# Webプログラミング
## 第10回

# 前回まで

- 入力に応じて動作を変更する
  - プロンプトの表示
  - 変数
  - 定数
  - if文
  - else if
- 数当てゲームを作る
  - 条件分岐と比較演算子
  - データ型
- 複数の条件を組み合わせる

# 今回の内容

- 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）

# 繰り返し

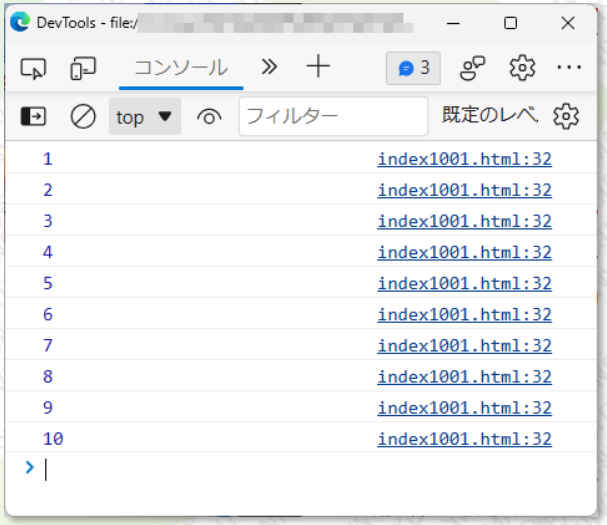
- 繰り返しの処理は大きくわけて２通り
  - 回数が決まっている繰り返し
    - 通常は for文 での処理が適している
  - 回数が決まっていない繰り返し
    - 通常は while文 での処理が適している
  - for文、while文は書き方によってどちらでも処理は可能

# 繰り返し（回数が決まっている）

- 回数が決まっている繰り返しの処理を行う
- コンソールに１枚、２枚、３枚・・・と出力する
  - テンプレートの「index.html」をコピーして、ファイル名を「index1001.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

index1001.html

```
27      </footer>
28      <script>
29          'use script';
30          let i;
31          for(i=0; i<10; i++){
32              console.log(i+1);
33          }
34      </script>
35  </body>
```




コンソールへ出力

# 繰り返し（回数が決まっている）

- 回数が決まっている繰り返しの処理を行う
- for文について

for文の書式

```
for(初期化; 条件式; 更新式){
   //  繰り返す処理
}
```

for文は、条件式が成り立つ間、処理を繰り返す

```
for(i=0;i<10;i++){
   ・・・
}
```

i=0で初期化され、iが10未満の間、ループする
iは、一回のループごとに インクリメント される
※ i は 先に定義する必要あり → let i;

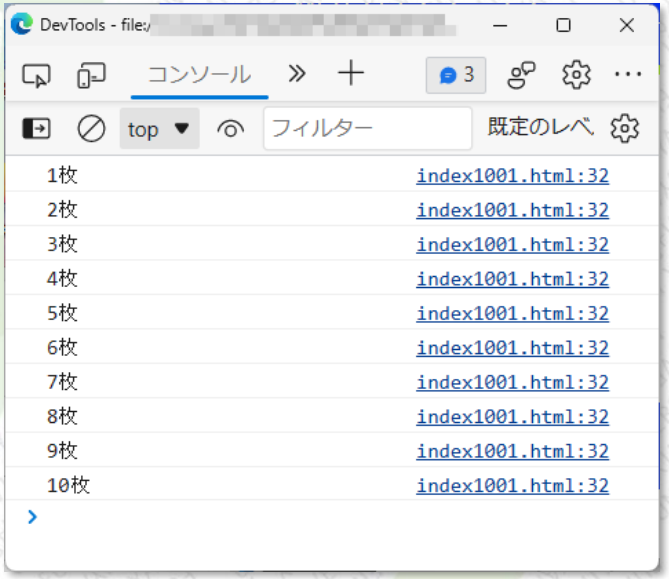
# 繰り返し（回数が決まっている）

- コンソールに１枚、２枚、３枚・・・と出力する
  - 「index1001.html」に下記の変更を加える

index1001.html

```
28          <script>
29              'use script';
30              let i;
31              for(i=0; i<10; i++){
32                  console.log((i+1) + '枚');
33              }
34          </script>
35      </body>
```

コンソールへ出力




- 文字列連結

```
console.log( (i+1) + '枚' );
```

'+'は、数値の足し算の他に、文字列の連結もできる

| プログラム | +の機能 | 結果 |
|---|---|---|
| `console.log(16 + 70);` | 足し算 | 86 |
| `console.log(name + 'さん');` | 文字列連結 | 田中さん |
| `console.log((16 + 70) + '個');` | 足し算、文字列連結 | 86個 |

# 繰り返し（回数が決まっていない）

- 回数が決まっていない繰り返しの処理を行う
- 実行例として、以下のようなルールを設定し、簡単なゲームっぽいものを作ってみる
  - あなたは勇者で、HP100のモンスターがいて、倒さなければいけない
  - あなたの攻撃力は１回につき30P以下、毎回ランダムで決定される
  - 攻撃によりモンスターのHPが減る
  - モンスターのHPが0以下になるまで攻撃を繰り返す

# 繰り返し（回数が決まっていない）

- 回数が決まっていない繰り返しの処理を行う
  - テンプレートの「index.html」をコピーして、ファイル名を「index1002.html」に変更、下記を追記する

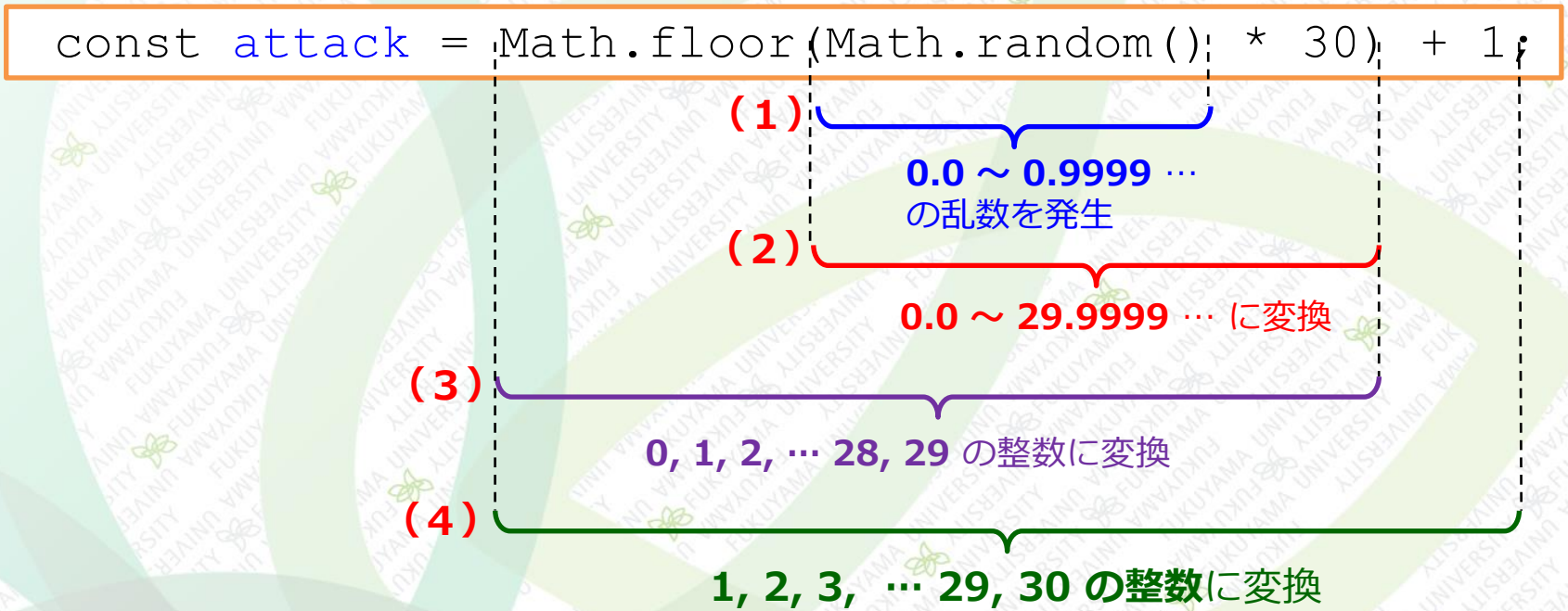
index1002.html

```
27          </footer>
28          <script>
29              'use strict';
30              let enemy = 100;
31              window.alert('戦闘スタート！');
32              while(enemy > 0) {
33                  const attack = Math.floor(Math.random() * 30) + 1;
34                  console.log('モンスターに' + attack + 'のダメージ！');
35                  enemy = enemy - attack;
36              }
37              console.log('モンスターを倒した！');
38          </script>
39      </body>
```

モンスターの体力を変数 enemy、攻撃力を attack とする
attack は、毎回の攻撃時に30以下の乱数が代入される

# 繰り返し（回数が決まっていない）

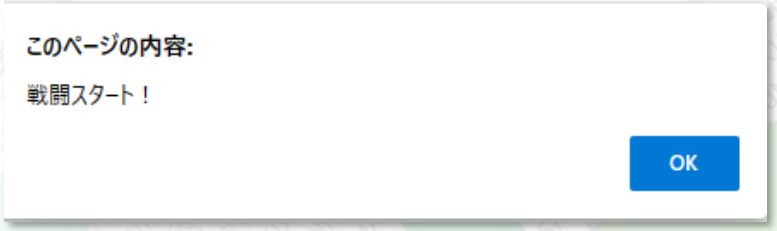
- 回数が決まっていない繰り返しの処理を行う
  - 乱数について
    - 変数（定数）の attack には、以下の（1）～（4）のような流れで 1～30 の乱数（整数）が代入される



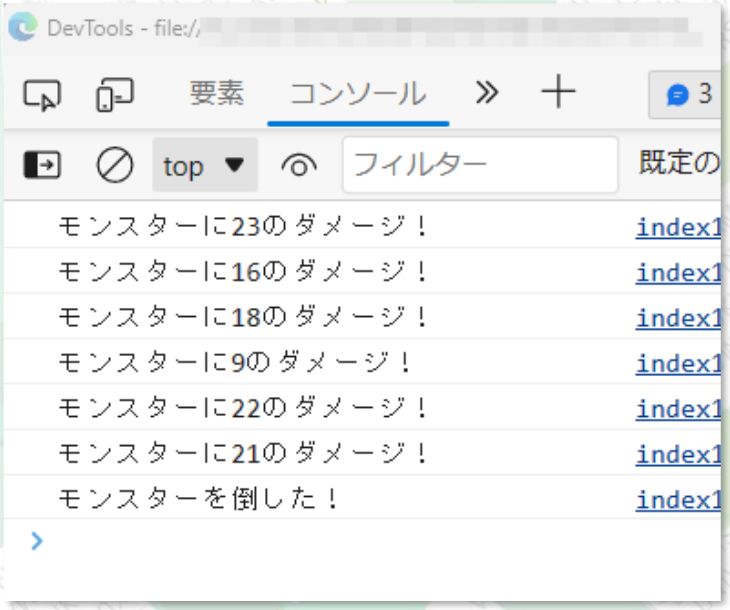

※Math.floor() は、小数点以下を切り捨てて、整数に変換するメソッド

# 繰り返し（回数が決まっていない）

- 回数が決まっていない繰り返しの処理を行う




はじめにアラートダイアログボックスが表示され、[OK]を押すとスタート




戦況がコンソールに表示される

モンスターのHPが減っていき、最後にHPが0(以下)になればゲーム終了

# 繰り返し（回数が決まっていない）

- 回数が決まっていない繰り返しの処理を行う
  - while文について
    - while文のループは、回数が未定のときに適している

while文の書式

```
while( 条件式 ){
   //  繰り返す処理
}
```

while文は、条件式が成り立つ間、処理を繰り返す

```
let i = 0; //カウンタ用変数

while(i<10){
   ・・・
   i++; //iを1増やす
}
```

i=0で初期化され、iが10未満の間ループする
iは、ループの中でインクリメント しなければいけない

# 繰り返し（回数が決まっていない）

- 回数が決まっていない繰り返しの処理を行う
  - while文について

変数enemyにが 0 より大きい間、{ }の中を実行する

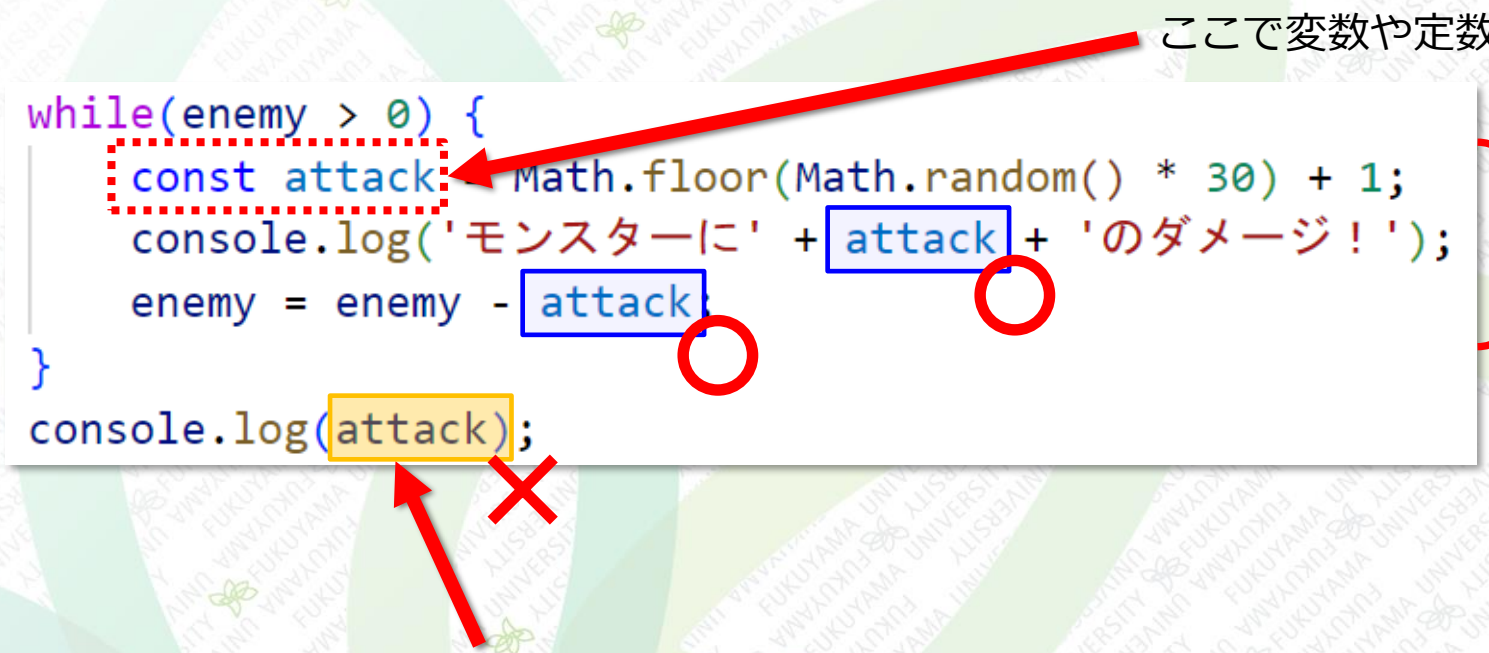
```
while(enemy > 0) {
    const attack = Math.floor(Math.random() * 30) + 1;
    console.log('モンスターに' + attack + 'のダメージ！');
    enemy = enemy - attack;
}
```

「モンスターに」という文字列に続けて、定数 attackに代入されている数値を結合、さらにその後ろに「のダメージ！」を連結したものを表示している

変数enemyに保存されている数値から、定数attackに保存されている数値を引いて、変数enemyに代入しなおしている

# 繰り返し（回数が決まっていない）

- 変数（定数）の有効範囲
  - 変数および定数は、定義した場所によって有効範囲が異なる



# 繰り返し（回数が決まっていない）

- 変数（定数）の有効範囲
  - 変数および定数は、定義した場所によって有効範囲が異なる

```
<script>
    'use strict';
    let enemy = 100;
    window.alert('戦闘スタート！');
    while(enemy > 0) {
        const attack = Math.floor(Math.random()
        console.log('モンスターに' + attack + 'の
        enemy = enemy - attack;
    }
    console.log(enemy);
    console.log('モンスターを倒した！');
</script>
    .
    .
<script>
    console.log('enemyの値:' + enemy);
</script>
```

ここで変数や定数を定義すると
変数の有効範囲は広くなる

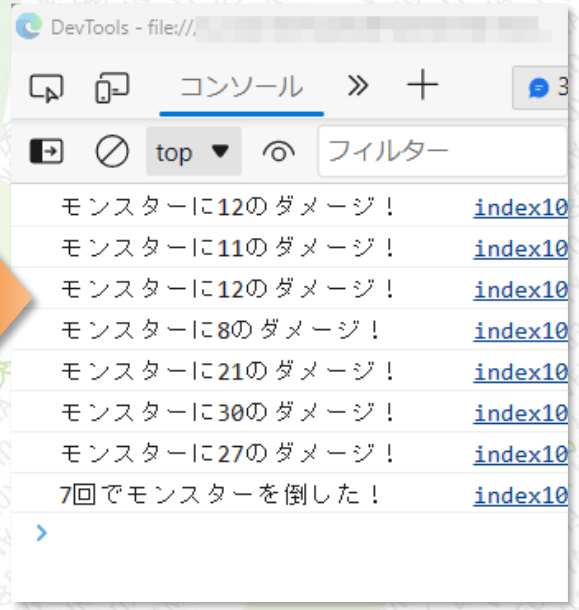
変数の定義場所とは別の
<script>～</script>タグ
に囲まれているところでも
有効範囲になる

# 繰り返し（回数が決まっていない）

- 繰り返しの回数をカウントする
  - while文のループで、繰り返された回数を表示する
  - 「index1002.html」に下記を修正・追記する

**index1002.html**

```
<script>
    'use strict';
    let enemy = 100;
    let count = 0;
    window.alert('戦闘スタート！');
    while(enemy > 0) {
        const attack = Math.floor(Math.random() * 30) + 1;
        console.log('モンスターに' + attack + 'のダメージ！');
        enemy -= attack;
        count += 1;
    }
    console.log(count + '回でモンスターを倒した！');
</script>
```




コンソールに表示

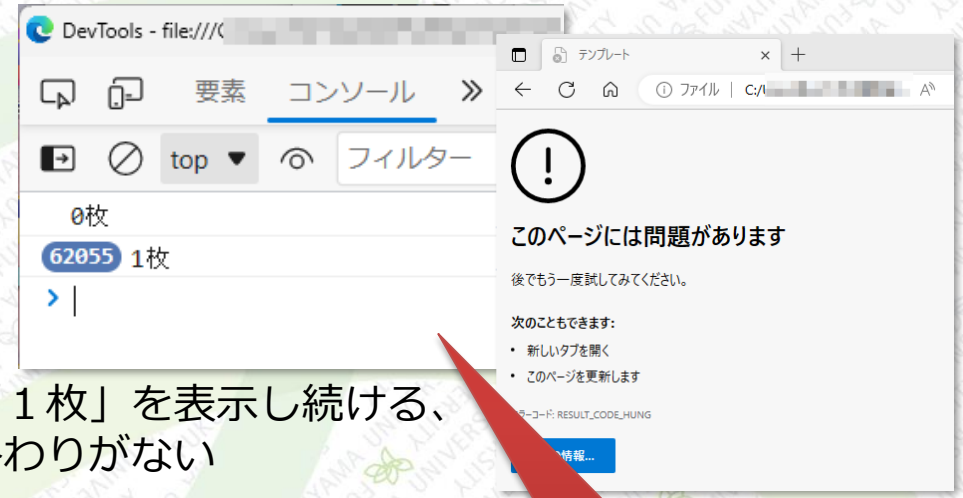
繰り返した回数を保存する変数 let count を新たに追加して、一回の処理ごとに count += 1; で count を 1 ずつ増やす

# 繰り返し（回数が決まっていない）

- 無限ループに要注意
  - 簡単なミスによる、意図しない無限ループに注意する

```
let i = 0;
while(i<10){
    console.log(i + '枚');
    i = 1;
}
```

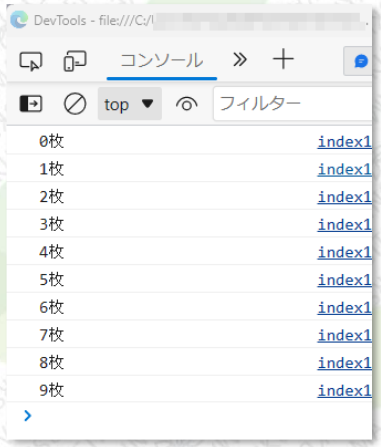
i += 1; を i = 1; と間違えた




「１枚」を表示し続ける、終わりがない

正しくは

```
let i = 0;
while(i<10){
    console.log(i + '枚');
    i += 1;
}
```




こうなってしまうと、タスクマネージャで**ブラウザを強制終了**するしかない

# 関数（ファンクション）

- 関数（ファンクション）
  - 関数はよく使う処理を１つにまとめた小さなサブ・プログラム
  - 他から呼び出して利用する

# 関数（ファンクション）

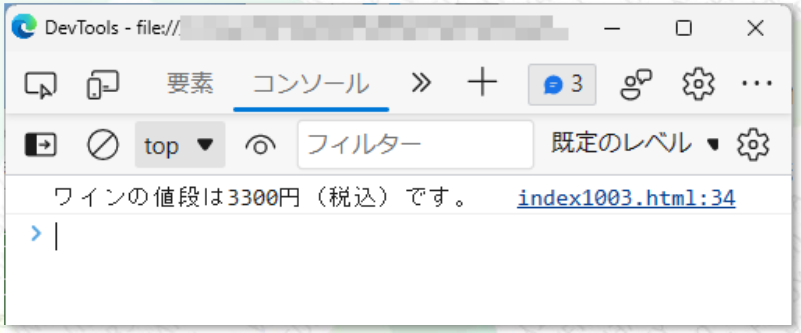
- 税込み価格を計算する
  - 3000円のワインをECサイトで販売する体で消費税込みの価格を計算してHTMLに表示する
  - 消費税込みの価格の計算は、関数で処理する
  - テンプレートの「index.html」をコピーして、ファイル名を「index1003.html」に変更、下記を追記する

index1003.html

```
27          </footer>
28          <script>
29              'use strict';
30              function total(price) {
31                  const tax = 0.1;
32                  return price + price * tax;
33              }
34              console.log('ワインの値段は' + total(3000) + '円（税込）です。');
35          </script>
36      </body>
```

# 関数（ファンクション）

- 税込み価格を計算する
  - 消費税込みの価格がコンソールに出力される




関数呼び出しの流れ

```
<script>
    'use strict';
    function total(price) {
        const tax = 0.1;
        return price + price * tax;
    }
    console.log('ワインの値段は' + total(3000) + '円（税込）です。');
</script>
```

```
'ワインの値段は' + total(3000) + '円
```

関数呼び出し

price に 3000 が代入される

```
function total(price) {
    const tax = 0.1;
    return price + price * tax;
}
```

# 関数（ファンクション）

- 税込み価格を計算する
  - 消費税込みの価格がコンソールに出力される

関数呼び出しの流れ

```
<script>
    'use strict';
    function total(price) {
        const tax = 0.1;
        return price + price * tax;
    }
    console.log('ワインの値段は' + total(3000) + '円（税込）です。');
</script>
```

price に 3000 が入っているので、
計算結果は 3300

```
function total(price) {
    const tax = 0.1;
    return price + price * tax;
}
```

3300 が戻ってくる

```
console.log('ワインの値段は' + total(3000) + '円（税込）です。');
```

# 関数（ファンクション）

## 関数について

### 関数の書式

```
function 関数名(引数){
        :
   //処理の内容
        :
}
```

※引数は複数個の場合、カンマ「,」で区切って記述する

```
function caluc(yen, num){
        :
}
```

### 引数の有効範囲

- 引数の変数の有効範囲は、中カッコ{ }の中のみ

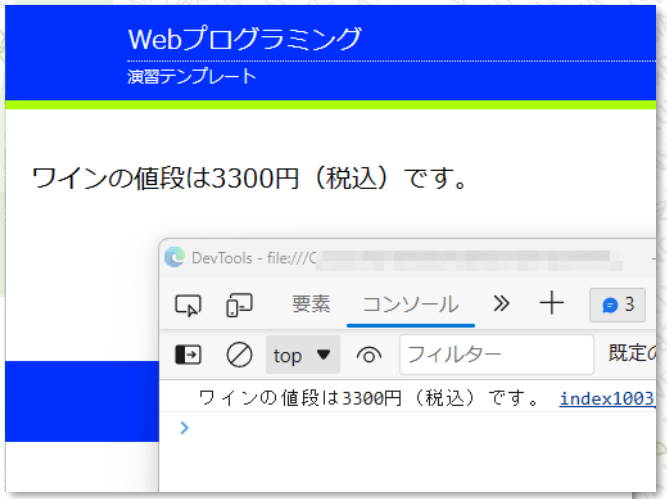
```
function total(price){
        :
   priceはこの中だけで使える
        :
}
```

# 関数（ファンクション）

- HTMLに出力する
  - 税込価格をHTMLに出力する
  - 「index1003.html」に下記を追記・修正する

index1003.html

```
18          <section>
19              <p id="output"></p>
20          </section>
            :
            :
28          <script>
29              'use strict';
30              function total(price) {
31                  const tax = 0.1;
32                  return price + price * tax;
33              }
34              console.log('ワインの値段は' + total(3000) + '円（税込）です。');
35              document.getElementById('output').textContent = 'ワインの値段は'
                + total(3000) + '円（税込）です。';
36          </script>
37      </body>
```

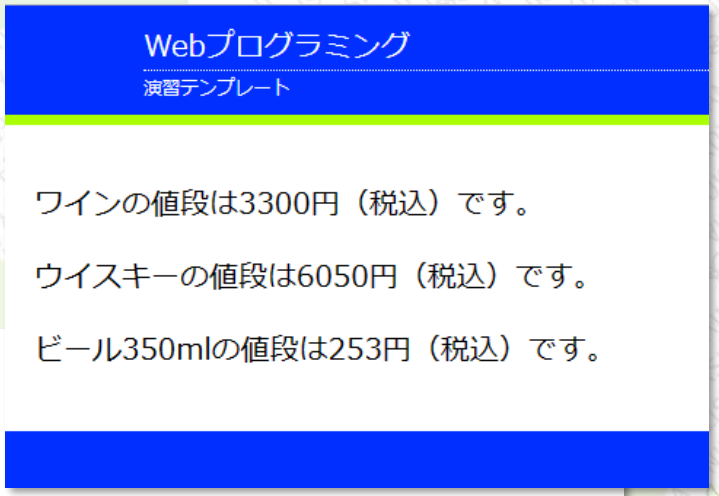


# 関数（ファンクション）

- HTMLに出力する
  - 価格表示を増やす、「index1003.html」に下記を追記する

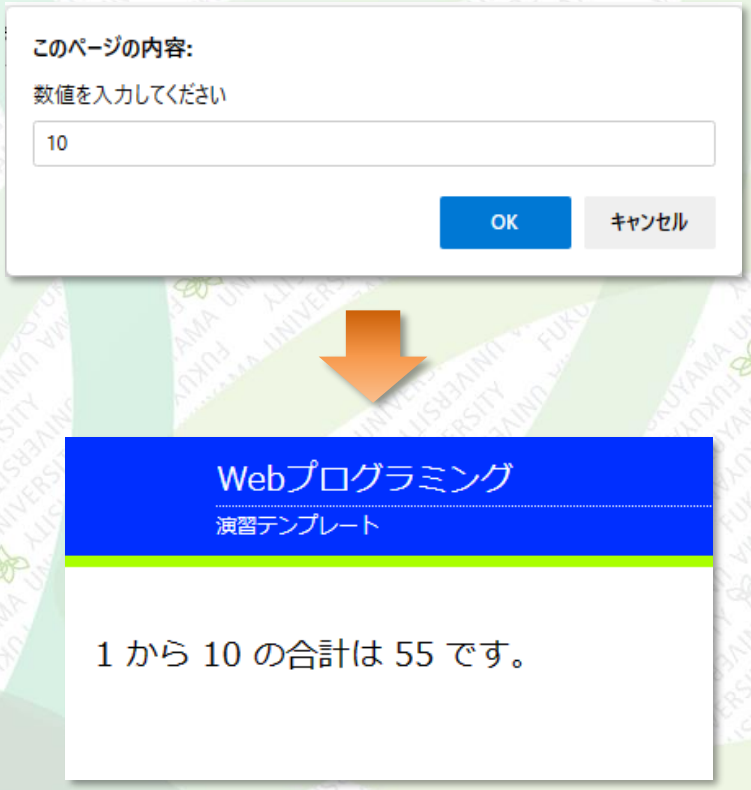
index1003.html

```
18          <section>
19              <p id="output"></p>
20              <p id="output2"></p>
21              <p id="output3"></p>
22          </section>
            ⋮
32          function total(price) {
33              const tax = 0.1;
34              return price + price * tax;
35          }
36          console.log('ワインの値段は' + total(3000) + '円（税込）です。');
37          document.getElementById('output').textContent = 'ワインの値段は'
            + total(3000) + '円（税込）です。';
38          document.getElementById('output2').textContent = 'ウイスキーの値
            段は' + total(5500) + '円（税込）です。';
39          document.getElementById('output3').textContent = 'ビール350mlの
            値段は' + total(230) + '円（税込）です。';
40      </script>
```



# 演習１

まずはじめに、テンプレートの「index.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）。ブラウザで「ensyu01.html」を読み込むと、プロンプトを表示して１以上の数値(ここではNとする)を入力し、１からNまでの合計をHTMLに表示するようにしなさい。表示の書式などは、実行例と同じにすること。




実行例

# 演習２

演習１の「ensyu01.html」をコピーして、ファイル名「ensyu02.html」に変更しなさい。次に、１からｎまでの合計を返す関数 goukei(n) を作成して、この関数を呼び出して処理するように修正しなさい。


実行例