# Webプログラミング
## 第９回

# 前回まで

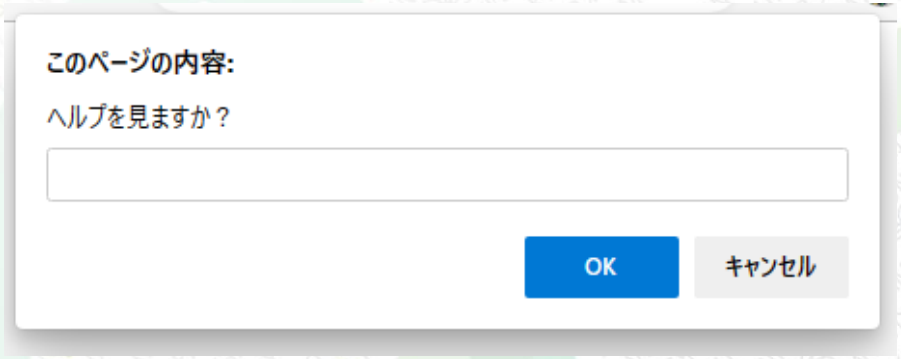
- JavaScriptとは
- コンソールにアウトプット
- JavaScriptを記述する場所
- ダイアログボックスを表示する
- HTMLを書き換える
- JavaScriptの基本的な機能
  - 確認ダイアログボックス、if文

# 今回の内容

- 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

# 入力に応じて動作を変更する

- テキストフィールドを持つダイアログボックスである「プロンプト」を使う



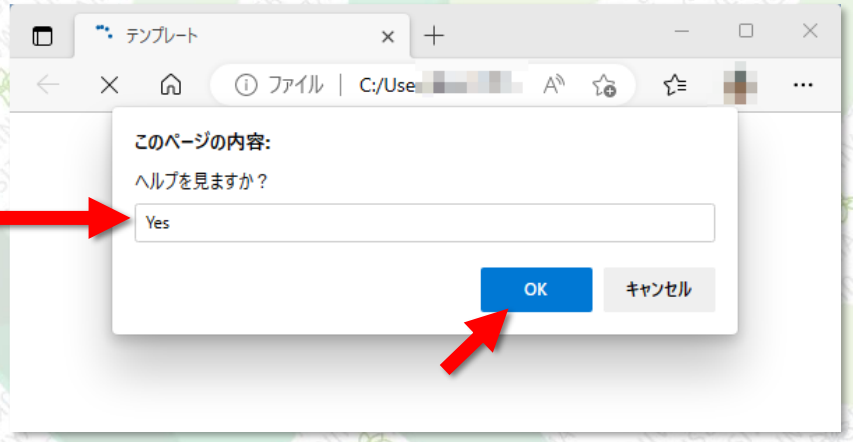

- プロンプトは入力したテキストをリターンとして返す
- リターンを変数に保存して様々な処理を行う

# 入力に応じて動作を変更する

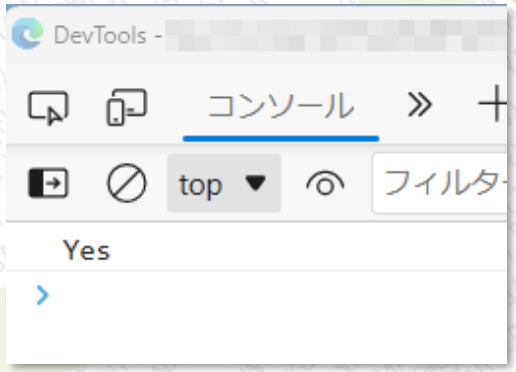
- テキストを入力して結果を確認する
  - テンプレートの「index.html」をコピーして、ファイル名を「index0901.html」に変更、下記を追記する
  - テンプレートの「style.css」もコピーする

**index0901.html**

```
27          </footer>
28          <script>
29              'use strict';
30
31              let answer = window.prompt('ヘルプを見ますか？');
32              console.log(answer);
33          </script>
34      </body>
```




「**Yes**」と入力



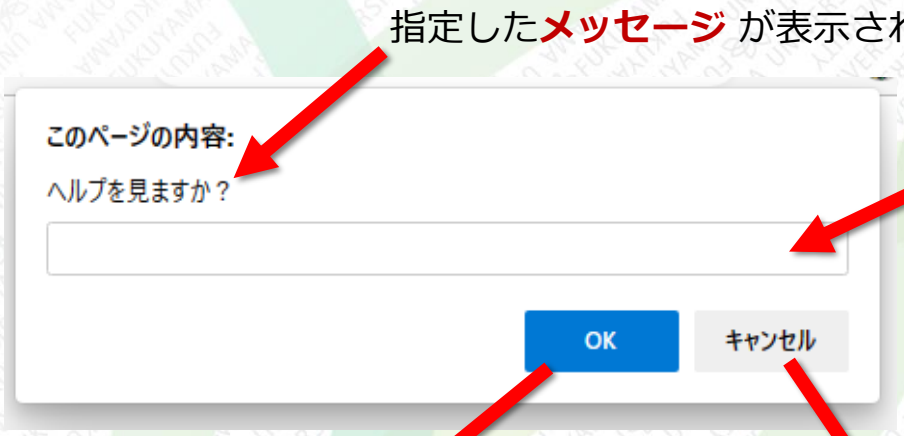

コンソールへ出力

# 入力に応じて動作を変更する

## プロンプトの表示

プロンプト表示の書式

```
変数 = window.prompt(メッセージ);
```




```
let answer = window.prompt('ヘルプを見ますか？');
```

# 入力に応じて動作を変更する

- 変数について
  - JavaScriptでは、変数を定義する必要がある

変数の定義（変数名 val を定義）

```
let val;
val = 'こんにちは';
```

変数の定義と値の代入を同時に行ってもOK

```
let val = 'こんにちは';
```

変数の定義は初めに１度だけでよい

```
let val = 'こんにちは';
console.log(val);
val = 'さようなら';
console.log(val);
```

変数の基本的な型は
数値型 と 文字列型

数値型は、int型 や float型、double型のような細かい区別はない

数値も扱える

```
let a = 3;
let b = 5;
let c = a + b;
```

# 入力に応じて動作を変更する

- 変数名のつけ方
  - 変数名の条件
    - アルファベットの大文字・小文字は区別する
    - 半角英数字、日本語の漢字やカタカナ、ひらがな は**OK**
    - アンダースコア(_)、ドルマーク($) は**OK**
    - 上記以外の記号(「-」「=」など)は使えない
    - １文字目に数字は使えない
    - 予約語は使えない

予約語一覧

| | | | | | | |
|---|---|---|---|---|---|---|
| break | case | catch | class | continue | debugger | default |
| delete | do | else | enum | export | extends | finally |
| for | function | if | implements | import | in | instanceof |
| interface | let | new | package | private | protected | public |
| return | static | super | switch | this | throw | try |
| typeof | var | void | while | with | yield | |

# 入力に応じて動作を変更する

- 変数の定義について
  - 変数の定義は var も使える

どちらでもOK

```
let answer = window.prompt('ヘルプを見ますか？');
```

```
var answer = window.prompt('ヘルプを見ますか？');
```

  - しかし、var はもう古いかも・・
    - var は、プログラムの規模が大きくなると、varの性質が原因でエラーが生じやすい点が問題になっていた
    - JavaScriptの新しいバージョンES6以降では、let や const が新しく登場した、今後はこちらが主流になる

再定義と再代入の可否

| | var | let | const |
|---|---|---|---|
| 再定義 | できる | できない | できない |
| 再代入 | できる | できる | できない |

定数を定義

# 入力に応じて動作を変更する

- 定数
  - 値を更新しない変数は定数にした方がよい
  - 定数として定義すると、最初に代入した値は変更できない
    - 「index0901.html」を下記のように修正して再読み込みしてみる

index0901.html

```
27      </footer>
28      <script>
29          'use strict';
30
31          const answer = window.prompt('ヘルプを見ますか？');
32          console.log(answer);
33          answer = 'no';
34          console.log(answer);
35      </script>
36  </body>
```


ページを再読み込みすると、コンソールにエラーが表示される
「定数を書き換えようとしてるけど、それはできないよ」のような意味

# 入力に応じて動作を変更する

- 定数
  - 値を書き換えようとしているところを削除する
    - 「index0901.html」を下記のように修正して再読み込みしてみる

index0901.html

```
27          </footer>
28          <script>
29              'use strict';
30
31              const answer = window.prompt('ヘルプを見ますか？');
32              console.log(answer);
33              answer = 'no';
34              console.log(answer);
35          </script>
36      </body>
```

削除

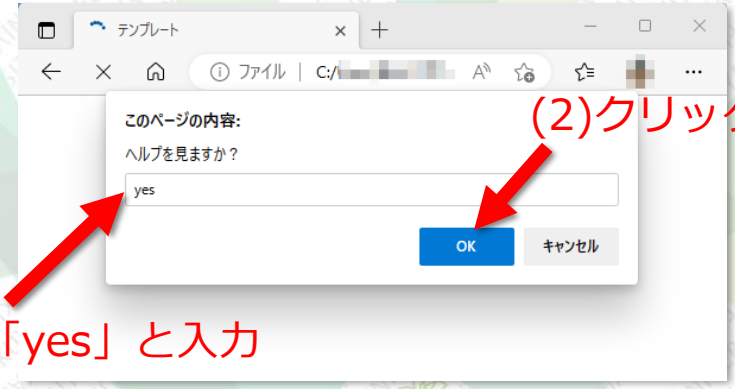
エラーは表示されない

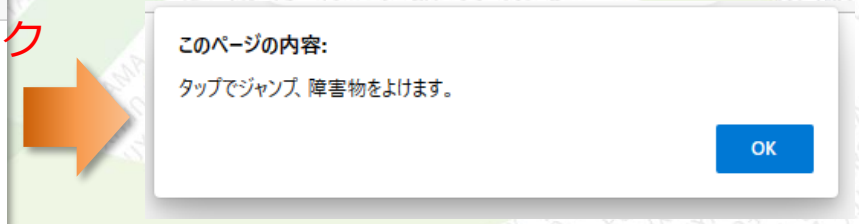
# 入力に応じて動作を変更する

- if文で動作を切り替える
  - 入力されたテキストの内容によって動作を切り替える
  - 「index0901.html」を下記のように修正して再読み込みしてみる

index0901.html

```
27          </footer>
28          <script>
29              'use strict';
30
31              const answer = window.prompt('ヘルプを見ますか？');
32              if(answer === 'yes') {
33                  window.alert('タップでジャンプ、障害物をよけます。');
34              }
35          </script>
36      </body>
```




[キャンセル]は何も起こらない

# 入力に応じて動作を変更する

- if文について
  - 条件式の書き方

変数 answer の内容が ‘yes’ に等しいとき・・

```
if( answer === ‘yes’ ){
```

「===」は比較演算子、左右の値が同じであれば評価結果が ture になる

「=」の場合(‘=‘が 1 個)は、代入になるので比較にならない

```
if( a = b ){
```

×

「==」の書き方(‘=‘が 2 個)も使えるが、注意が必要

```
if( a == b ){
```

△

動作のしくみを正確に把握していないと、予想外の結果になるリスクがある

「===」… 左右の値を型変換せずに、厳密に等しいかどうかを調べる

「==」… 左右の値がなんとかして同じものに見えないか、JavaScriptが試行錯誤して比べる（左右値を型変換して、評価結果が ture にならないかを努力して調べる？）

# 入力に応じて動作を変更する

- if文の分岐を増やす(else if)
  - else if で分岐を増やし、動作のバリエーションを追加する
  - 「index0901.html」を下記のように追記・修正する

index0901.html

```
28          <script>
29              'use strict';
30
31              const answer = window.prompt('ヘルプを見ますか？');
32              if(answer === 'yes') {
33                  window.alert('タップでジャンプ、障害物をよけます。');
34              } else if(answer === 'no') {
35                  window.alert('ゲーム起動中...');
36              } else {
37                  window.alert('yesかnoでお答えください。');
38              }
39          </script>
40      </body>
```

※ else if の使い方は、JavaやC言語などの他の言語とほぼ同じ

# 入力に応じて動作を変更する

- if文の分岐を増やす(else if)
  - 入力された文字が「yes」、「no」、それ以外で表示メッセージが変わる

# 数当てゲームを作る

- 数当てゲームを作ってみる
- 数値を入力し、あらかじめ用意しておいた数(乱数で生成)と比較して大小を評価する
- 結果はダイアログボックスに表示する
  - 同じであれば「当たり！」
  - 答えの方が大きい場合「残念！もっと大きい」
  - 答えの方が小さい場合「残念！もっと小さい」

# 数当てゲームを作る

- テンプレートの「index.html」をコピーして、ファイル名を「index0902.html」に変更して下記を追記する

index0902.html

```
27          </footer>
28          <script>
29              'use strict';
30
31              const number = Math.floor(Math.random() * 6);
32              const answer = parseInt(window.prompt('数当てゲーム/0～5の数字を入力してね'));
33          </script>
34      </body>
```

変数 number に0～5の乱数（ここでは整数）が代入される（0,1,2,3,4,5のいずれか）

```
const number = Math.floor(Math.random() * 6);
```

文字列の数字を、整数に変換する

```
parseInt(変換したい文字列);
```

例えば、文字の ‘3’ が 数値の 3 になる

# 数当てゲームを作る

- 続けて「index0902.html」に下記を追記する

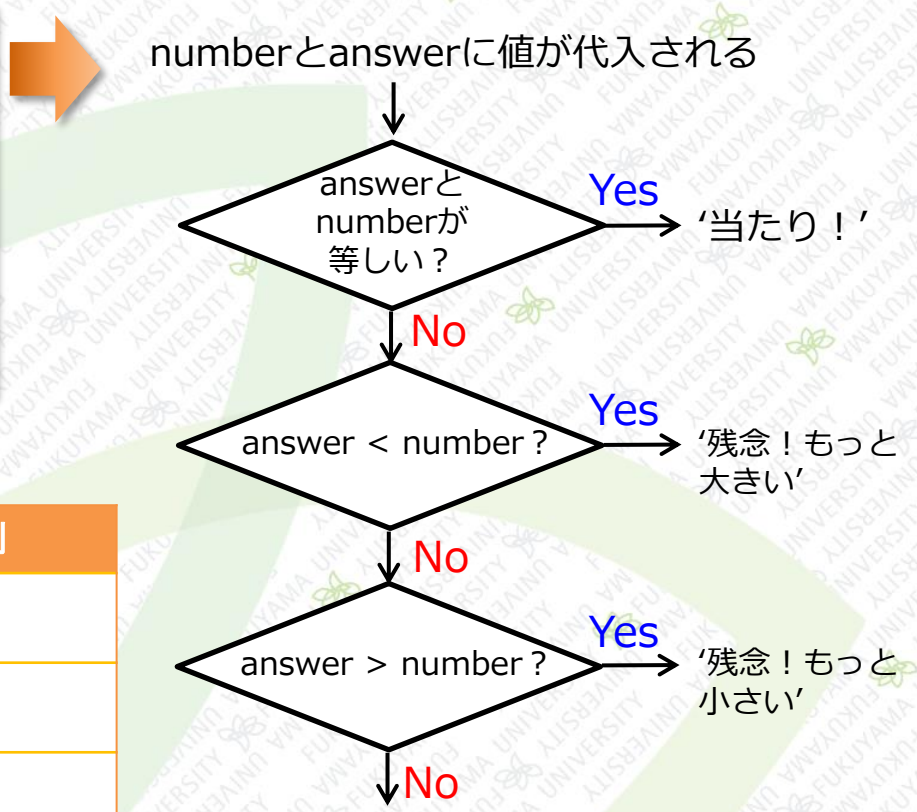
index0902.html

```
28          <script>
29              'use strict';
30
31              const number = Math.floor(Math.random() * 6);
32              const answer = parseInt(window.prompt('数当てゲーム/0～5の数字を入力してね'));
33              let message;
34              if(answer === number) {
35                  message = '当たり！';
36              } else if(answer < number) {
37                  message = '残念！もっと大きい';
38              } else if(answer > number) {
39                  message = '残念！もっと小さい';
40              } else {
41                  message = '0～5の数字を入力してね';
42              }
43              window.alert(message);
44              window.alert('答えは '+ number + 'でした');
45          </script>
46      </body>
```

# 数当てゲームを作る

## 条件分岐の流れと比較演算子

```
if(answer === number) {
    message = '当たり！';
} else if(answer < number) {
    message = '残念！もっと大きい';
} else if(answer > number) {
    message = '残念！もっと小さい';
} else {
    message = '0～5の数字を入力してね';
}
```



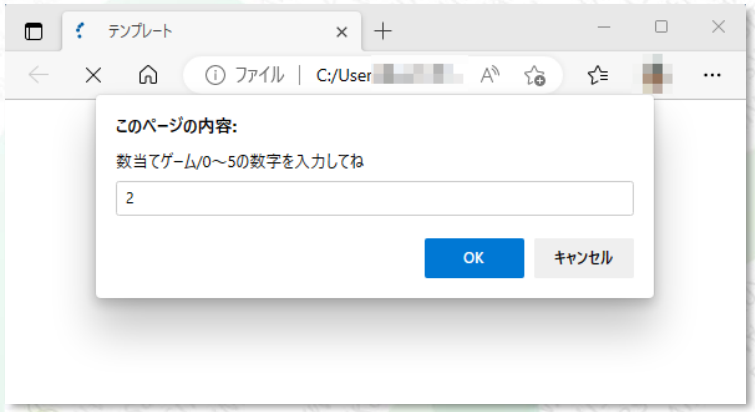

**<分岐のフローチャート>**

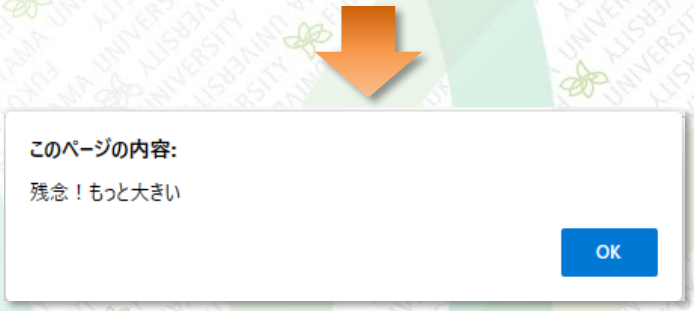
<比較演算子一覧>

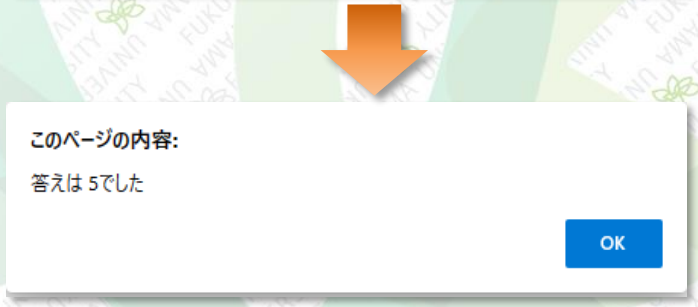
| 演算子 | 意味 | tureになる例 |
|---|---|---|
| a === b | aとbが同じとき true | ‘abc’ === ‘abc’<br>3 + 6 === 9 |
| a !== b | aとbが異なるとき true | ‘abc’ !== ‘123’<br>40 + 6 !== 42 |
| a < b | aがbより小さいとき true | 7 * 50 < 365 |
| a <= b | aがb以下のとき true | 3 * 5 <= 21<br>3 * 7 <= 21 |
| a > b | aがbより大きいとき true | 15 * 4 > 45 |
| a >= b | aがb以上のとき true | 4 * 60 >= 180<br>1 + 2 >= 3 |

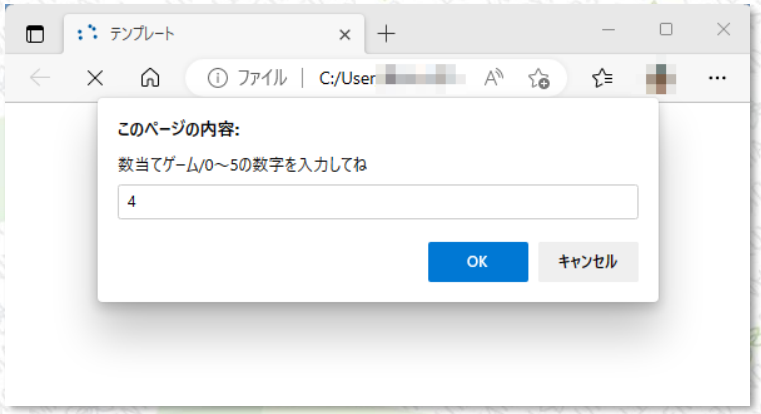
# 数当てゲームを作る

- 実行例

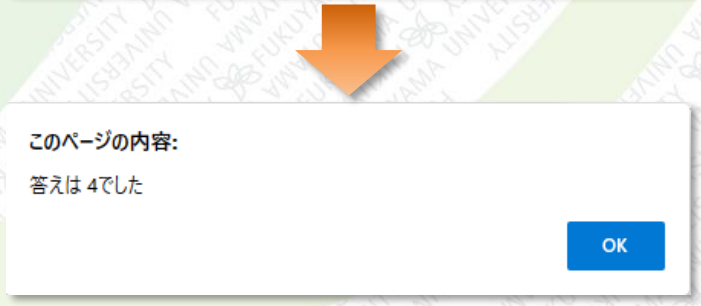






# 数当てゲームを作る

- データとデータ型
  - JavaScriptで扱うデータは、文字列や数値、ブール値などがある
  - これらのデータの種類を「データ型」(略して型)と呼ぶ
  - 各データ型の処理には特徴がある
    - 数値と数値は足し算などの計算ができるが、文字列はできない
    - 数値と数値は、大小を比較できるが、文字列はできない
    - 文字列と文字列は連結できるが、数値と数値はできない

```
const answer = parseInt(window.prompt('数当てゲーム/0～5の数字を入力してね'));
```

ここで得られるのは、文字列の数字

parseInt()により、文字列を数値に変換

answerには、数値が代入される

# 複数の条件を組み合わせる

- 時刻を入力して、値によって表示を切り替える
- 以下のように学校のスケジュールを表示する
  - 時刻は 0時から23時 の 0～23 を入力
  - 8：通学の移動
  - 9～11：午前の授業
  - 12：昼休み
  - 13～16：午後の授業
  - 17：通学の移動
  - 上記以外：学校にいない

# 複数の条件を組み合わせる

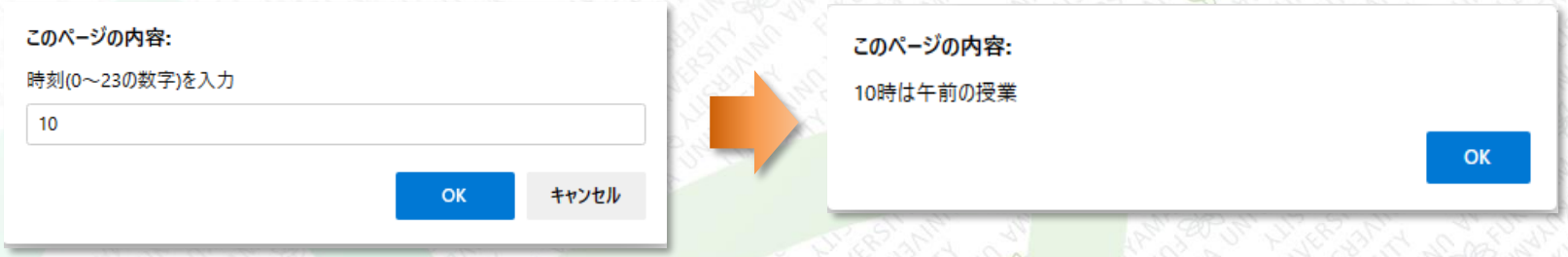
- テンプレートの「index.html」をコピーして、ファイル名を「index0903.html」に変更して下記を追記する

index0903.html

```
27          </footer>
28          <script>
29              'use strict';
30
31              const time = parseInt(window.prompt('時刻(0～23の数字)を入力'));
32              let message;
33              if( time === 8 || time === 17 ) {
34                  message = '通学の移動';
35              } else if( 9 <= time && time < 12 ) {
36                  message = '午前の授業';
37              } else if( time === 12 ) {
38                  message = '昼休み';
39              } else if( 13 <= time && time <= 16 ) {
40                  message = '午後の授業';
41              } else {
42                  message = '学校にいない';
43              }
44              window.alert(time + '時は' + message);
45          </script>
46      </body>
```

# 複数の条件を組み合わせる

- ブラウザで動作を確認してみる




- &&演算子（～かつ～, 論理積, and）

```
if( 9 <= time && time < 12 ){
```

timeが9以上 **かつ** timeが12未満のとき

- ||演算子（～または～, 論理和, or）

```
if( time === 8 || time === 17 ){
```

timeが8に等しい **または** timeが17に等しいのとき

＜論理演算子一覧（A、Bは条件式）＞

| 演算子 | 意味 |
|---|---|
| A && B | AとBが 両方true のとき 全体の評価結果が true |
| A \|\| B | AとBの 少なくともどちらか１つがtrue のとき 全体の評価結果が true |
| ! A | Aが trueでない とき 評価結果が true |

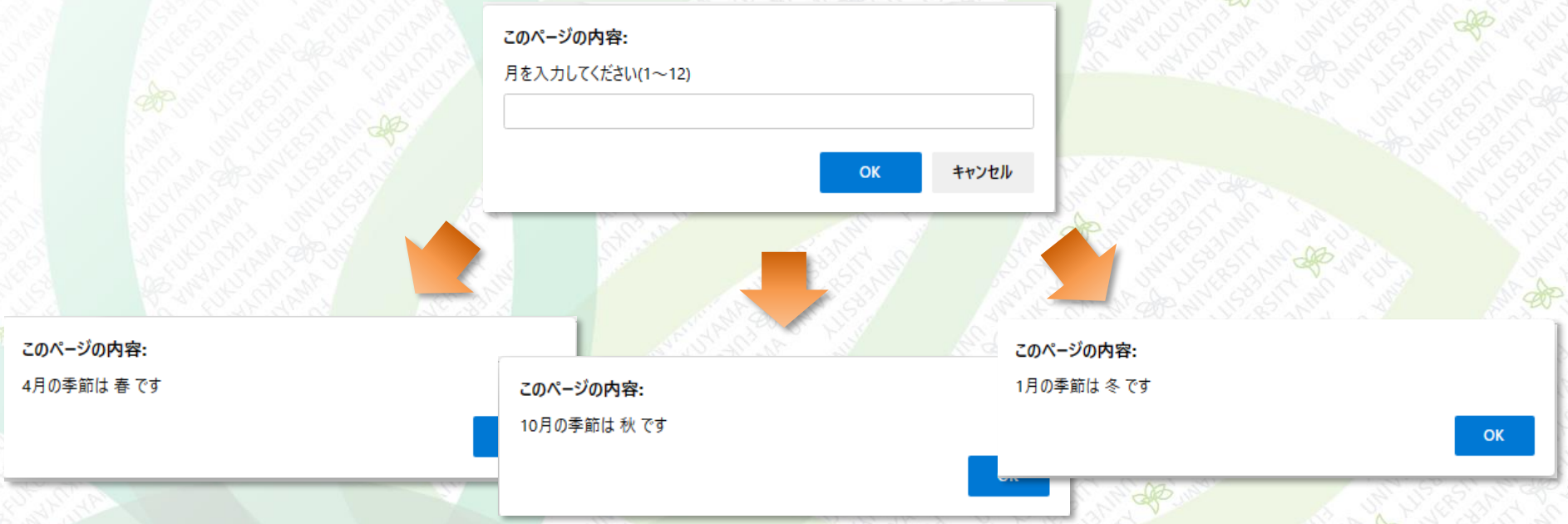
# 演習１

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）。実行例のようなプロンプトを表示して、ユーザが0～100の点数を入力すると、その成績評価（秀・優・良・可・不可）が表示されるように「ensyu01.html」を修正しなさい。点数の区分は
　秀：100～90、優：89～80、良：79～70、可：69～60、不可：59以下とする。


実行例

# 演習２

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「ensyu02.html」に変更しなさい。何月かの数値を入力して、その月の季節（春・夏・秋・冬）の表示をしたい。プロンプトを表示して、ユーザが1～12の数値を入力すると、その季節が表示されるように「 ensyu02.html 」を修正しなさい。季節は
　春：3, 4, 5月、夏：6, 7, 8月、秋：9, 10, 11月、冬：12, 1, 2月
とする。




実行例