

要素の作成・追加・削除


要素の動的作成


document.createElement(TagName)

createElementは、TagNameで指定されたHTML要素を生成します。

Document.createElement() - Web API | MDN

HTML 文書において、document.createElement() メソッドは tagName で指定された HTML 要素を生成し、または tagName が認識できない場合は HTMLUnknownElement を生成します。

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

 mdn web docs

```
<!DOCTYPE html>
<html lang="ja">
<head>
  <meta charset="UTF-8">
  <title>Web演習 2 - サンプル 10</title>
</head>
<body>
  <h1>要素の動的作成と追加・削除</h1>

  <div>
    <h2>リストの操作</h2>
    <div>
      <label>項目のテキスト</label>
      <input type="text" id="list_text" value="">
    </div>
    <button id="btn_add">登録</button>
    <button id="btn_remove_all">すべて削除</button>
  </div>

  <div>
    <h2>登録済みリスト</h2>
    <ul id="list">
      <li>サンプル項目</li>
    </ul>
  </div>

  <script src="sample10.js"></script>
</body>
</html>
```

```
{
  // element
  const doc = document;
  const list = doc.querySelector('#list');
  const listText = doc.querySelector('#list_text');
  const addButton = doc.querySelector('#btn_add');
  const removeAllButton = doc.querySelector('#btn_remove_all');

  // setting
  const listItemTagName = 'li';
  const listItemTemplate = doc.createElement(listItemTagName);
  console.log(listItemTemplate);
}
```

要素の追加


parentElement.append(elementORstring)

appendは、elementまたは、テキストを親要素の最終子要素として追加します。

Element.append() - Web API | MDN

Element.append() メソッドは、一連の Node または DOMString オブジェクトを Element の最後の子の後に挿入します。DOMString オブジェクトは等価な Text ノードとして挿入されます。

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


 mdn web docs


parentElement.appendChild(element)

appendChildは、elementを親要素の最終子要素として追加します。

Node: appendChild() メソッド - Web API | MDN

appendChild() は Node インターフェイスのメソッドで、指定された親ノードの子ノードリストの末尾にノードを追加します。追加しようとしたノードが既に存在していた場合は、appendChild() はその子ノードを現在の位置から新しい位置へ移動します。

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

 mdn web docs

```
{  
  
  // element  
  const doc = document;  
  const list = doc.querySelector('#list');  
  const listText = doc.querySelector('#list_text');  
  const addButton = doc.querySelector('#btn_add');  
  const removeAllButton = doc.querySelector('#btn_remove_all');  
  
  // setting  
  const listItemTagName = 'li';  
  const listItemTemplate = doc.createElement(listItemTagName);  
  console.log(listItemTemplate);  
  
  // #btn_add click event  
  // リスト項目の追加  
  btn_add.addEventListener('click', (e)=>{  
    let text = listText.value;  
    if( text ) {  
      // listItemTemplateをappendすると1回しか追加できない。  
      const listItem = doc.createElement(listItemTagName);
```

```

        listItem.innerText = text;
        list.append(listItem);
    }
});
}

```

要素の削除


element.remove()

removeは、要素を削除します。

Element.remove() - Web API | MDN

Element.remove() は所属するツリーから要素を削除します。

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


 mdn web docs


element.removeChild(child)

removeChildは、親要素からchildで指定した子要素を削除します。

Node.removeChild() - Web API | MDN

removeChild() は Node インターフェイスのメソッドで、子ノードを DOM から取り除き、取り除いたノードを返します。

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

 mdn web docs

```

{
    // element
    const doc = document;
    const list = doc.querySelector('#list');
    const listText = doc.querySelector('#list_text');
    const addButton = doc.querySelector('#btn_add');
    const removeAllButton = doc.querySelector('#btn_remove_all');
}

```

```

// setting
const listElementTagName = 'li';
const listItemTemplate = doc.createElement(listElementTagName);
console.log(listItemTemplate);

// #btn_add click event
// リスト項目の追加
btn_add.addEventListener('click', (e)=>{
  let text = listText.value;
  if( text ) {
    // listItemTemplateをappendすると1回しか追加できない。
    const listItem = doc.createElement(listElementTagName);
    listItem.innerText = text;
    list.append(listItem);
  }
});

// #btn_remove_all click event
// リスト項目のすべて削除
removeAllButton.addEventListener('click', (e)=>{
  const listItems = list.querySelectorAll('li');
  listItems.forEach((listItem)=>{
    listItem.remove();
  });
});
}

```

ノードの複製

node.cloneNode(deep = false)

cloneNodeは、メソッドを呼び出したnodeの複製を返します。


複製されたノードは、固有のリスナーを含む、ノードのすべての属性とその値が複製されませんが、addEventListenerやイベントプロパティで追加されたイベントは複製されません。

deepにtrueを指定した場合、子ノードのテキストも含め複製されます。

Node: cloneNode() メソッド - Web API | MDN

cloneNode() は Node インターフェイスのメソッドで、このメソッドが呼び出されたノードの複製を返します。 引数でノードに含まれるサブツリーと一緒に複製するかどうかを制御できます。

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

 mdn web docs

```
{
```

```

// element
const doc = document;
const list = doc.querySelector('#list');
const listText = doc.querySelector('#list_text');
const addButton = doc.querySelector('#btn_add');
const removeAllButton = doc.querySelector('#btn_remove_all');

// setting
const listItemTagName = 'li';
const listItemTemplate = doc.createElement(listItemTagName);
console.log(listItemTemplate);

// #btn_add click event
// リスト項目の追加
btn_add.addEventListener('click', (e)=>{
  let text = listText.value;
  if( text ) {

    // listItemTemplateをappendすると1回しか追加できない。
    // listItemTemplateをcloneNodeで複製することで、毎回createElementをしないで済みます。
    const listItem = listItemTemplate.cloneNode(true);
    console.log(listItem);

    listItem.innerText = text;
    list.append(listItem);
  }
});

// #btn_remove_all click event
// リスト項目のすべて削除
removeAllButton.addEventListener('click', (e)=>{
  const listItems = list.querySelectorAll('li');
  listItems.forEach((listItem)=>{
    listItem.remove();
  });
});
}

```

最小の文書オブジェクト

DocumentFragment - Web API | MDN


DocumentFragment インターフェイスは、親ノードを持たない最小限の文書オブジェクト (文書の断片) を表します。こ

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



Document.createDocumentFragment() - Web API | MDN

新しい空の DocumentFragment を作成し、そこに DOM ノードを追加して画面外の DOM ツリーを作成します。

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

動的に追加した要素イベントを設定する

```

{

  // element
  const doc = document;

```

```

const list = doc.querySelector('#list');
const listText = doc.querySelector('#list_text');
const addButton = doc.querySelector('#btn_add');
const removeAllButton = doc.querySelector('#btn_remove_all');

// setting
const listItemTagName = 'li';
const listItemTemplate = doc.createElement(listItemTagName);
console.log(listItemTemplate);

// #btn_add click event
// リスト項目の追加
btn_add.addEventListener('click', (e)=>{
  let text = listText.value;
  if( text ) {
    // listItemTemplateをappendすると1回しか追加できない。
    const listItem = doc.createElement(listItemTagName);
    console.log(listItem);
    listItem.innerText = text;
    list.append(listItem);
  }
});

// #btn_remove_all click event
// リスト項目のすべて削除
removeAllButton.addEventListener('click', (e)=>{
  const listItems = list.querySelectorAll('li');
  listItems.forEach((listItem)=>{
    listItem.remove();
  });
});

// #list > li click event
// クリックしたリスト項目へ目印をつける
list.addEventListener('click', (e) => {
  console.log(e.target);
  const target = e.target;
  if(target.tagName == 'LI') {
    target.style.backgroundColor = 'tomato';
  }
});
}

```

要素内のマークアップへのアクセス


Element.innerHTML

innerHTMLは、要素内のマークアップを取得したり設定したりします。

要素の内容を置き換えるというより、文書に HTML を挿入するという場合には、insertAdjacentHTML を使用してください。

Element.innerHTML - Web API | MDN

Element オブジェクトの innerHTML プロパティは、要素内の HTML または XML のマークアップを取得したり設定したりします。

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

 mdn web docs

要素内への追加

Element.insertAdjacentHTML(position, text)

insertAdjacentHTMLは、第1引数で指定した位置に、第2引数で指定するテキストを HTML または XML としてパースした結果を挿入します。


これは挿入先の要素を再度パースするものではないため、既存の要素や要素内部の破壊を伴いません。

余分なシリアル化のステップを回避できる分、innerHTMLよりも高速な動作となります。

element.insertAdjacentHTML - Web API | MDN

insertAdjacentHTML() は、第二引数で指定するテキストを HTML または XML としてパースし、その結果であるノードを DOM ツリー内の指定された位置（第一引数で指定）に挿入します。これは挿入先の要素を再度パースするものではないため、既存の要素や要素内

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

 mdn web docs

Element.insertAdjacentElement(position, element)

insertAdjacentElementは、第1引数で指定した位置に、第2引数で指定する要素を挿入します。

Element.insertAdjacentElement() - Web API | MDN

insertAdjacentElement() は Element インターフェイスのメソッドで、呼び出された要素から相対的に指定された位置に、指定された要素ノードを挿入します。

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

 mdn web docs


Element.insertAdjacentText(position, text)

insertAdjacentHTMLは、第 1 引数で指定した位置に、第 2 引数で指定するテキストを挿入します。

Element.insertAdjacentText() - Web API | MDN

insertAdjacentText() メソッドは、与えられたテキストノードを、メソッドを実行した要素に対する相対的な位置に挿入します。

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

 mdn web docs

positionの指定文字列

文字列	説明
beforebegin	element の直前に挿入
afterbegin	element 内部の、最初の子要素の前に挿入
beforeend	element 内部の、最後の子要素の後に挿入
afterend	element の直後に挿入

