Webプログラミング
第11回

前回まで

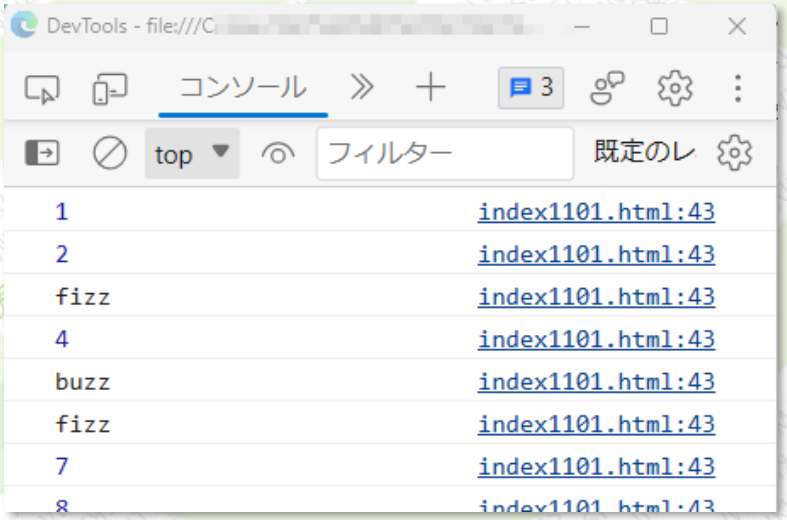
- 繰り返し
  - 回数が決まっている繰り返し
  - for文
  - 回数が決まっていない繰り返し
  - 乱数
  - while文
  - 変数、定数
- 関数（ファンクション）

今回の内容

- FizzBuzzを作ってみる
  - 関数、if文、繰り返しの復習
- 項目リストを表示する（配列）
  - 配列のデータを作成、１個だけ表示してみる
  - 配列
  - テンプレートリテラル
  - 項目をHTMLに出力する

FizzBuzzを作ってみる

- FizzBuzzとは
  - ２人以上の何人かで行う簡単なゲーム
  - 「１」「２」と順番に数字を言っていき、
  - ３で割り切れるときは「Fizz」と言う
  - ５で割り切れるときは「Buzz」と言う
  - ３でも５でも割り切れるときは「FizzBuzz」と言う
  - 間違えた人は脱落

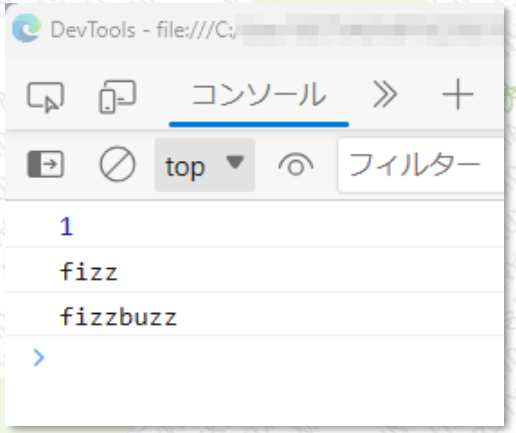


FizzBuzzを作ってみる

- テンプレートの「index.html」をコピーして、ファイル名を「index1101.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

```
27          </footer>
28          <script>
29              'use strict';
30              function fizzbuzz(num) {
31                  if(num % 3 === 0 && num % 5 === 0) {
32                      return 'fizzbuzz';
33                  } else if(num % 3 === 0) {
34                      return 'fizz';
35                  } else if(num % 5 === 0) {
36                      return 'buzz';
37                  } else {
38                      return num;
39                  }
40              }
41              console.log(fizzbuzz(1));
42              console.log(fizzbuzz(3));
43              console.log(fizzbuzz(15));
44          </script>
45      </body>
```

index1101.html



FizzBuzzを作ってみる

- FizzBuzzの判別を行い、その結果をリターンする関数を使う

```
function fizzbuzz(num) {
```

numに渡された数値が、

1. 3と5のどちらでも割り切れる場合は 'FizzBuzz' を返す
2. それ以外で、3で割り切れる場合は 'Fizz' を返す
3. それ以外で、5で割り切れる場合は 'Buzz' を返す
4. それ以外は渡された値(num)をそのまま返す

- 3で割り切れるか、5で割り切れるかどうかは割った余りが0かどうかを調べる

```
} else if(num % 3 === 0) {
```

a を b で割った余りを x に代入

```
let x = a % b;
```

- 複数の条件式を設定

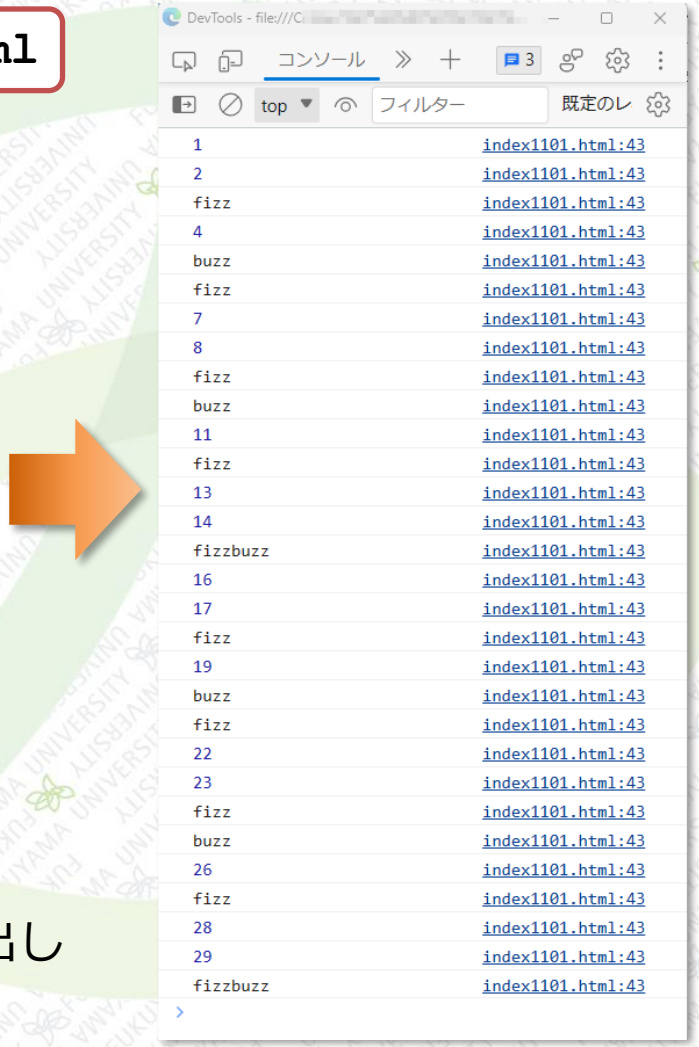
```
if(num % 3 === 0 && num % 5 === 0) {
```

FizzBuzzを作ってみる

- 1から30までの数字で表示するように修正
- 「index1101.html」に下記の内容を追記・修正する

index1101.html

```
28      <script>
29          'use strict';
30          function fizzbuzz(num) {
31              if(num % 3 === 0 && num % 5 === 0) {
32                  return 'fizzbuzz';
33              } else if(num % 3 === 0) {
34                  return 'fizz';
35              } else if(num % 5 === 0) {
36                  return 'buzz';
37              } else {
38                  return num;
39              }
40          }
41          let i = 1;
42          while( i<=30 ) {
43              console.log( fizzbuzz(i) );
44              i += 1;
45          }
46      </script>
```

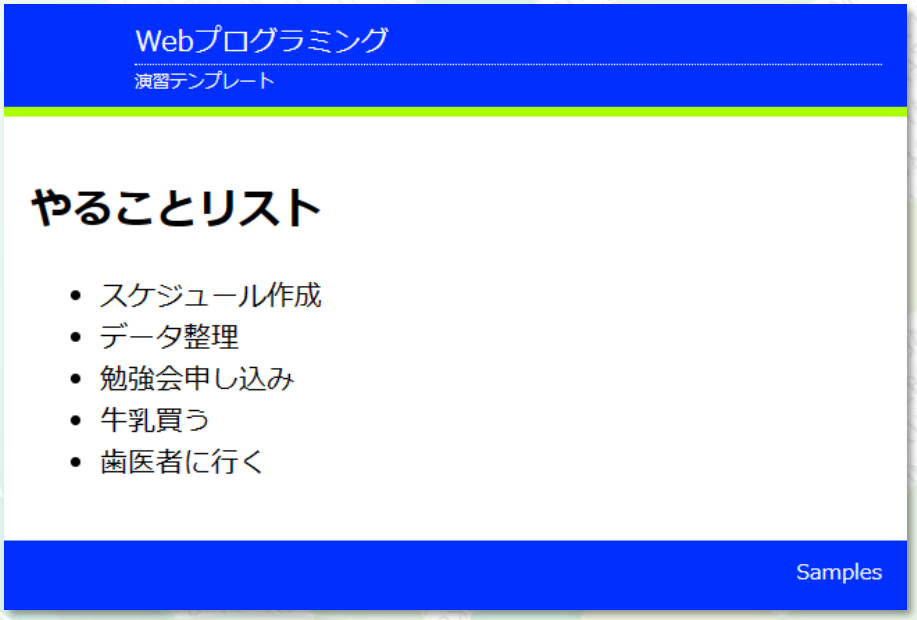


変数 i をループで回し、 fizzbuzz( ) の呼び出しで引数として i を渡す

項目をリスト表示する（配列）

- 配列のデータから箇条書きを作成・表示する

```
let todo = ['スケジュール作成', 'データ整理', '勉強会申し込み', '牛乳買う'];
todo.push('歯医者に行く');
```

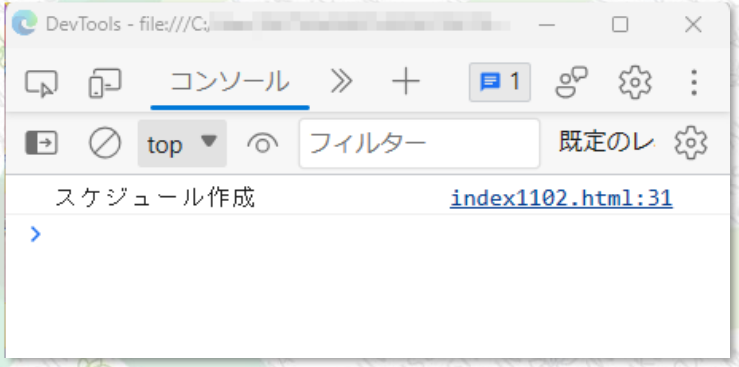


項目をリスト表示する（配列）

- 配列のデータを作成、１個だけ表示してみる
  - テンプレートの「index.html」をコピーして、ファイル名を「index1102.html」に下記を追記する

index1102.html

```
27      </footer>
28      <script>
29          'use strict';
30          let todo = ['スケジュール作成', 'データ整理', '勉強会申し込み', '牛乳買う'];
31          console.log(todo[0]);
32      </script>
33  </body>
```



項目をリスト表示する（配列）

- 配列のデータを作成、１個だけ表示してみる
  - 配列の定義と初期化

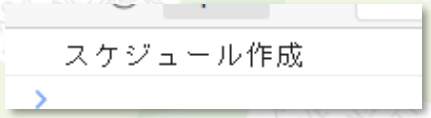
変数 todo に配列のデータを代入している

```
let todo = ['スケジュール作成', 'データ整理', '勉強会申し込み', '牛乳買う'];
```

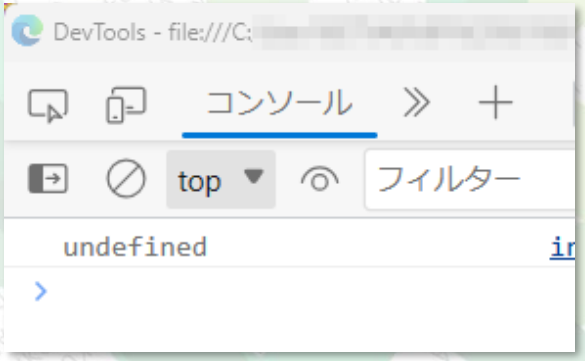
todo[0]　todo[1]　todo[2]　todo[3]

配列変数 todo の１番目のデータは todo[0]

```
console.log(todo[0]);
```



ちなみに、todo[4]を表示しようとすると、変数が存在しないので
「undefined」と表示される



項目をリスト表示する（配列）

配列

配列の書式

項目数が0個の配列を作成して変数に保存する

```
let 変数名 = [];
```

1 個以上のデータを持つ配列を作成する

```
let 変数名 = [データ0, データ1, データ2, ・・ ,データN ];
```

配列データの参照

配列 todo のデータを参照する

```
todo[インデックス値]
```

インデックス値は、 0 からスタートすることに注意する

項目をリスト表示する（配列）

- 配列のすべての要素を表示する
  - for … of 文を使う
  - 「index1102.html」を下記のように修正する

index1102.html

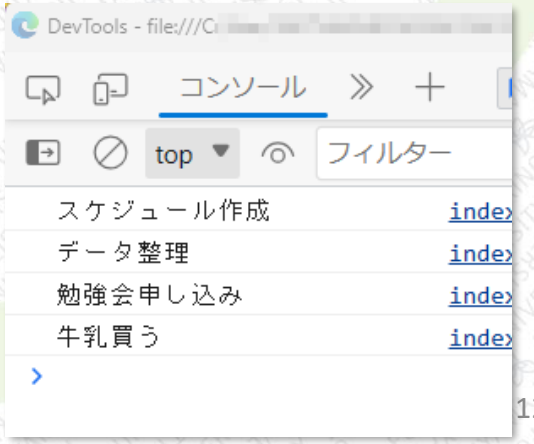
```
28      <script>
29          'use strict';
30          let todo = ['スケジュール作成', 'データ整理', '勉強会申し込み', '牛乳買う'];
31          for(let item of todo) {
32              console.log(item);
33          }
34      </script>
```

for … of を使った処理の書き方

```
for(let 変数 of 配列){
    //変数を使った処理
}
```

1回の処理ごとに、配列の要素が変数に代入される

ちなみに for … in は、要素を取り出す順番が不確定で並び順に処理されることは保証されていない

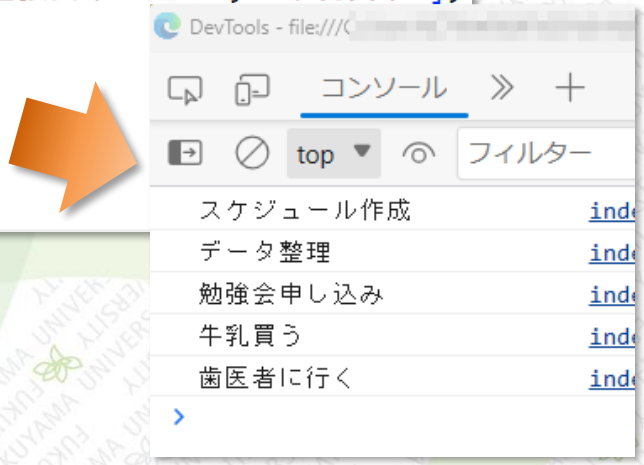


項目をリスト表示する（配列）

- 既存の配列に項目を追加する
  - 「index1102.html」で下記を追加する

index1102.html

```
28      <script>
29          'use strict';
30          let todo = ['スケジュール作成', 'データ整理', '勉強会申し込み', '牛乳買う'];
31          todo.push('歯医者に行く');
32          for(let item of todo) {
33              console.log(item);
34          }
35      </script>
```



配列の最後にデータを追加する

配列の変数.push(追加するデータ)

配列(配列オブジェクト)にデータを追加・削除する主なメソッド

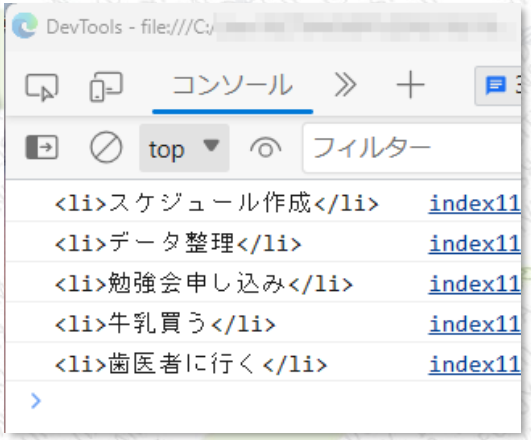
メソッド名	意味
配列の変数.pop()	最後のデータを削除する(取り出す)
配列の変数.push(データ)	配列の最後にデータを追加する
配列の変数.shift()	最初のデータを削除する
配列の変数.unshift(データ1,データ2,…)	配列の最初にデータ1,データ2,…を追加する

項目をリスト表示する（配列）

- 各項目を
～
で囲む
  - 「index1102.html」を下記のように修正する

```
28      <script>
29          'use strict';
30          let todo = ['スケジュール作成', 'データ整理', '勉強
31          todo.push('歯医者に行く');
32          for(let item of todo) {
33              const li = `<li>${item}</li>`;
34              console.log(li);
35          }
36      </script>
```

index1102.html




```
const li = `<li>${item}</li>`;
```

この文字は「バッククォート」または「バックティック」であり、「シングルクォーテーション」ではないので注意

バッククォートはキーボードの **[Shiht]+[@]** で入力する

項目をリスト表示する（配列）

- 各項目を
～
で囲む
  - テンプレートリテラル

```
const li = `<li>${item}</li>`;
```

バッククォート(`)で囲んだ文字列のことを「テンプレートリテラル」と呼ぶ

テンプレートリテラル(文字列)の中に変数の値を埋め込む

```
${変数}
```

テンプレートリテラルを使わない書き方（結果は同じ）

```
const li = '<li>' + item + '</li>';
```

項目をリスト表示する（配列）

- テンプレートリテラルのその他の機能
  - ${～}内で関数(function)を呼び出す

文字列連結での書き方（ total()関数を呼び出す ）

```
let s = '値段は' + total(8000) + '円（税込）です。';
console.log(s);
```

テンプレートリテラルの書き方

```
let s = `値段は${total(8000)}円（税込）です。`;
console.log(s);
```

  - ${～}内で計算する

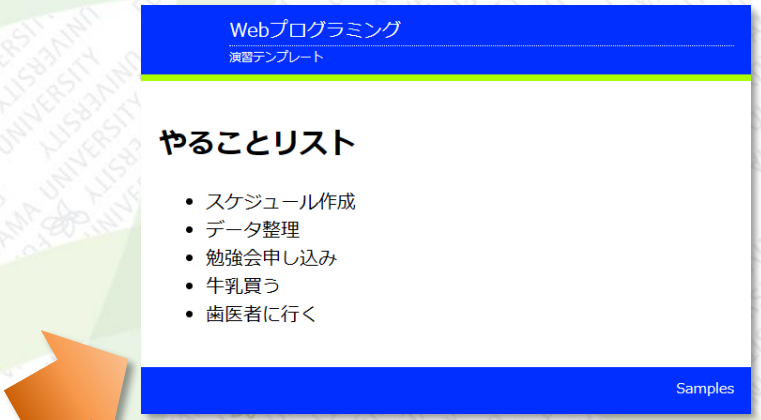
```
let sum = `大人２人：${1800*2}円`;
console.log(sum);
```

項目をリスト表示する（配列）

- 項目をHTMLに出力する
  - タグを追加して、HTMLに表示する
  - 「index1102.html」に下記を追記して修正する

index1102.html

```
18          <section>
19              <h1>やることリスト</h1>
20              <ul id="list">
21              </ul>
22          </section>
```



:

```
30      <script>
31          'use strict';
32          let todo = ['スケジュール作成', 'データ整理', '勉強会申し込み', '牛乳買う'];
33          todo.push('歯医者に行く');
34          for(let item of todo) {
35              const li = `<li>${item}</li>`;
36              document.getElementById('list').insertAdjacentHTML('beforeend', li);
37          }
38      </script>
```

項目をリスト表示する（配列）

- 項目をHTMLに出力する
  - タグを追加して、HTMLに表示する

```
document.getElementById('list').insertAdjacentHTML('beforeend', li);
```

idが 'list' の要素を取得

```
<ul id="list">
</ul>
```

HTMLの要素（ここでは<li>）を挿入する
'beforeend'の場合、開始タグの前に挿入

insertAdjacentHTML()メソッドの書式

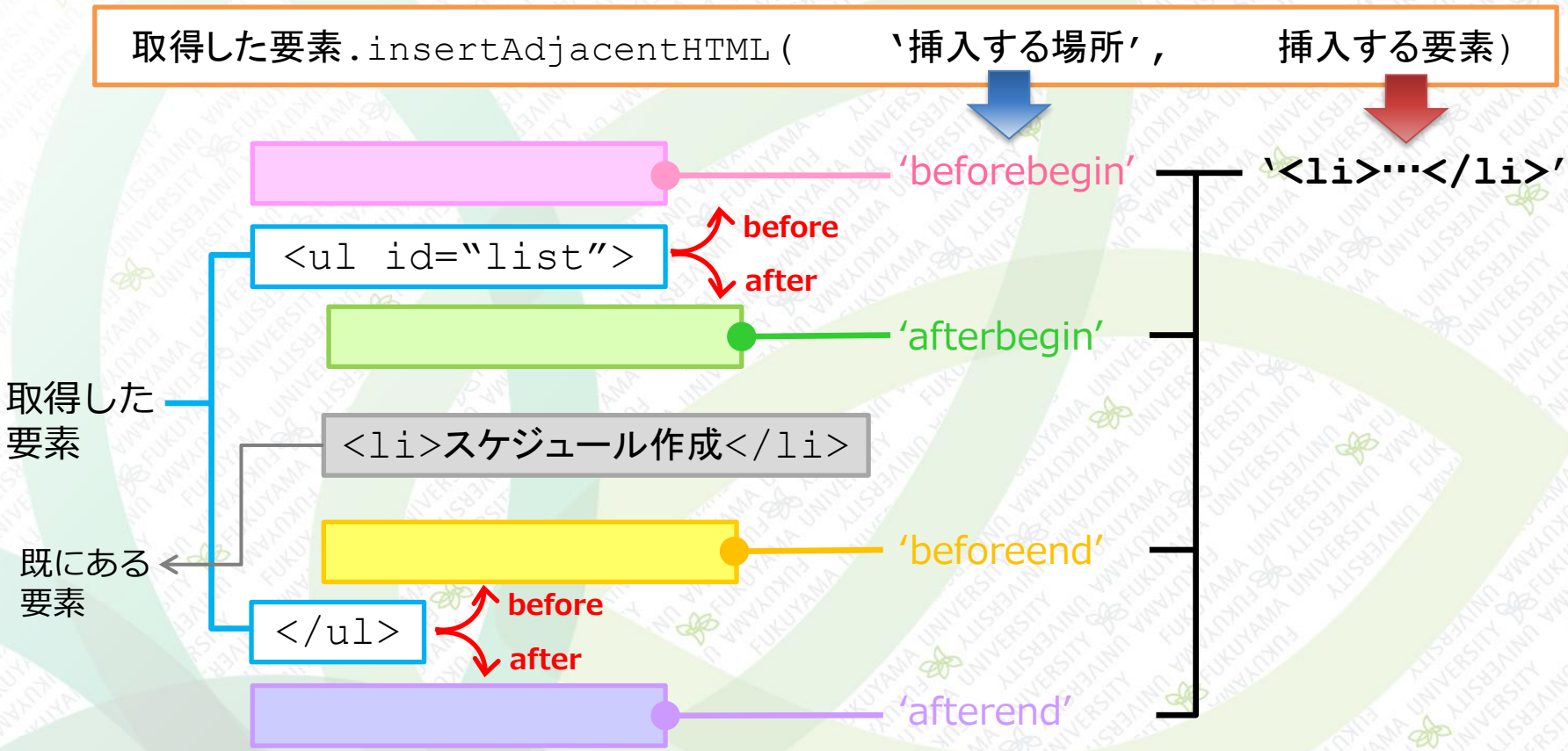
取得した要素.insertAdjacentHTML('挿入する場所', 挿入する要素)

'挿入する場所'に設定できるキーワード

キーワード	意味
'beforebegin'	取得した要素の開始タグの前に挿入
'afterbegin'	取得した要素の開始タグのすぐ後に挿入
'beforeend'	取得した要素の終了タグのすぐ前に挿入
'afterend'	取得した要素の終了タグの後に挿入

項目をリスト表示する（配列）

- 項目をHTMLに出力する
  - insertAdjacentHTML()メソッドのパラメータと挿入される場所

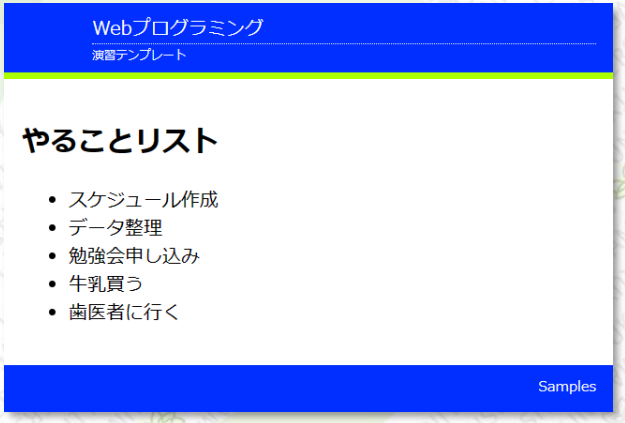


`.insertAdjacentHTML('beforeend', li);` は、 の位置に
…
を挿入する

項目をリスト表示する（配列）

- 項目をHTMLに出力する
  - タグを追加して、HTMLに表示する
  - JavaScriptの処理により、最終的に下記のようなHTMLが出力される

```
<section>
    <h1>やることリスト</h1>
    <ul id="list">
        <li>スケジュール作成</li>
        <li>データ整理</li>
        <li>勉強会申し込み</li>
        <li>牛乳買う</li>
        <li>歯医者に行く</li>
    </ul>
</section>
```



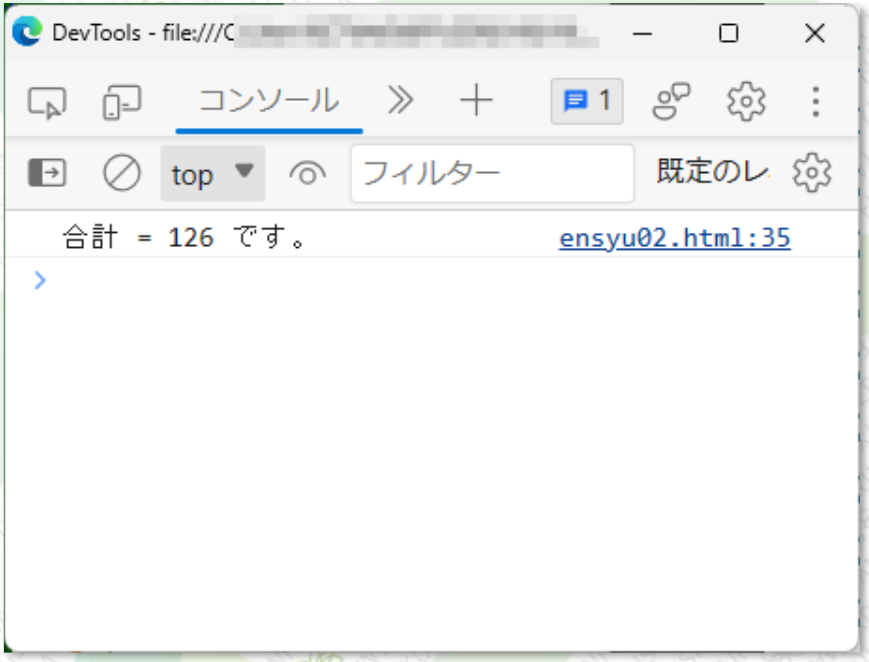
演習１

まずはじめに、解説で作成した「index1101.html」をコピーして、ファイル名を「ensyu01.html」に変更しなさい（style.cssもコピーする）。
次に、3のつく数字と、3の倍数の数字は「アホになる」とコンソールに表示するように修正しなさい。ただし、表示する数字の範囲は1～33とする

実行例

演習 2

テンプレートの「index.html」をコピーして、ファイル名「ensyu02.html」に変更しなさい。次に、**let data = [35, 16, 8, 21, 14, 32];** の合計を計算してコンソールに表示するようにしなさい。繰り返しは for … of を使うこと。



実行例