JavaScript基礎-第5回資料

※Arrayの各種メソッド、Objectのクラス的使用法、prototypeについては次回以降で解説予定

◇配列

例1

```
1  var arrA = new Array();//空配列、長さ0
2  arrA.length = 3;//長さ3に設定（自動拡張されるので実際は不要）
3  arrA[0]= 0;
4  arrA[1]= 1;
5  arrA[2]= 2;
6  var arrB = new Array(3);//空配列、長さ3（自動拡張されるので実際は不要）
7  arrB[0]= 0;
8  arrB[1]= 1;
9  arrB[2]= 2;
10 var arrC = new Array(0,1,2);//0,1,2の配列、長さ3
11 var arrD =[];//空配列、長さ0
12 arrD.length = 3;//長さ3に設定（自動拡張されるので実際は不要）
13 arrD[0]= 0;
14 arrD[1]= 1;
15 arrD[2]= 2;
16 var arrE =[0,1,2];//0,1,2の配列、長さ3
```

配列とは変数が箱だとすればそれを1個以上つなげた物である
<例1>ではいずれの変数も、内容が0,1,2で長さ3の配列（3つの変数が一まとめになっている）になっている
"配列名.length"はその配列の長さを得ること・変更することができる
"print(配列名.join(','))"としてみれば配列の内容が確認できるはずだ

配列への値の代入は3-5行目や7-9行目や13-15行目のように1つずつ行うか、10行目や16行目のようにもともと配列作成時に値を入れておくことによって行う
内容の参照は"配列名[添え字]"で行い、ここでの添え字とは配列の何番目ということであるが、添え字は0から始まる点に注意が必要である

例2

```
1  var arr =[];
2  arr["apple"]= "りんご";
3  arr["banana"]= "バナナ";
4  arr["orange"]= "オレンジ";
5  print(arr["orange"]);
```

さらに、JavaScriptでは添え字を文字列にすることもできる
これをハッシュ配列や連想配列と呼ぶことがある

◆Object

例3

```
1  var objA = new Object();
2  objA.apple = "りんご";
3  objA.banana = "バナナ";
4  objA.orange = "オレンジ";
5  objA["apple"] = "りんご";
6  objA["banana"] = "バナナ";
7  objA["orange"] = "オレンジ";
8  var objB ={apple:"りんご",banana:"バナナ",orange:"オレンジ"};
9  objB ={"apple":"りんご","banana":"バナナ","orange":"オレンジ"};
10 print(objA.apple);
11 print(objA["apple"]);
12 print(objB.apple);
13 print(objB["apple"]);
```

Objectとは他言語のクラスのようなものであり、2-4行目のように"オブジェクト名.プロパティ名=値"のようにして内部に値を保持するか、objBのようにして初期化時にまとめて値を設定できる
もちろん、"obj={}"のように宣言しておき、後からプロパティを追加してもよい
変数がデータを入れておく箱だとすれば、オブジェクトは箱（変数）を格納する名前つきの引き出しと考えることができる
データの参照は連想配列のように行うか、"オブジェクト名.プロパティ名"で行う
なお、2-4行目と5-7行目、8行目と9行目の意味は同じである

例4

```
1  var arr =[];
2  arr["apple"] = "りんご";
3  arr["banana"] = "バナナ";
4  arr["orange"] = "オレンジ";
5  var obj ={
6        apple:"りんご",
7        banana:"バナナ",
8        orange:"オレンジ"
9  };
10 listup(arr);
11 listup(obj);
12 function listup(target){
13       print("listup()¥nlength:" + target.length + '¥n');
14       for(var i = 0; i <target.length; i ++){
15             print(i + ':' + target[i]+ '¥n');
16       }
17 }
18 objListup(arr);
19 objListup(obj);
20 function objListup(target){
21       print("objListup()¥n");
22       for(var i in target){
23             print(i + ':' + target[i]+ '¥n');
24       }
25 }
```

察しのいい人ならObjectを変化させたものがArrayであることに気がついただろう
しかしながら、Objectは順序を持たず、Arrayは順序を持つことが大きな違いである

<例4>では10行目と11行目の違いはlengthプロパティの有無である
Objectがlengthプロパティを持たないというよりは、Objectにlengthプロパティと順序をつけたものがArrayである
つまり、Arrayの連想配列の機能はObjectとしての機能によってサポートされていると考えてほしい
そのため、10行目では配列の内容が表示されないのである

なお、Objectのすべての要素をクロールする場合、関数listupのようなfor構文を使う方法ではなく、関数objListupのようなfor in構文を用いる
それは、Arrayもfor inでクロールできるが、Arrayオブジェクトの拡張時などに問題が発生するためであり、Arrayには極力forを使うようにする
以上より、連想配列はArrayを使わず、Objectを使うほうが多いが、Arrayの持つ各種機能（ソートetc.）を使いたい場合はあえて使うこともある

◇for in構文

例5

```
1  var obj ={
2        apple : "りんご",
3        banana : "バナナ",
4        orange : "オレンジ"
5  };
6  for ( var i in obj ){
7        print ( "プロパティ名:" + i + "¥n内容:" + obj [ i ]+ '¥n' );
8  }
```

for in構文はObjectのすべての要素を参照する場合に用いられる 構文は"for(プロパティ名の格納される変数 in 対象){処理}"であり、各プロパティ名が例で言うところの変数iに1ループ毎に次々と代入され、対象の持つプロパティ数だけループする
他の言語で言うところのforeach構文にあたる