const template = document.getElementById('item-box');
const clone = document.importNode(template.content, true);

// 複製した要素の編集
clone.querySelector('.item').style.border = '1px solid #000';
clone.querySelector('h3').textContent = 'りんご';
clone.querySelector('img').setAttribute('src', 'apple.jpg');
clone.querySelector('.price').textContent = '100';

// DOMに追加
document.body.append(clone);
```

子孫要素も含めて取得

querySelector()等を使用して、
複製要素を編集する

練習

- 練習01-3