JavaScript入門
2.4~2.6

10NM733X　林華

2-4 制御構造

- if
- for
- while
- break,continue
- try～catch

2-4-1 if文

- if(条件式) ステートメント;
- if(条件式){

ステートメント１;

・・・

}

- if(条件式) ステートメント１;else ステートメント２;
- if(条件式){

ステートメント１;

・・・

} else{

ステートメント１;

・・・

}

- 条件式？式A：式B　　（条件演算子というif～else～文の特殊な形式）

条件式の結果が0,null,undefined,"" 以外はすべて「真」である

2-4-2 for文

- for （初期条件処理式；条件式；再処理式）ステートメント；
- for （初期条件処理式；条件式；再処理式){
    ステートメント１；
    ・・・
  }
- for(変数名 in オブジェクト){
    ステートメント；
    ・・・
  }　//オブジェクトのプロパティを順に変数名に格納

2-4-3 while文

- while（条件式）ステートメント；
- while（条件式）{

    ステートメント１；

    ・・・・・・

}

while文は、ループの最初に「条件式」を判定し、真である間はループを実行し続ける。

2-4-4 break & continue

- break : 一番近いfor,while文などのループから抜ける。

```
var i = 0;
while (1){
    i++;
    if (i==5) break;
}
```

- Continue : ループの途中で作業を中断し、強制的にループの次の繰り返しを開始させる。

```
for (i=1; i<max; i++){
  if (i%2) continue;
  document.write(i,"<br>");
}
```

2-4-5 try～catch

- try {
  ステートメント;
  ・・・
  }catch (e){
  ステートメント;
  ・・・
  }
- 例

```
try {
  var today = new Date();
  today.setmonth(3);
}catch(e){
  document.write(e);
}
```

2-5-1 配列とは

- 一つの変数名に対して複数の領域を用意
- 配列にアクセスには、**配列名[添字]**と指定
- 配列の添字は「0」から
  - DayOfTheWeek[2] ←3番目の要素

2-5-2 配列を生成するArrayコンストラクタ

- Arrayコンストラクタ

```
var dayOfTheWeek = new Array(7);
  dayOfTheWeek[0] = “Sunday”;
  dayOfTheWeek[1] = “Monday”
var name = new Array(“山田”,“田中”);
```

- JavaScriptの要素数は可変

```
var name = new Array(2);
    name[0]=″山田”;name[1]=″田中”;name[2]=″貴方”
```

- 各要素には別のデータ型の値を格納できる

```
    var customer = new Array(2);
    customer[0] = 10;
    customer[1] = “you”;
```

- 配列リテラル

```
var name = new Array(“山田”,“田中”);
var name = [“山田”,“田中”];
```

2-5-3 ユーザ定義コンストラクタによる配列オブジェクトの作成

- ユーザが独自にコンストラクタを定義し、配列オブジェクトを作成することができる
- 例:1000個要素を持つ配列を生成し、要素の値を100に初期化する。

```
function MakeArray(len){
        this.length = len;
        for (i=0; i<this.length; i++)
            this[i] = 100;
 }
 var myArray = new MakeArray(1000);
```

2-5-4 配列の並べ替え

- メソッド：sort()
- 構文：配列名.sort(比較関数)
- 注：比較関数が省略された場合、配列の要素値が数値であっても、文字列としてソートされる

```
var numArray = new Array(5,50,9,10,5);
numArray.sort();  <- 10,5,5,50,9


function compare(a, b){ //昇順
  return a-b;
}
var numArray = new Array(5,50,9,10,5);
numArray.sort(compare);    <- 5,5,9,10,50
```

2-5-5 Arrayオブジェクトのメソッド

- toString() 要素をカンマ[,]で繋げた文字列にして戻す
- join()　　要素を繋げた文字列を戻す
- reverse()　要素を逆順にする
- concat()　配列に要素を追加して新たな配列を生成する
- slice()　　配列一部を取り出した新たな配列を生成する
- push()　　配列の最後に要素を追加する
- pop()　　配列の最後に要素を取り出す
- shift()　　配列の最初に要素を追加する
- unshift()　配列の最初に要素を取り出す

「配列名.メソッド名」のように利用する

2-5-6 連想配列

- 普通の配列

  数値である「添字」で配列をアクセス。

  「添字」は配列の要素値とは関係無い。

- 連想配列

  「添字」の代わりに数値または文字列を使う。ここで「キー」という。

  この数値または文字列は配列の要素の値とは関連性を持つ

- 連想配列使用例

```
var tells = new Array(3);
var name;
tells[“田中一郎”]=”03-1111-xxxx”;
tells[“田中三郎”]=”03-3333-xxxx”;
tells[“鈴木一郎”]=”03-5555-xxxx”;
for (name in tells){
     document.write(name, “: “, tells[name], “<br>”);
}
```

2-6-1 関数の定義

- function 関数名(引数１,引数２,…){
  ステートメント１;
  ステートメント２;
  …
  return 戻り値;
  }

- 関数は通常HTMLドキュメントのヘッダ部に記述

2-6-2 変数の有効範囲―「スコープ」

- ローカル変数：プログラムの一部分で有効な変数
- グローバル変数：プログラム全体を通して有効な変数
- 関数内で定義した変数は、その関数の中だけで有効なローカル変数になり、関数の外側で定義した変数はグローバル変数となる。
- 同じ名前のグローバル変数とローカル変数がある場合、関数の内部ではローカル変数の方が有効となる

2-6-3 引数を格納するarguments配列

- arguments配列のそれぞれの要素には渡された引数が順に格納される。
- arguments.lengthには渡された引数の実際の数が格納される。
- 例：渡された引数の総和を求める関数

```
function sum(){
  var result = 0;
  for (var i=0; i<arguments.length; i++){
    result += arguments[i];
  }
  return result;
}
result = sum(10,100,200,300);
sumValue = sum(1,2,3);
```

2-6-4 関数の再帰呼び出し

- 関数の再帰呼び出し：関数の内部で自分自身を呼び出すこと
- 例：自然数の階乗を求める関数

```
function fact(n){
  if (n < 1) return null;
  if (n > 1){
    return (n * fact(n-1));
  } else {
    return 1;
  }
}
var num = 10;
document.write("10の階乗は", fact(10), "です");
```

2-6-6 イベント処理の基礎知識-1

- JavaScriptでは、必ずしもスクリプトを予め決められた順序で実行し続ける必要はない。
- ユーザがボタンをクリックしたりするようなイベントに応じて、特定な処理をさせることが可能。
- イベントを捕まえて処理を行う関数のことを「イベントハンドラ」と呼ぶ。
- 例：

  HTMLドキュメントロード完了→onloadイベントハンドラ

  ボタンクリック→onclickイベントハンドラ

  テキスト選択された→onselectイベントハンドラ

- 記述方：
  - HTMLタグのアトリビュートに記述：イベントハンドラ名＝“実行するステートメント”

    ```
    <form>
      <input type="button" value="Handler1"
      onclick="window.alert('ハンドラのテストです')">
    </form>
    ```

    または onclick="count++; window.alert('ハンドラのテストです')">
  - スクリプトに記述し、イベントハンドラをプロパティのように使用

    ```
    document.myForm.myBtn.onclick = func2;
    ```

2-6-6 イベント処理の基礎知識-2

- イベントハンドラの種類

onclick	オブジェクトがクリックされたとき
ondbclick	オブジェクトがダブルクリックされたとき
onmousedown	マウスボタンを押したとき
onmouseup	マウスボタンを離したとき
onmousemove	マウスボタンを動かしたとき
onchange	テキストの内容が変更されたとき
onsubmit	フォームが送信されたとき
onmouseover	オブジェクトにマウスが入ったとき
onmouseout	オブジェクトからマウスが出たとき
onload	ドキュメントがロードされたとき
onreset	フォームをリセットしたとき
onerror	エラーが起こったとき
onselect	テキストが選択されたとき

今週の課題ー計算ゲーム

- ブラウザにランダムに1～13の数字を4つ生成
- 入力フォームと送信ボタンと答えボタンを生成
- ルール：２＊（１２－３）＋６＝２４のように、4つの数字を四則演算(＋－＊/)を用いて、結果が２４になれるよう計算すること。数字の順位に制限がない。つまり、６＋（１２－３）＊２＝２４も大丈夫。
- 答えボタンを押せば、解答が出る
- 間違った解答にメッセージを出す。