JavaScript基礎-第6回資料

◇Arrayの各種メソッド

リスト1

```
1
2   Array . toString ();
3   //配列の文字表現を返す
4   Array . concat ( A [, B [, C ,[,...]]]);
5   //1つ以上の配列を配列に連結する
6   Array . join ( separator );
7   //配列の要素をセパレータでつないだ文字列にする
8   Array . pop ();
9   //配列の最後の要素を無くし、最後の要素を返す
10  Array . push ([ A [, B [,...]]]);
11  //配列の最後尾に1つ以上の要素を追加し、lengthを返す
12  Array . reverse ();
13  //配列を逆順にする
14  Array . shift ();
15  //配列の最初の要素を無くし、最初の要素を返す
16  Array . slice ( start , end );
17  /*
18  start番目からend番目の要素を持つ配列を返す
19  end省略でstart番目以降の配列
20  start（例：-3）が負数でendがないときは
21      最後尾から-start番目（例：後ろから3個目）以降を返す
22  また、startが正数でendが負数のときは
23      start番目から最後尾から-end番目までを返す
24   （例：[0,1,2,3,4].slice(2,-1)==[2,3]）
25  */
26  Array . sort ( func );
27  /*
28  比較関数func(A,B)の戻り値を元に配列をソートする
29   （要素の添え字が0未満でBA、0ならそのまま、
30  0より大きい場合ABの順、funcが無ければ辞書順になる）
31  */
32  Array . splice （ start , delete [, A [, B [,...]]]);
33  /*
34  start番目からdelete個の要素を消去し、
35  後に続く要素（A,B,C,...）があれば追加する
36  */
37  Array . unshift ([ A [, B [,...]]]);
38  //配列の先頭に1つ以上の要素を追加し、lengthを返す
```

<リスト1>はArrayオブジェクトのメソッドである
リストのArray部分に配列を指定し、それぞれにあった引数を与えることで各種機能を使うことができる
（注：あまり使わない一部メソッドは省略した）

◆Objectのクラス的使用法

例1

```
1  var vCollege = function (){}; //v大学クラス（空のコンストラクタ）
2  vCollege . prototype . getSchoolName = function (){ //学校名を返すメソッド
3      return "V大学" ;
4  };
5
6  var F1 = new vCollege (); //V大学クラスのインスタンス（≒コピー）生成
7  F1 . getClassName = function (){ //講座名を返すメソッド
8      return "F1における(ry" ;
9  };
10 print ( F1 . getSchoolName ()+ '¥n' ); //V大学
11 print ( F1 . getClassName ()+ '¥n' ); //F1における(ry
12
13 var jsBasic = function (){}; //JavaScript基礎クラス（空のコンストラクタ）
14 jsBasic . prototype = new vCollege (); //V大学クラスを継承
15 jsBasic . prototype . getClassName = function (){ //講座名を返すメソッド
16     return "JavaScript基礎" ;
17 };
18
19 var newClass = new jsBasic ();
20 print ( newClass . getSchoolName ()+ '¥n' ); //V大学
21 print ( newClass . getClassName ()+ '¥n' ); //JavaScript基礎
```

Objectは多言語のクラスの役割を持つ
実際は関数（Function）がObjectのサブクラスであるため、Functionをコンストラクタとし、Objectの機能を使ってクラス的な動作を実現する
prototypeとはそのObjectに要求されているプロパティが無い場合にプロパティが検索される特殊なプロパティであり、コンストラクタと違ってnewされるたびにプロパティを設定しないため処理を軽くできる
また、<例1>の11行目のように自Objectのprototypeに他Objectのインスタンスを代入すると、 要求されているプロパティが自Objectに存在しない場合、自Objectのprototypeを探し、次に他Objectから探し、次に他Objectのprototypeから探す
これはprototypeをどんどんとたどるprototypeチェーンと呼ばれる仕組みであり、これを利用して継承を実現している
例ではprototype以下のメソッドしか参照していないが、11行目の方法であれば継承したクラスのプロパティも参照可能である