ORCA (Online Research Control system Architecture)

ORCA Script リファレンス

ORCA Script Ver 1.1

もくじ

ORCA Script リファレンス

表記の規則

このマニュアルで使用されている書式の表記について説明します。

要素	例
定数、変数、設問情報、コメント、演算子、制御構文、組み込み関数名のフォントには、Arial, 9 ポイントを太字で使用する。	return Q1000.value
引数は斜体で表記する。	*Value*
中カッコは､その中のオプションを選択しても省略してもよいことを意味する。記述には中カッコは入力しない。	{}
省略記号(...)は、直前の要素を必要な回数だけ繰り返し指定できることを意味する。	...
文章内の true, false(太字)は、真偽値(0 or 1)と同義で使用します。	**true** の場合は‥‥‥
回答情報関数に記述されている *QuestionNumber* は設問を表す「Q+設問番号」を意味する。	*QuestionNumber*

定数

数値

記述例

■文字列の表記方法

```
1
10
+10
-10
```

■16 進数

```
0x1234
0Xabcd
0xABcD
```

■小数

```
1.2
1.5e2
1.5e-2
-1.2
-1.2e2
```

文字列

' ' , " "

書式	'*String*'
	"*String*"

説明

シングルクォーテーション（'）、またはダブルクォーテーション（""）で囲まれたものは文字列として処理します。

記述例

■文字列の表記方法

```
'ABC'
```

```
"abc"
```

■シングルクォーテーション、またはダブルクォーテーションを文字列に使用する

```
var $str1 = "'ABC'";
var $str2 = '"abc"';
―――――――――――――――――――――――――――――――
[結果] $str1 は 「'ABC'」 です。
       $str2 は 「"abc"」 です。
```

■バックスラッシュ(¥)を使用する

```
var $str1 = '¥'ABC¥'';
var $str2 = "¥"abc¥"";
―――――――――――――――――――――――――――――――
[結果] $str1 は 「'ABC'」 です。
       $str2 は 「"abc"」 です。
```

配列

[ ]

書式	[*Array1*, *Array2*, *Array3* ...]

記述例

■数値の配列

```
var $array = [ 1, 2, 3, 4, 5 ];
―――――――――――――――――――――――――――――
[結果]$array は 「 1, 2, 3, 4, 5 」 です。
```

■数値の配列（略記）

```
var $array = [ 1~5 ];
―――――――――――――――――――――――――――――
[結果]$array は 「 1, 2, 3, 4, 5 」 です。
```

■文字列の配列

```
$array = [ "AAA", "BBB", "CCC" ];
―――――――――――――――――――――――――――――
[結果]$array は 「 AAA, BBB, CCC 」 です。
```

■簡易配列定数

```
$array = 1~5;
―――――――――――――――――――――――――――――
[結果]$array は 「 1, 2, 3, 4, 5 」 です。
```

特殊表記

表記	説明
¥n	改行
¥r	改行
¥t	タブ
¥"	ダブルクォーテーション
¥'	シングルクォーテーション
¥¥	円マーク
¥uFFFF	16 進表記 ¥u0123、¥uabcd、¥uABCD 等の 16 進数 4 桁で表記
¥000	8 進表記 ¥012、¥47 等 8 進数で表記 ¥0～¥377(0～255)まで
skip	常に-2 を返す
error	常に-1 を返す
true	常に 1 を返す
ok	常に 1 を返す
yes	常に 1 を返す
on	常に 1 を返す
false	常に 0 を返す
ng	常に 0 を返す
no	常に 0 を返す
off	常に 0 を返す

変数

変数の宣言

var

書式	**var $変数名**

説明

変数を使用するには必ず宣言を行わなければなりません。

変数の宣言は、var $argument のように記述します。

記述例

```
var $argument;      // 宣言
var $i = 0;         // 宣言して 0 で初期化
var $val = Q1200.V // 宣言して Q1200 の入力内容で初期化
```

変数の表記

$

書式	**$変数名**

説明

変数は、接頭に「$」をともなって表記されます。

使用できる文字は、1 文字目が a～ z 、_(アンダーバー)で、2 文字目以降が a～z、A～Z、_(アンダーバー)です。

変数は使用する前に必ず宣言して使用します。宣言のない変数は書式チェックでエラーとなります。

設定した変数が、配列かスカラー(単一の値)かは、代入される値で変化します。変数の値が配列かスカラーかを確認するには isarray などを用いて確認します。

記述例

```
$value
$array_1
$_default
```

関数

関数の定義

function

書式	**function 関数名(** 引数 **)**
	function 関数名(引数 1, 引数 2 ... **)**

説明

関数を定義して作成します。

作成した関数は、呼び出されるより前に記述されていなければなりません。

また、引数に変数の宣言(var)は必要ありません。変数のスコープは関数内だけとなります。

記述例

```
function f1 ( $x )
{
        var $y;
        $y = $x * $x;
        return $y;
}
```

return

書式	**return**
	retunr 式

説明

式を返り値として関数を終了します。

記述例

```
return;
```

```
return Q1000.N + 100;
```

設問情報

設問

Q

書式　　**Q 設問番号**

説明

接頭の Q に続いて、設問番号を表記することで、表記された設問番号の設問を表現します。

ORCA Script で設問番号は、必ず回答情報関数とセットで表記します。

記述例

■設問番号 1000 の設問

```
Q1000
```

選択肢

Q 設問番号.回答情報関数[ ]

書式　　**Q 設問番号.回答情報関数[***Numeric***]**

説明

回答情報関数の中でも選択肢を取り扱う関数に使用し、選択肢を直接指定します。

たとえば Q1000 の選択肢 1 番の、選択の有無を確認するには「Q1000.C[1]」というように記述します。

記述例

■Q1000 の選択肢 1 の選択肢（choice 関数使用時）

```
Q1000.C[1]
```

■Q1000 の選択肢 1 ～3 の選択肢（choice 関数使用時）

```
Q1000.C[1~3]
```

■Q1000 の選択肢 1 ～3 以外の選択肢（choice 関数使用時）

```
Q1000.C[^1~3]
```

■Q1000 の選択肢 1 と 3 以外の選択肢（choice 関数使用時）

```
Q1000.C[^1,3]
```

■Q1000 の選択肢 1 ,3,5 の選択肢（choice 関数使用時）

```
Q1000.C[1,3,5]
```

P

書式	**P[選択肢番号].設問番号**

説明

従属設問を親の選択肢から指定するための修飾子です。

記述例

■ステップ設問の選択肢１～３に該当する Q1200 の値(配列)

```
var $var = P[1~3].Q1200.N;
```

■マトリクス設問の選択肢２に該当する Q1200 の選択内容を Q1300 に代入

```
Q1300.S = P[2].Q1200.S;
```

ブロック

ブロック

{ }

書式
```
{
    処理 1;
    処理 2;
}
```

説明

指定されたブロックに挟まれた複数の行の処理を一まとまりの処理として実行します。

記述例

Q1000 が 18 より小さければブロック内の処理を行う。

```
if( Q1000.N < 18 )
{
        Q1200.N = Q1000.N;
        Q1300.C[1] = 1;
}
```

コメント

ブロックコメント

/*　*/

書式　　**/***
　　コメント
　　/

説明

指定されたブロックコメントに挟まれた複数の行をコメントとして処理し、スクリプト上で無視します。

記述例

```
/*
　　コメント
 */
```

行コメント

//,

書式　　**//** *コメント*
　　# *コメント*

説明

指定されたコメント行をコメントとして処理し、スクリプト上で無視します。

記述例

```
# コメント
```

```
// コメント
```

演算子

算術演算子

演算子	説明
＋	加算
-	減算
*	乗算
/	除算
%	剰余算

数値列比較演算子

演算子	説明
==	数値一致
!=	数値不一致
<>	数値不一致
<	数値小なり
<=	数値小なり、または一致
>	数値大なり
>=	数値大なり、または一致

文字列比較演算子

演算子	説明
eq	文字列一致
ne	文字列不一致
lt	文字列小なり
le	文字列小なり、または一致
gt	文字列大なり
ge	文字列大なり、または一致

論理演算子

演算子	説明
|	論理和
||	論理和
or	論理和
&	論理積
&&	論理積
and	論理積
!	論理否定

代入演算子

演算子	説明
=	代入
+=	加算代入
-=	減算代入
*=	乗算代入
/=	除算代入
%=	剰余算代入
<<=	文字結合代入

制御構文

条件文

if, if-else, if-elsif

書式	**if (条件式) { 処理 }**
	if (条件式) { 処理 } else { 処理 }
	if (条件式) { 処理 } elsif (条件式) { 処理 } else { 処理 }

説明

条件式を評価し、条件式が **true** の場合、処理を実行します。

記述例

■if

・条件式が真の場合処理を実行する

```
if ( 条件式 ) { 処理 }
```

■if-else

・条件式が真の場合は処理 1 を実行し、**false** の場合は処理 2 を実行する

```
if ( 条件式 )
{
        /* 処理 1 */
}
else
{
        /* 処理 2 */
}
```

■if -elsif

・条件式１が真の場合は処理１を実行する

・条件式１が偽で条件式２が真の場合は処理２を実行する

・条件式１、条件式２がともに偽の場合は処理３を実行する

```
if ( 条件式１ )
{
        /* 処理１ */
}
elsif( 条件式２ )
{
        /* 処理２ */
}
else
{
        /* 処理３ */
}
```

for

書式	**for (初期化式; 条件式; 更新式) { 処理 }**
	初期化式 : 初期化する式
	条件式 : ループの条件となる式
	更新式 : ループのたびに更新される式

説明

for 文は条件式が **true** を返すまで処理を繰り返します。

for 文の動作は以下のようになります。

・初期化式は、ループの最初に一回だけ実行する

・処理の最後に更新式を実行し条件式を評価する

・条件式が真の場合、再度処理を実行する

・条件式が偽の場合、ループを終了する

記述例

```
for ( $i = 0; $i < 10; $i += 1 )
{
        /* 処理 */
}
```

while

書式　　**while (条件式) 処理**

説明

while 文は条件式が **true** を返すまで処理を繰り返します。

ループの最初に処理を実行し、処理終了後に条件式を評価し、**true** の場合は処理を実行し、**false** の場合は処理を実行せずにループを終了します。**true** の場合は処理終了後、再度条件式を評価し、**true** の場合は処理を実行し、**false** の場合はループを終了します。

記述例

```
while ( $i < 10 )
{
      /* 処理 */
}
```

do-while

書式　　**do 処理 while (条件式)**

説明

処理を実行後、条件式を評価します。条件式が **true** の場合は、再度処理を実行し、**false** の場合はループを終了します。

処理が必ず一度は実行されます。

記述例

```
do
{
      /* 処理 */
}
while ( $i < 10 )
```

break

書式	**break**

説明

ループ内で使用し、ループを終了させます。

記述例

■例) break の if 文の条件式が**真**の場合、処理 2 を実行せず for 文が終了する。

```
for ( $i =0; $i < 10; $i += 1 )
{
        /* 処理 1 */
        if ( 条件式 ) { break; }
        /* 処理 2 */
}
```

continue

書式	**continue**

説明

ループ内で使用し、この文以降の処理を行わず、次のループに

移行し評価を行います。

for の場合、continue 文の実行後、更新式を実行し、評価式を評価します。

記述例

■例) continue の if 文の条件式が**真**の場合、処理 2 を実行せずに次のループに移行します。

```
for ( $i =0; $i < 10; $i += 1 )
{
        /* 処理 1 */
        if ( 条件式 ) { continue; }
        /* 処理 2 */
}
```

組み込み関数

特殊関数

return

書式	**return(** *Value* **)** **return** *Value* **return**
引数	*Value*：値

説明

Value を戻り値として返し、処理を終了します。*Value* がない場合は、最後に評価した値を返します。

記述例

■true(真)を返す

```
return true;
```

```
return 1;
```

■false(偽)を返す

```
return false;
```

```
return 0;
```

■文字列を返す

```
return 'ABC';
```

■数字を返す

```
return 100;
```

■Q1000 の入力内容を返す

```
retrn (Q1000.V);
```

result

書式	**result()**
引数	なし

説明

最後に評価した値を格納します。

記述例

■result()の格納値

```
var $value = 2;
result();
----------------------------------------
[結果] result()は 「2」 です。
```

回答情報関数

回答情報関数は、回答情報の取得と設定を行える関数群で、必ず設問番号をともなって使用します。
スペック内でのみ有効な関数で、集計では使用できません。

time, TM

書式	*QuestionNumber*.**time** *QuestionNumber*.**TM** (省略書式)
引数	なし

説明

回答者の回答時刻を秒数で返します。

返される回答時刻は回答開始時刻からの経過秒数です。

記述例

```
var $time = Q1000.time;
```

```
var $time = Q1000.TM;
```

pass, PS

書式	*QuestionNumber*.**pass** *QuestionNumber*.**PS**(省略書式)
引数	なし

説明

回答者が *QuestionNumber* を通過していれば **true** を、通過していなければ **false** を返します。

記述例

■回答者の設問通過情報を取得する

```
var $pass = Q1000.pass;
```

```
var $pass = Q1000.PS;
```

value, V

書式	*QuestionNumber*.**value** *QuestionNumber*.**V**(省略書式)
引数	なし
設定	可能

説明

QuestionNumber の入力欄の値を返します。

また左辺代入することで、設問の入力欄に数値を設定します。

記述例

■Q1400 の入力内容を取得する

```
var $val = Q1000.value;
```

```
var $val = Q1000.V;
```

■Q1000 の入力内容に「ABC」を設定する

```
Q1000.value = 'ABC';
```

```
Q1000.V = 'ABC';
```

number, N

書式	*QuestionNumber*.**number** *QuestionNumber*.**N**(省略書式)
引数	なし
設定	可能

説明

QuestionNumber の入力欄の数値を返します。

また左辺代入することで、設問の入力欄に数値を設定します。

記述例

■Q1000(数値回答)の入力欄に入力された数値を取得する

```
$num = Q1000.number;
```

```
$num = Q1000.N;
```

■Q1000 の入力内容に数値 100 を設定する

```
Q1000.number = 100;
```

```
Q1000.N = 100;
```

min, MI

書式	*QuestionNumber*.**min** *QuestionNumber*.**MI**(省略書式)
引数	なし

説明

QuestionNumber の設問が数値回答設問の場合に設定された入力可能な最小値を返します。

QuestionNumber が数値回答設問の場合のみ有効です。

文字回答設問の場合は常に０を返します。

記述例

■Q1000(数値回答)の入力可能な最小値を取得する

```
var $min = Q1000.min;
```

```
var $min = Q1000.MI;
```

max, MA

書式	*QuestionNumber*.**max** *QuestionNumber*.**MA**(省略書式)
引数	なし

説明

QuestionNumber の設問が数値回答設問の場合に設定された入力可能な最大値を返します。

QuestionNumber が数値回答設問の場合のみ有効です。

文字回答設問の場合は常に０を返します。

記述例

■Q1000(数値回答)の入力可能な最大値を取得する

```
var $max = Q1000.max;
```

```
var $max = Q1000.MX;
```

multiple, M

書式	*QuestionNumber*.**multiple** *QuestionNumber*.**M** (省略書式)
引数	なし

説明

対象とする設問が、単一回答設問か複数回答設問かを **true** または **false** で返します。

複数回答設問の場合は **true**、単一回答設問の場合は **false** を返します。

記述例

■Q1000 が単一回答か複数回答かを確認する

```
var $type = Q1000.multiple;
```

```
var $type = Q1000.M;
```

limitupper, LU

書式	*QuestionNumber*.**limitupper** *QuestionNumber*.**LU**(省略書式)
引数	なし

説明

選択できる選択肢の個数(ORCA Editor で「指定個数」を選択した場合の「選択肢数」)を返します。

対象とする設問が単一回答の場合は、常に 1 を返します。

複数回答設問の場合は、1 以上の設定されている選択肢個数までの値を返します。この値に設定されている値以上の選択個数は原則エラーであることを示します。

記述例

■Q1000 の選択肢の最大数を取得する

```
var $val = Q1000.limitupper;
```

```
var $val = Q1000.LU;
```

limitlower, LL

書式	*QuestionNumber*.**limitlower** *QuestionNumber*.**LL**(省略書式)
引数	なし

説明

選択できる選択肢の数(ORCA Editor で「指定個数以下のみ」を選択した場合の「選択肢数」)を返します。

対象とする設問が単一回答設問の場合は、常に 0 又は 1 を返します。

複数回答設問の場合は、0 から設定されている選択肢個数か、Limitupper の値以下の値を返します。この値が Limitupper の値と同値の場合は、選択できる個数が Limitlower の値であることを意味し、それ以外の場合は原則エラーであることを示します。

記述例

■Q1000 の選択肢の最小数を取得する

```
var $val = Q1000.limitlower;
```

```
var $val = Q1000.LL;
```

choice, C

書式	*QuestionNumber*.**choice** *QuestionNumber*.**choice[** *ChoiceNum*, *ChoiceNum* **]** *QuestionNumber*.**choice[^***ChoiceNum*, *ChoiceNum* **]** *QuestionNumber*.**C**(省略書式) *QuestionNumber*.**C[** *ChoiceNum*, *ChoiceNum* **]** (省略書式) *QuestionNumber*.**C[^***ChoiceNum*, *ChoiceNum* **]** (省略書式)
引数	なし
設定	可能

説明

指定された選択肢が選択されている場合は **true** を、選択されていない場合は **false** を返します。
また、左辺代入することで、選択状態に設定することが出来ます。
リストを指定していない場合は、全ての選択肢を対象として、一つでも選択されていれば **true** を返します。
リストを指定されている場合は、その選択肢を対象として、一つでも選択されていれば **true** を返します。

記述例

■Q1000(単一回答)が選択されている場合、Q1200 の選択肢 1 をチェックする

```
if( Q1000.C )
{
        Q1200.C[1] = on;
}
```

■Q1100(複数回答)の選択肢 1 と選択肢 3 を選択状態にする

```
Q1100.C[1,3] = on;
```

choiceall, CA

書式	*QuestionNumber*.**choiceall** *QuestionNumber*.**CA**(省略書式)
引数	なし

説明

選択肢が選択されているかの判定結果を、**true** または **false** で返します。

リストを指定していないときは、全ての選択肢を対象として、すべての選択肢が選択されていれば **true** を返します。

リストを指定しているときは､その選択肢を対象として､すべての選択肢が選択されていれば **true** を返します。

記述例

■Q1200 の選択肢 1～3 が選択されているかを確認する

```
var $flg = Q1200.CA[1~3];
```

count, CT

書式	*QuestionNumber*.**count** *QuestionNumber*.**count[** *ChoiceNum*, *ChoiceNum* **]** *QuestionNumber*.**count[^***ChoiceNum*, *ChoiceNum* **]** *QuestionNumber*.**CT**(省略書式) *QuestionNumber*.**CT[** *ChoiceNum*, *ChoiceNum* **]** (省略書式) *QuestionNumber*.**CT[^***ChoiceNum*, *ChoiceNum* **]** (省略書式)
引数	なし

説明

選択肢が選択されている個数を数値で返します。

選択肢リストを指定している場合は、指定された選択肢にのみ着目し、選択肢個数を返します。

記述例

■Q1100(複数回答)の選択を選択した数を取得する

```
var $count = Q1100.count;
```

```
var $count = Q1100.CT;
```

■Q1100(複数回答)で選択肢を 2 つ以上選択した場合は **true** を返す

```
if( Q1100.CT >= 2 )
{
        return true;
}
```

choicelist, CL

書式	*QuestionNumber*.**choicelist** *QuestionNumber*.**choicelist[** *ChoiceNum*, *ChoiceNum* **]** *QuestionNumber*.**choice;ost[^***ChoiceNum*, *ChoiceNum* **]** *QuestionNumber*.**CL**(省略書式) *QuestionNumber*.**CL[** *ChoiceNum*, *ChoiceNum* **]** (省略書式) *QuestionNumber*.**CL[^***ChoiceNum*, *ChoiceNum* **]** (省略書式)
引数	なし

説明

QuestionNumber に設定されている選択肢の番号リストを返します。

返されるリストは常にソートされています。

選択肢リストを指定している場合は、指定された選択肢にのみ着目し、選択肢の番号リストを返します。

記述例

■Q1000 の選択肢の番号リストを取得する

```
Q1000.choicelist;
```

```
Q1000.CT;
```

■Q1000(単一回答)の選択肢 1～3 の選択肢の番号リストを取得する

```
var $list = Q1000.CT[1~3];
```

selection, S

書式	*QuestionNumber*.**selection** *QuestionNumber*.**selection[** *ChoiceNum*, *ChoiceNum* **]** *QuestionNumber*.**selection[^***ChoiceNum*, *ChoiceNum* **]** *QuestionNumber*.**S**(省略書式) *QuestionNumber*.**S[** *ChoiceNum*, *ChoiceNum* **]** (省略書式) *QuestionNumber*.**S[^***ChoiceNum*, *ChoiceNum* **]** (省略書式)
引数	なし
設定	可能

説明

対象設問にて選択された選択肢の番号リストを返します。返される値は常にソートされています。選択肢リストをしている場合は、指定された選択肢にのみ着目し、選択肢の番号のリストを返します。

また、対象設問の選択を設定することができます。

■スカラー値を使用した場合

選択肢が選択されている場合は指定された選択肢のみに着目し、スカラー値が **true** の場合は選択状態とし、**false** の場合は非選択状態とします。

■配列を使用した場合

配列に格納されている番号の選択肢のみ選択状態とし、配列にない場合は非選択状態とします。部分指定リストが指定されている場合は、その選択肢のみに着目し、選択状態を設定します。

設定する配列はソートされていなくても構いません。

■単一回答に複数の選択肢を選択した場合のルール

選択状態	ルール
Q1.S = on	選択肢番号が最小のものを選択状態とする。
Q1.S = 配列の場合	配列に指定されている番号が最小のものを選択状態とする。
Q1.S[…] = on	部分指定リストの番号が最小のものを選択状態とする。
Q1.S[^…] = on	部分指定リスト以外の選択肢番号が最小のものを選択状態とする。

また、上記ルールにより、以前に選択されていた選択肢と選択されようとする選択肢が異なる場合は、以前に選択されていた選択肢は非選択状態となります。
単一回答では常に選択されている選択肢は一つになります。
設問を表示する場合は、選択された選択肢だけが表示されます。

記述例

■Q1200 で選択された選択肢を取得する

```
var $sel = Q1200.selection;
```

```
var $sel = Q1200.S;
```

■Q1300 に、Q1200 で選択された選択肢を設定する

```
Q1300.S = Q1200.S;
```

unselection, US

書式	*QuestionNumber*.**unselection** *QuestionNumber*.**unselection[** *ChoiceNum*, *ChoiceNum* **]** *QuestionNumber*.**unselection[^***ChoiceNum*, *ChoiceNum* **]** *QuestionNumber*.**US**(省略書式) *QuestionNumber*.**US[** *ChoiceNum*, *ChoiceNum* **]** (省略書式) *QuestionNumber*.**US[^***ChoiceNum*, *ChoiceNum* **]** (省略書式)
引数	なし
設定	可能

説明

対象設問にて選択されていない選択肢の番号リストを返します。返される値は常にソートされています。選択肢リストをしている場合は、指定された選択肢にのみ着目し、選択肢の番号のリストを返します。

また、対象設問の非選択を設定することができます。
ただし単一回答に複数の選択状態を設定することは出来ません。設定した場合は、全ての選択肢が未選択状態となります。

■スカラー値を使用した場合

選択肢が選択されている場合は指定された選択肢のみに着目し、スカラー値が true の場合は非選択状態とし、false の場合は選択状態とします。

■配列を使用した場合

配列に格納されている番号の選択肢のみ非選択状態とし、配列にない場合は選択状態とします。部分指定リストが指定されている場合は、その選択肢のみに着目し、非選択状態を設定します。

設定する配列はソートされていなくても構いません。

記述例

■Q1200 で選択していない選択肢を取得する

```
var $sel = Q1200.unselection;
```

```
var $sel = Q1200.US;
```

■Q1300 に、Q1200 で選択していない選択肢を設定する

```
Q1300.S = Q1200.US;
```

display, D

書式	*QuestionNumber*.**display** *QuestionNumber*.**display[** *ChoiceNum*, *ChoiceNum* **]** *QuestionNumber*.**display[^***ChoiceNum*, *ChoiceNum* **]** *QuestionNumber*.**D**(省略書式) *QuestionNumber*.**D[** *ChoiceNum*, *ChoiceNum* **]** (省略書式) *QuestionNumber*.**D[^***ChoiceNum*, *ChoiceNum* **]** (省略書式)
引数	なし
設定	可能

説明

設問画面に表示された選択肢の配列を返します。また、選択肢の表示と表示する順番を設定することが出来ます。

記述例

■Q1200(複数回答)で表示した選択肢の一覧を取得する

```
var $d = Q1200.display;
```

```
var $d = Q1200.D;
```

■Q1200 の設問画面に選択肢１、３、４を１、３、４の順番で表示設定する

```
Q1200.D = [ 1, 3, 4 ];
```

■Q1200 の設問画面に選択肢１、３、４を４、３、１の順番で表示設定する

```
Q1200.D = [ 4, 3, 1 ];
```

■Q1200 で選択された選択肢を、Q1400 の設問画面で表示にする

```
Q1400.D = Q1200.CL;
```

undisplay, UD

書式	*QuestionNumber*.**undisplay** *QuestionNumber*.**undisplay[** *ChoiceNum*, *ChoiceNum* **]** *QuestionNumber*.**undisplay[** **^***ChoiceNum*, *ChoiceNum* **]** *QuestionNumber*.**UD**(省略書式) *QuestionNumber*.**UD[** *ChoiceNum*, *ChoiceNum* **]** (省略書式) *QuestionNumber*.**UD[** **^***ChoiceNum*, *ChoiceNum* **]** (省略書式)
引数	なし
設定	可能

説明

設問画面に表示されなかった選択肢の配列を返します。また、選択肢の非表示を設定することが出来ます。

記述例

■Q1200(複数回答)で表示されなかった選択肢の一覧を取得する

```
var $d = Q1200.undisplay;
```

```
var $d = Q1200.UD;
```

■Q1200 で選択された選択肢を、Q1400 の設問画面で非表示にする

```
Q1400.UD = Q1200.CL;
```

excludes, E

書式	*QuestionNumber*.**excludes** *QuestionNumber*.**excludes[** *ChoiceNum*, *ChoiceNum* **]** *QuestionNumber*.**excludes[^***ChoiceNum*, *ChoiceNum* **]** *QuestionNumber*.**E**(省略書式) *QuestionNumber*.**E[** *ChoiceNum*, *ChoiceNum* **]** (省略書式) *QuestionNumber*.**E[^***ChoiceNum*, *ChoiceNum* **]** (省略書式)
引数	なし

説明

QuestionNumber から排他選択肢を抽出し、選択肢の番号リストを返します。

返される番号リストは常にソートされています。

記述例

■Q1200（複数回答）の排他選択肢の番号リストを取得する

```
var $list = Q1200.excludes;
```

```
var $list = Q1200.E;
```

choicetext, TX

書式	*QuestionNumber*.**choicetext** *QuestionNumber*.**choicetext[** *ChoiceNum*, *ChoiceNum* **]** *QuestionNumber*.**choicetext[^***ChoiceNum*, *ChoiceNum* **]** *QuestionNumber*.**TX**(省略書式) *QuestionNumber*.**TX[^***ChoiceNum*, *ChoiceNum* **]** (省略書式) *QuestionNumber*.**TX[^***ChoiceNum*, *ChoiceNum* **]** (省略書式)
引数	なし
設定	可能

説明

選択肢名称の取得・設定を行います。

記述例

■Q1000 の選択肢 1 の選択肢（名称）を取得する

```
$cv = Q1000.choicetext[1];
```

```
$cv = Q1000.TX[1];
```

■Q1000 の入力内容を選択肢 1 の名称として設定する

```
Q1000.TX[1] = Q1000.V;
```

choicevalue, CV

書式	*QuestionNumber*.**choicevalue[***ChoiceNum***][***InputFileIdNum***]** *QuestionNumber*.**CV[***ChoiceNum***][***InputFileIdNum***]** (省略書式)
引数	なし
設定	可能

説明

追加入力欄の値を取得します。

また追加入力欄に値を設定することが出来ます。

記述例

■Q1000 の 4 番目の選択肢に登録されている 1 番目の追加入力欄の入力内容を取得する

```
$val = Q1000.choicetextvalue[4][1];
```

```
$val = Q1000.CV[4][1];
```

■Q1000 の 4 番目の選択肢に登録されている 1 番目の追加入力欄に「ABC」と入力します

```
Q1000.choicetextvalue[4][1] = 'ABC';
```

```
Q1000.CV[4][1] = 'ABC';
```

currentstep, CS

書式	*QuestionNumber*.**currentstep** *QuestionNumber*.**CS**(省略書式)
引数	なし
設定	可能

説明

ステップ設問の現在のループ回数を返します。

記述例

■Q1500(ステップ設問)のループ回数を取得する

```
var $step = Q1500.currentstep;
```

```
var $step = Q1500.CS;
```

valuecount, VC

書式	*QuestionNumber*.**valuecount** *QuestionNumber*.**valuecount[** *ChoiceNum*, *ChoiceNum* **]** *QuestionNumber*.**valuecount[^***ChoiceNum*, *ChoiceNum* **]** *QuestionNumber*.**VC**(省略書式) *QuestionNumber*.**VC[** *ChoiceNum*, *ChoiceNum* **]** (省略書式) *QuestionNumber*.**VC[^***ChoiceNum*, *ChoiceNum* **]** (省略書式)
引数	なし
設定	可能

説明

QuestionNumber に設定された選択肢ごとの追加入力欄の数を配列で返します。

記述例

■Q1000(単一回答)の選択肢ごとの追加入力欄の数を取得する

```
var $count = Q1000.valuecount;
```

```
var $count = Q1000.VC;
```

■Q1000(単一回答)の選択肢 4 の追加入力欄の数を取得する

```
var $count = Q1000.VC[4];
```

配列操作関数

配列操作関数は、１つの配列に対して操作を行う関数群です。

front

書式	**front(** *Array* **)**
引数	*Array* ：配列

説明

Array に格納されている先頭の値を返します。

front(*Array*)は、*Array* [0]と同じです。

Array が配列でない場合は、空値を返します。

記述例

■配列の先頭の値を取得

```
var $array = [0, 1, 2, 3];

var $value = front($array);
----------------------------------------
[結果] $value は 「0」 です。
```

■Q1100(複数回答)で選択された選択肢の先頭を取得する

```
var $array = Q1100.S;
var $value = front($array);
```

back

書式	**back(** *Array* **)**
引数	*Array* ：配列

説明

Array に格納されている末尾の値を返します。

back(*Array*)は、*Array* [size(*Array*) - 1]と同じです。

Array が配列でない場合は、空値を返します。

記述例

■配列の末尾の値を取得する

```
var $array = [0, 1, 2, 3];

var $value = back($array);
----------------------------------------
[結果] $value は 「3」 です。
```

■Q1100(複数回答)で選択された選択肢の末尾を取得する

```
var $array = Q1100.S;
var $value = back($array);
```

push

書式	**push(** *Array*, {*Value1 ...* } **)**
引数	*Array* ：配列 *Value1 ...* ：値

説明

Array の末尾に *Value1* 以降の引数の値を追加し、*Array* を返します。

Value1 以降の引数が配列の場合、その配列はリスト展開されます。

記述例

■$array に要素を追加する

```
var $array = [0, 1, 2, 3];
var $value1 = 5;
var $value2 = 10;

var $p_array = push($array, [$value1, $value2] );
----------------------------------------
[結果] $array は 「0,1,2,3,5,10」 です。
```

■$array に要素(配列)を追加する

```
var $array    = [0, 1, 2, 3];
var $value1   = [0, 1, 2, 5];
var $value2   = [10, 11];

var $p_array = push( $array,[$value1, $value2] );
----------------------------------------
[結果] $array は 「0,1,2,3,0,1,2,5,10,11」 です。
```

pop

書式	**pop(** *Array* **)**
引数	*Array* ：配列

説明

Array の末尾の値を削除し、削除した末尾の値を返します。

Array が空配列または配列でない場合は、何もせず空値を返します。

記述例

■配列の末尾の値を削除し、取得する

```
var $array = [0, 1, 2, 3];
var $value = pop($array);
----------------------------------------
[結果] $array は 「0,1,2」 です。
$value は 「3」 です。
```

■Q1100(複数回答)の選択肢の末尾を取得する

```
var $value = pop( Q1100.S );
```

shift

書式	**shift(** *Array* **)**
引数	*Array* ：配列

説明

Array の先頭の値を削除し、削除した先頭の値を返します。

Array が空配列または配列でない場合は、何もせず空値を返します。

記述例

■配列の先頭の値を削除し、取得する

```
var $array = [0, 1, 2, 3];

var $value = shift($array);
----------------------------------------
[結果] $array は 「1,2,3」 です。
$value は 「0」 です。
```

■Q1100(複数回答)の選択肢の先頭を取得する

```
var $value = shift( Q1100.S );
```

unshift

書式	**push(** *Array*, {*Value1 ...* } **)**
引数	*Array* ：配列 *Value1 ...* ：値

説明

Array の先頭に *Value1* 以降の引数の値を追加し、*Array* を返します。

Value1 以降の引数が配列の場合、その配列はリスト展開されます。

Array が配列でない場合は、何もせず空値を返します。

記述例

■$array に要素を追加する

```
var $array = [0, 1, 2];
var $value1 = 5;
var $value2 = 10;

var $p_array = unshift($array, [$value1, $value2] );
----------------------------------------
[結果] $array は 「5,10,0,1,2」 です。
```

■$array に要素(配列)を追加する

```
var $array    = [0, 1, 2];
var $value1   = [0, 1, 5];
var $value2   = [10, 11];

var $p_array = unshift($array,[$value1, $value2] );
----------------------------------------
[結果] $array は 「0,1,5,10,11,0,1,2,」 です。
```

slice

書式	**slice(** *Array*, *Index0*, {*Index1 ...* } **)**
	slice(*Array*, {*Index0~Index* } **)**
引数	*Array* ：配列
	Index0：*Array* の要素数
	Index1 ... ：*Array* の要素数
	Index0~Index1：*Array* の要素範囲

説明

Array より、*Index* で指定された数の要素を抽出し、新たに生成した配列を返します。

生成された配列の要素数０は、*Array*[*Index0*]と等しい

記述例

■配列$array から要素数 1 と 3 を抽出して配列$s_array を生成する

```
var $array = [0, 1, 2, 3];

var $s_array = slice($array, 1, 3);
----------------------------------------
[結果] $s_array は 「1,3」 です。
```

■配列$array から要素数 0～2 を抽出して配列$s_array を生成する。

```
var $array = [0, 1, 2, 3];

var $s_array = slice($array, [0~2]);
----------------------------------------
[結果] $s_array は 「0,1,2」 です。
```

size

書式	**size(** *Array* **)**
引数	*Array* ：配列

説明

Array に格納されている要素の個数を返します。

Array が配列でない場合は、０を返します。

記述例

■配列に登録されている値の個数を取得する

```
var $array = [ 0, 1, 2, 3 ];

var $size = size( $arra y);
----------------------------------------
[結果] $size は 「4」 です。
```

■Q1100(複数回答)で選択された選択肢の数を取得する

```
var $array = Q1100.S;

var $value = size( $array );
```

isarray

書式	**isarray(** *Array* **)**
引数	*Array* ：配列

説明

Array が配列である場合は **true** を、配列でない場合は **false** を返します。

記述例

■$array が配列かどうかを確認する（配列の場合）

```
var $array = [ 0, 1, 2, 3 ];

var $value = isarray( $array );
----------------------------------------
[結果] $value は 「1」 です。
```

■$array が配列かどうかを確認する（配列でない場合）

```
var $array = 1;

var $value = isarray( $array );
----------------------------------------
[結果] $value は 「0」 です。
```

リスト演算関数

リスト演算関数は、リストを演算して結果を返す関数群です。

any

書式	**any(** *Value1* { , *Value2* ... } **)**
引数	*Value1* ：値１ *Value2*～ ：値２～

説明

引数に指定されている値のいずれかが **true** の場合、**true** を返します。また、引数に指定されている値のすべてが **false** である場合、**false** を返します。

第一引数 *Value1* 以降が配列であった場合、その配列はリスト展開されます。

記述例

■$select1～3 のいずれかが **true** であるかを確かめる

```
var $select1  = [ 0, 0, 0, 0 ];
var $select2  = [ 0, 1, 0, 0 ];
var $select3  = [ 0, 0, 0, 0 ];

var $any = any( $select1, $select2, $select3 );
----------------------------------------
[結果] $any は 「 1 」 です。
```

■Q1000、Q1010、Q1020 (すべて単一回答)のいずれかが選択されている場合、100 を返す。

```
if( any( Q1000.C, Q1010.C, Q1020.C ) )
{
      return 100;
}
```

all

書式	**all(** *Value1{* **,***Value2 ...* **}** **)**
引数	*Value1*：値１ *Value2 ...* ：値２～

説明

引数に指定されている値のすべてが **true** である場合、**true** を返します。引数に指定されている値のいずれかが **false** である場合、**false** を返します。

第一引数 *Value1* 以降が配列であった場合、その配列はリスト展開されます。

記述例

■$select1～3 がすべて **false** であるかを確かめる

```
var $select1  = 1;
var $select2  = [ 1, 0, 1, 1 ];
var $select3  = 1;

var $all = all( $select1, $select2, $select3 );
----------------------------------------
[結果] $all は 「0」 です。
```

■Q1000、Q1010、Q1020 (すべて単一回答)に未選択がある場合、-100 を返す。

```
if( ! all( Q1000.C, Q1010.C, Q1020.C ) )
{
      return -100;
}
```

count

書式	**count(** *Value1* { **,***Value2 ...* } **)**
引数	*Value1*：値１ *Value2 ...* ：値２～

説明

引数に指定されている値のうち、**true** であるものの個数を返します。

第一引数 *Value1* 以降が配列であった場合、その配列はリスト展開されます。

記述例

■$select1～3 の **true** の個数を取得する

```
var $select1  = 1;
var $select2  = [1, 0, 0, 0];
var $select3  = 0;

var $count    = count( $select1, $select2, $select3 );
----------------------------------------
[結果] $count は 「2」 です。
```

■Q1100(複数回答)の選択数を取得する

```
var $count = count(Q1100.S);
```

sum

書式	**sum(***Numeric1* { **,** *Numeric2 ...* } **)**
引数	*Numeric1*：数値 1 *Numeric2 ...* ：数値 2 ...

説明

引数に指定されている値をすべて加算した値を返します。

第一引数以降の引数が配列であった場合、その配列はリスト展開されます。

記述例

■$sum1、$sum2 の合計を取得する

```
var $sum1 = [ 0, 1, 2, 3 ];
var $sum2 = 10;

var $total = sum( $sum1, $sum2 );
----------------------------------------
[結果] $total は 「16」 です。
```

■Q1200(数値回答)、Q1210(数値回答)の合計を取得する

```
var $total = sum( Q1200.N, Q1210.N );
```

avg

書式	**avg(** *Numeric1* { **,** *Numeric2* ... } **)**
引数	*Numeric1*：加算する数値 1 *Numeric2* ... ：加算する数値 2 ...

説明

引数に指定されている数値すべてを加算し、その数値を加算した個数で割った数値を返します。

第一引数以降の引数が配列であった場合、その配列はリスト展開されます。

記述例

■$sum1、$sum2 の合計を取得する

```
var $sum1 = [ 0, 1, 2, 3 ];
var $sum2 = 10;

var $avg = avg( $sum1, $sum2 );
----------------------------------------
[結果] $avg は 「3.2」 です。
```

■Q1200(数値回答)、Q1210(数値回答)の平均を取得する

```
var $avg = avg( Q1200.N, Q1210.N );
```

max

書式	**max(***Numeric1* { **,** *Numeric2* ... } **)**
引数	*Numeric1*：数値 1 *Numeric2* ... ：数値 2 ...

説明

引数に指定されている数値のうち、もっとも大きい数値を返します。

第一引数以降の引数が配列であった場合、その配列はリスト展開されます。

記述例

■$num1, $num2 の最大値を取得する

```
var $num1 = 4;
var $num2 = [ 1, 2, 3, 5 ];

var $max= max( $num1, $num2 );
----------------------------------------
[結果] $max は 「5」 です。
```

■Q1200(数値回答)、Q1210(数値回答)の回答から最大の値を一つ取得する

```
var $max = max( Q1200.N, Q1210.N );
```

min

書式	**min(** *Numeric1* { **,** *Numeric2* ... } **)**
引数	*Numeric1*：数値 1 *Numeric2* ... ：数値 2 ...

説明

引数に指定されている数値のうち、もっとも小さい数値を返します。

第一引数以降の引数が配列であった場合、その配列はリスト展開されます。

記述例

■$num1, $num2 の最小値を取得する

```
var $num1 = 4;
var $num2 = [ 1, 2, 3, 5 ];

var $min = min( $num1, $num2 );
----------------------------------------
[結果] $min は 「1」 です。
```

■Q1200(数値回答)、Q1210(数値回答)の最小値を取得する

```
var $min = min( Q1200.N, Q1210.N );
```

sort

書式	**sort(** *Numeric1* { **,** *Numeric2 ...* } **)**
引数	*Numeric1*：数値 1 *Numeric2 ...* ：数値 2 ...

説明

引数に指定されている値を、数値として昇順に並べられた配列を返します。

第一引数以降の引数が配列であった場合、その配列はリスト展開されます。

記述例

■$num1、$num2、$num3 を数値として昇順に並べ替えた配列を取得する

```
var $num1 = 4;
var $num2 = 10;
var $num3 = 2;

var $array = sort( $num1, $num2, $num3 );
----------------------------------------
[結果] $array は 「2,4,10」 です。
```

■$num1、$num2 を数値として昇順に並べ替えた配列を取得する

```
var $num1 = [ 4, 10, 2 ];
var $num2 = [ 100, 2, 1 ];

var $array = sort( $num1, $num2 );
----------------------------------------
[結果] $array は 「1,2,2,4,10,100」 です。
```

unique

書式	**unique(** *Numeric1* { **,** *Numeric2* ... } **)**
引数	*Numeric1*：数値 1 *Numeric2* ... ：数値 2 ...

説明

引数に指定されている値を、数値として重複のない昇順に並べられた配列を返します。

第一引数以降の引数が配列であった場合、その配列はリスト展開されます。

記述例

■$num1、$num2、$num3 を数値として重複のない昇順に並べ替えられた配列を取得する

```
var $num1 = 2;
var $num2 = 10;
var $num3 = 2;

var $array = unique( $num1, $num2, $num3 );
----------------------------------------
[結果] $array は 「2,10」 です。
```

■$num1、$num2 を数値として重複のない昇順に並べ替えられた配列を取得する

```
var $num1 = [ 4, 10, 2 ];
var $num2 = [ 100, 2, 1 ];

var $array = unique( $num1, $num2 );
----------------------------------------
[結果] $array は 「1,2,4,10,100」 です。
```

isunique

書式	**isunique(***Numeric1* { **,** *Numeric2 ...* } **)**
引数	*Numeric1*：数値 1 *Numeric2 ...* ：数値 2 ...

説明

引数に指定されている値すべてが、数値として重複がない場合は **true** を、重複がある場合は **false** を返します。

第一引数以降の引数が配列であった場合、その配列はリスト展開されます。

記述例

■$num1、$num2、$num3 に数値として重複があるかを調べる

```
var $num1 = 2;
var $num2 = 10;
var $num3 = 1;

var $uniq = isunique( $num1, $num2, $num3 );
----------------------------------------
[結果] $uniqは 「1」 です。
```

■$num1、$num2 に数値として重複があるかを調べる

```
var $num1 = [ 4, 10, 2 ];
var $num2 = [ 100, 2, 1 ];

var $uniq = isunique( $num1, $num2 );
----------------------------------------
[結果] $uniqは 「0」 です。
```

max_s

書式	**max_s(** *String1* { **,***String2*... } **)**
引数	*String1*：文字列 1 *String2* ... ：文字列 2 ...

説明

引数に指定されている文字列のうち、文字列としてもっとも大きい値を返します。

第一引数以降の引数が配列であった場合、その配列はリスト展開されます。

記述例

■$char1、$char2、$char3 の文字列としての最大値を取得する

```
var $char1 = 'X';
var $char2 = 'f';
var $char3 = 'a';

var $max_s = max_s( $char1, $char2, $char3 );
----------------------------------------
[結果] $max_s は 「f」 です。
```

■$array の文字列としての最大値を取得する

```
var $array = [ 'a', 'c', 'e' ];

var $max_s = max_s( $array );
----------------------------------------
[結果] $max_s は 「e」 です。
```

min_s

書式	**min_s(** *String1* { **,***String* ... } **)**
引数	*String1*：数値 1 *String2* ... ：数値 2 ...

説明

引数に指定されている文字列のうち、文字列としてもっとも小さい値を返します。

第一引数以降の引数が配列であった場合、その配列はリスト展開されます。

記述例

■$char1、$char2、$char3 の文字列としての最小値を取得する

```
var $char1 = 'X';
var $char2 = 'f';
var $char3 = 'a';

var $min_s = min_s( $char1, $char2, $char3 );
----------------------------------------
[結果] $min_s は 「X」 です。
```

■$array の文字列としての最小値を取得する

```
var $array = [ 'a', 'c', 'e' ];

var $min_s = min( $array );
----------------------------------------
[結果] $min_s は 「a」 です。
```

sort_s

書式	**sort_s(** *String1* { **,** *String2* ... } **)**
引数	*String1*：文字列 1 *String2* ... ：文字列 2 ...

説明

引数に指定されている値を、文字列として昇順に並べられた配列を返します。

第一引数以降の引数が配列であった場合、その配列はリスト展開されます。

記述例

■$char1、$char 2、$char3 を文字列として昇順に並べ替えた配列を取得する

```
var $char1 = 'X';
var $char2 = 'f';
var $char3 = 'a';

var $array = sort_s( $char1, $ char2, $char3 );
----------------------------------------
[結果] $array は 「X,a,f」 です。
```

■$char1、$char2 を文字列として昇順に並べ替えた配列を取得する

```
var $char1 = [ 'X', 'f', 'a' ];
var $char2 = [ '1', 'a', 'Z' ];

var $array = sort_s( $char1, $char2 );
----------------------------------------
[結果] $array は 「1,X,Z,a,a,f」 です。
```

unique_s

書式	**unique_s(** *String1* { **,***String2* ... } **)**
引数	*String1*：文字列 1 *String2* ... ：文字列 2 ...

説明

引数に指定されている値を、文字列として重複のない昇順に並べられた配列を返します。

第一引数以降の引数が配列であった場合、その配列はリスト展開されます。

記述例

■$char1、$char 2、$char3 を文字列として重複のない昇順に並べ替えられた配列を取得する

```
var $char1 = 'a';
var $char2 = 'f';
var $char3 = 'a';

var $array = unique_s( $char1, $char2, $char3 );
----------------------------------------
[結果] $array は 「a,f」 です。
```

■$char1、$char2 を文字列として重複のない昇順に並べ替えられた配列を取得する

```
var $char1 = [ 'X', 'f', 'a' ];
var $char2 = [ '1', 'a', 'Z' ];

var $array = unique_s( $char1, $char2 );
----------------------------------------
[結果] $array は 「1,X,Z,a,f」 です。
```

isunique_s

書式	**isunique_s(** *String1* { **,***String2* ... } **)**
引数	*String1*：文字列 1 *String2* ... ：文字列 2 ...

説明

引数に指定されている値すべてが、文字列として重複がない場合は **true** を、重複がある場合は **false** を返します。

第一引数以降の引数が配列であった場合、その配列はリスト展開されます。

記述例

■$char1、$char2、$ har3 に文字列として重複があるかを調べる

```
var $char1 = 'a';
var $char2 = 'f';
var $char3 = 'b';

var $uniq = isunique_s( $char1, $char2, $char3 );
----------------------------------------
[結果] $uniq は 「1」 です。
```

■$char1、$char2 に文字列として重複があるかを調べる

```
var $char1 = [ 'X', 'f', 'a' ];
var $char2 = [ '1', 'a', 'Z' ];

var $uniq = isunique_s( $char1, $char2 );
----------------------------------------
[結果] $uniq は 「0」 です。
```

reverse

書式	**reverse(** *Value1* { **,***Value2* ... } **)**
引数	*Value1*：数値 1 *Value2* ... ：数値 2 ...

説明

引数に指定されている値の順番に対し、逆順となる配列を生成し、それを返します。

第一引数以降の引数が配列であった場合、その配列はリスト展開されます。

記述例

■$num1、$num2、$num3 の順番を逆に並べ替えた配列を取得する

```
var $num1 = 4;
var $num2 = 10;
var $num3 = 2;

var $array = reverse( $num1, $num2, $num3 );
----------------------------------------
[結果] $array は 「2,10,4」 です。
```

■$num1、$num2 の順番を逆に並べ替えた配列を取得する

```
var $num1 = [ 4, 10, 2 ];
var $num2 = [ 100, 2, 1 ];

var $array = reverse( $num1, $num2 );
----------------------------------------
[結果] $array は 「1,2,100,2,10,4」 です。
```

rotate

書式	**rotate(** *Value1* { **,***Value2 ...* } **,** *Rotation* **)**
引数	*Value1*：数値 1 *Value2 ...* ：数値 2 ... *Rotation*：ローテーション回数

説明

Value1～*Value2 ....* の並び順を持ったリストから、*Rotation* に指定した回数だけ順番をローテーション(回転)した配列を生成し、それを返します。

配列の順番は、以下の例のようにローテーションします。

■ローテーション回数と配列

ローテーション回数	配列
-2	4, 5, 1, 2, 3
-1	5, 1, 2, 3, 4
0	1, 2, 3, 4, 5
1	2, 3, 4, 5, 1
2	3, 4, 5, 1, 2
:	:
:	:

※ローテーション回数=0 が *Value1*～*Value2 ....*の並び順そのままの配列となります

Rotation が配列の場合は、配列の最初の要素(*Array[0]*)を使用します。

記述例

■$array1 の順番を 2 回ローテーションした配列を生成する

```
var $array1 = [1, 2, 3, 4, 5];
var $array 2 = rotate( $array1, 2 );
―――――――――――――――――――――――――――――
［結果］$array は 「3, 4, 5, 1, 2」 です。
```

shuffle

書式	**shuffle(** *Value1* { **,***Value2* ... } **)**
引数	*Value1*：数値 1 *Value2* ... ：数値 2 ...

説明

引数に指定されている値の順番に対し、その順がランダムである配列を生成し、それを返します。

第一引数以降の引数が配列であった場合、その配列はリスト展開されます。

記述例

■$num1、$num2、$num3 の順番をランダムに並べ替えた配列を取得する

```
var $num1 = 4;
var $num2 = 10;
var $num3 = 2;

var $array = shuffle( $num1, $num2, $num3 );
----------------------------------------
[結果例] $array は 「4,2,10」 です。※結果はランダムです
```

■$num1、$num2 の順番をランダムに並べ替えた配列を取得する

```
var $num1      = [4, 10, 2];
var $num2      = [100, 2, 1];

var $array     = shuffle( $num1, $num2 );
----------------------------------------
[結果例] $array は 「2,2,100,1,10,4」 です。※結果はランダムです
```

配列演算関数

配列演算関数は、２つ以上の配列の演算を行いその結果を返す関数群です。

difference

書式	**difference(** *Array1*, *Array2* **)**
引数	*Array1*：配列 1
	Array2：配列 2

説明

Array1 の格納値より、*Array2* に格納されている値を取り除いた値の配列を返します（差集合）。

Array1、*Array2* の格納値は常に数値として評価します。

また、返り値となる配列は常に unique(一意)で昇順の配列です。

Array1、*Array2* のいずれかが配列でない場合は、空値（非配列）を返します。

記述例

```
var $array1 = [ 1, 2, 3, 4 ];
var $array2 = [ 2, 4, 6 ];

var $result = difference( $array1, $array2 );
----------------------------------------
［結果例］$result は 「1,3」 です。
```

■Q1000 の選択から Q1100 の選択を引いた選択肢一覧(差集合)

```
var $sel = difference( Q1000.S, Q1100.S );
```

intersection

書式	**intersection(** *Array1*, *Array2* **)**
引数	*Array1*：配列 1
	Array2：配列 2

説明

Array1 と *Array2* の格納値の両方に含まれる値を格納した配列を返します（交差)。

Array1、*Array2* の格納値は常に数値として評価します。

また、返り値となる配列は常に unique(一意)で昇順の配列です。

Array1、*Array2* のいずれかが配列でない場合は、空値（非配列）を返します。

記述例

```
var $array1 = [ 1, 2, 3, 4 ];
var $array2 = [ 2, 4, 6 ];

var $result = intersection( $array1, $array2 );
----------------------------------------
[結果例] $result は 「2,4」 です。
```

■Q1000 と Q1100 の選択で合致する選択肢一覧(交差)

```
var $sel = difference( Q1000.S, Q1100.S );
```

sym_difference

書式	**sym_difference(** *Array1*, *Array2* **)**
引数	*Array1*：配列 1 *Array2*：配列 2

説明

Array1 と *Array2* の格納値のいずれかにしか含まれていない値を格納した配列を返します（対象差）。

Array1、*Array2* の格納値は常に数値として評価します。

また、返り値となる配列は常に unique(一意)で昇順の配列です。

Array1、*Array2* のいずれかが配列でない場合は、空値（非配列）を返します。

記述例

```
var $array1 = [ 1, 2, 3, 4 ];
var $array2 = [ 2, 4, 6 ];

var $result = sym_difference( $array1, $array2 );
```

［結果例］$result は 「1, 3, 6」 です。

■Q1000 と Q1100 の選択がかぶっていない選択肢一覧(対象差)

```
var $sel = sym_difference( Q1000.S, Q1100.S );
```

union

書式	**union(** *Array1*, *Array2* **)**
引数	*Array1*：配列 1 *Array2*：配列 2

説明

Array1 と *Array2* の格納値のいずれか、または両方に含まれている値を格納した配列を返します（和集合）。

Array1、*Array2* の格納値は常に数値として評価します。

また、返り値となる配列は常に unique(一意)で昇順の配列です。

Array1、*Array2* のいずれかが配列でない場合は、空値（非配列）を返します。

記述例

```
var $array1 = [ 1, 2, 3, 4 ];
var $array2 = [ 2, 4, 6 ];

var $result = union( $array1, $array2 );
```

```
[結果例] $result は 「1, 2, 3, 4, 6」 です。
```

■Q1000 と Q1100 のいずかれで選択されているか、両方で選択されている選択肢一覧(和集合)

```
var $sel = union ( Q1000.S, Q1100.S );
```

equal

書式	**equal(** *Array1*, *Array2* **)**
引数	*Array1*：配列 1 *Array2*：配列 2

説明

Array1 と *Array2* の格納値が等しい場合は **true** を、等しくない場合は **false** を返します。

Array1、*Array2* の格納値は常に数値として評価します。

また、返り値となる配列は常に unique(一意)で昇順の配列です。

Array1、*Array2* のいずれかが配列でない場合は、空値（非配列）を返します。

記述例

```
var $array1 = [ 1, 2, 3, 4 ];
var $array2 = [ 2, 4, 6 ];
var $array3 = [ 4, 2, 6 ];
var $array4 = [ 2, 4, 6 ];

var $result = equal( $array2, $array4 );
```

――――――――――――――――――――――――――――――――――

［結果例］$result は 「1」 です。

※$array2 と$array4 以外の組み合わせはすべて false を返します

■Q1000 と Q1100 の選択が同じ場合

```
equal( Q1000.S, Q1100.S );
```

sort_equal

書式	**sort_equal(** *Array1*, *Array2* **)**
引数	*Array1*：配列 1 *Array2*：配列 2

説明

Array1 と *Array2* の格納値に対し、それぞれを昇順に並べられたとして等しい場合は **true** を返し、等しくない場合は **false** を返します。

Array1、*Array2* の格納値は常に数値として評価します。

Array1、*Array2* のいずれかが配列でない場合は、空値（非配列）を返します。

記述例

```
var $array1 = [ 1, 2, 3, 4 ];
var $array2 = [ 2, 4, 6 ];
var $array3 = [ 4, 2, 6 ];
var $array4 = [ 2, 4, 6 ];

var $result = sort_equal( $array2, $array3 );
----------------------------------------
[結果例] $result は 「1」 です。
※$array1 の組み合わせ以外はすべて true を返します
```

■Q1000 と Q1100 の選択が同じ場合

```
sort_equal( Q1000.S, Q1100.S );
```

disperse

書式	**disperse(** *Array1*, *Array2* **)**
引数	*Array1*：配列 1 *Array2*：配列 2

説明

引数に指定された配列間で、重複した値があった場合は **false** を返し、重複がなかった場合は **true** を返します。ただし 1 つの配列の格納値内にある重複した値は評価対象外となります。

引数の格納値は常に数値として評価します。

引数に指定された値が配列でない場合、その値を唯一の格納値とした配列として評価します。

記述例

```
var $array1 = [ 1, 2, 3, 4 ];
var $array2 = [ 5, 6, 7 ];
var $result = disperse( $array1, $array2 );
----------------------------------------
［結果例］$result は 「1」 です。
```

数学関数

数学関数は、基礎的な数学処理を行うための関数群です。

ceil

書式	**ceil(** *Numeric* **)**
引数	*Numeric*：数字

説明

Numeric が整数の場合はその値を返します。*Numeric* が小数を含む場合は、その値より 0 から遠い方の整数値を返します。

記述例

```
var $num = ceil(12.5);
----------------------------------------
[結果] $num は 「13」 です。
```

```
var $num = ceil(-12.5);
----------------------------------------
[結果] $num は 「-13」 です。
```

floor

書式	**floor(** *Numeric* **)**
引数	*Numeric*：数字

説明

Numeric が整数の場合はその値を返します。*Numeric* が小数を含む場合は、その値より 0 から近い方の整数値を返します。

記述例

```
var $num = floor(12.5);
----------------------------------------
[結果] $num は 「12」 です。
```

```
var $num = floor(-12.5);
----------------------------------------
[結果] $num は 「-12」 です。
```

round

書式	**round(** *Numeric* **)**
引数	*Numeric*：数字

説明

Numeric が整数の場合はその値を返します。*Numeric* が小数を含む場合は、小数部を四捨五入した整数値を返します。

記述例

```
var $num = round(12.5);
----------------------------------------
[結果] $num は 「13」 です。
```

```
var $num = round (-12.5);
----------------------------------------
[結果] $num は 「-13」 です。
```

sign

書式	**sign(** *Numeric* **)**
引数	*Numeric*：数字

説明

Numeric が 0 より大きい場合は 1 を返します。*Numeric* が 0 より小さい場合は-1 を返します。*Numeric* が 0 の場合は 0 を返します。

記述例

■0 より大きい場合

```
var $num = sign(10);
----------------------------------------
[結果] $num は 「1」 です。
```

■0 の場合

```
var $num = sign(0);
----------------------------------------
[結果] $num は 「0」 です。
```

■0 より小さい場合

```
var $num = sign(-12.5);
----------------------------------------
[結果] $num は 「-1」 です。
```

abs

書式	**abs(** *Numeric* **)**
引数	*Numeric*：数字

説明

Numeric の絶対値を返します。

記述例

```
var $num = abs(12);
----------------------------------------
[結果] $num は 「12」 です。
```

```
var $num = abs(-12);
----------------------------------------
[結果] $num は 「12」 です。
```

sqrt

書式	**sqrt(** *Numeric* **)**
引数	*Numeric*：数字

説明

Numeric の平方根を返します。*Numeric* が負数の場合は、0 を返します。

記述例

```
var $num = sqrt(2);
----------------------------------------
[結果] $num は 「1.414213562373095」 です。
```

exp

書式	**exp(** *Numeric* **)**
引数	*Numeric*：数字

説明

Numeric の指数を返します。

記述例

```
var $num = exp(2);
----------------------------------------
［結果］$num は 「7.38905609893065」 です。
```

pow

書式	**exp(** *Numeric1*, *Numeric2* **)**
引数	*Numeric1*：数字１ *Numeric2*：数字２

説明

Numeric 1 の Numeric2 乗の数値を返します。

記述例

```
var $num = pow(2, 4);
----------------------------------------
[結果] $num は 「16」 です。
```

log

書式	**log (** *Numeric* **)**
引数	*Numeric*：数字

説明

Numeric の自然対数値を返します。

Numeric が 0.0 または負数の場合は、空値を返します。

記述例

```
var $num = log(2);
――――――――――――――――――――――――――――――――
［結果］$num は 「0.6931471805599453」 です。
```

log10

書式	**log10 (** *Numeric* **)**
引数	*Numeric*：数字

説明

Numeric の常用対数値を返します。

Numeric が 0.0 または負数の場合は、空値を返します。

記述例

```
var $num = log(2);
----------------------------------------
[結果] $num は 「0.3010299956639812」 です。
```

rand

書式	**rand (** *Numeric* **)**
引数	*Numeric*：数字

説明

0 以上 *Numeric* 未満の一様乱数(整数)を返します。

記述例

■0 以上 10 未満のランダムな数値を取得する

```
var $num = rand(10);
----------------------------------------
[結果] $num には 0～9 の数値がランダムに代入されます。
```

bool

書式	**bool(** *Numeric* **)**
引数	*Numeric*：数字

説明

Numeric が 0 以外の場合は **true** を、0 の場合は **false** を返します。

記述例

```
var $bool = bool(100);
----------------------------------------
[結果] $bool は 「1」 です。
```

■Q1100(複数回答)が一つも選択されなかった場合、数字の 100 を返す。

```
if( bool(Q1100.CT) )
{
      return 100;
}
```

文字列関数

文字列関数は、文字列を操作するための関数群です。

format

書式	**format(** *Format* { , *String1 ...* } **)**
引数	*Format*：書式制御文字 *String1 ...*：文字列

説明

Format の書式制御に従って、第２引数(*String1*)以降の値を展開した文字列を返します。

Format の書式制御文字は、C 言語の printf に準じています。

記述例

■0 詰め 4 桁の 10 進数でフォーマットする

```
var $val = format( "%04d", 2 );
----------------------------------------
[結果] $val は 「0002」 です。
```

length

書式	**length(** *String* **)**
引数	*String*：文字列

説明

String の文字数を返します。

記述例

```
var $len = length( 'cyze' );
----------------------------------------
[結果] $len は 「4」 です。
```

```
var $len = length( 'サイズ' );
----------------------------------------
[結果] $len は 「3」 です。
```

lower

書式	**lower(** *String* **)**
引数	*String*：文字列

説明

String に含まれる文字のうち、大文字を小文字に変換した文字列を返します。

変換対象文字は、u0041～u005a、uff21～uff3a です。

記述例

```
var $str = lower( 'AbcdEfg' );
----------------------------------------
[結果] $str は 「abcdefg」 です。
```

upper

書式	**upper(** *String* **)**
引数	*String*：文字列

説明

String に含まれる文字のうち、小文字を大文字に変換した文字列を返します。

変換対象文字は、u0061～u007a、uff41～uff5a です。

記述例

```
var $str = upper( 'AbcdEfg' );
----------------------------------------
[結果] $str は 「ABCDEFG」 です。
```

rtrim

書式	**rtrim(** *String* **)**
引数	*String*：文字列

説明

String の右側にある連続した空白文字を取り去った文字列を返します。

空白文字は、u0020、u3000 です。

記述例

```
var $str = rtrim( ' abc012 ' );
----------------------------------------
[結果] $str は 「 abc012」 です。
```

ltrim

書式	**ltrim(** *String* **)**
引数	*String*：文字列

説明

String の左側にある連続した空白文字を取り去った文字列を返します。

空白文字は、u0020、u3000 です。

記述例

```
var $str = ltrim( ' abc012 ' );
----------------------------------------
[結果] $str は 「abc012 」 です。
```

trim

書式	**trim(** *String* **)**
引数	*String*：文字列

説明

String の両側にある連続した空白文字を取り去った文字列を返します。

空白文字は、u0020、u3000 です。

記述例

```
var $str = trim( ' abc012 ' );
----------------------------------------
[結果] $str は 「abc012」 です。
```

left

書式	**left(** *String* **,** *Numeric* **)**
引数	*String*：文字列 *Numeric*：数字

説明

String の左側から *Numeric* 文字抽出した文字列を返します。

Numeric が 0 以下、または *Numeric* が *String* の文字数以上の場合、空文字を返します。

記述例

```
var $str = left( 'ABCDE', 3 );
----------------------------------------
[結果] $str は 「ABC」 です。
```

right

書式	**right(** *String* **,** *Numeric* **)**
引数	*String*：文字列 *Numeric*：数字

説明

String の右側から *Numeric* 文字抽出した文字列を返します。

Numeric が 0 以下、または *Numeric* が *String* の文字数以上の場合、空文字を返します。

記述例

```
var $str = right( 'ABCDE', 3 );
----------------------------------------
[結果] $str は 「CDE」 です。
```

mid

書式	**mid(** *String* **,** *Position*, *Numeric* **)**
引数	*String*：文字列 *Position*：数字（開始文字位置） *Numeric*：数字（抽出文字数）

説明

String の左側 *Position* 文字目から *Numeric* 文字抽出した文字列を返します。

Postion は 0 以上で *Numeric* は 1 以上の値を設定します。

Postion が、0 以下または *String* の文字数以上に設定している場合、または *Numeric* が 1 より小さい場合は、空文字列を返します。

また、。

記述例

```
var $str = mid( 'ABCDE', 1, 3 );
----------------------------------------
[結果] $str は 「BCD」 です。
```

narrow

書式	**narrow(** *String* **)**
引数	*String*：文字列

説明

String に含まれる文字のうち、全角文字を半角文字に変換した文字列を返します。

変換対象文字は、u3000、uff01～uff5e です。

記述例

```
var $str = narrow( 'ＡＢＣＤ１２３４56' );
----------------------------------------
[結果] $str は 「ABCD123456」 です。
```

widen

書式	**widen(** *String* **)**
引数	*String*：文字列

説明

String に含まれる文字のうち、半角文字を全角文字に変換した文字列を返します。

変換対象文字は、u0020、u0021～u007e です。

記述例

```
var $str = widen( 'ＡＢＣＤ１２３４５６' );
----------------------------------------
[結果] $str は 「ＡＢＣＤ１２３４５６」 です。
```

html_encode

書式	**html_encode(** *String* **)**
引数	*String*：文字列

説明

String に含まれる以下の文字列を文字実体参照（実体参照）に置換して返します。

■置換対象と置換文字列

置換対象	置換文字列
<	<
>	>
&	&
″	"

記述例

```
var $str = html_encode( '<ABC>' );
----------------------------------------
[結果] $str は 「&lt;ABC&gt;」 です。
```

url_encode

書式	**url_encode(** *String { , Encode }* **)**
引数	*String*：文字列 *Encode*：文字列（エンコード）

説明

String を URL エンコードした文字列を返します。

デフォルトでは、*String* を UTF-8 に変換した文字列に対して URL エンコードを行います。

Encode には、URL エンコードを行う前に変換するエンコードを指定することができます。

■代表的なエンコード

"sjis"　"shift-jis"　"shift_jis"　"cp932"　"euc-jp"　"ujis"　"iso-2022-jp"　"iso2022jp"

記述例

■文字列を Shift-jis にエンコードして URL エンコードを行う

```
url_encode( 'ABC', 'shift-jis' );
```

■文字列を EUC-JP にエンコードして URL エンコードを行う

```
url_encode( 'ABC', 'euc-jp' );
```

■文字列を UTF-8 にエンコードして URL エンコードを行う

```
url_encode( 'ABC' );
```

match

書式	**match(***String*, *Regex*, { *ICase* } **)**
引数	*String*：文字列
	Regex：正規表現
	ICase：大文字小文字の区別

説明

String に対して、*Regex* で指定した正規表現で検索を行い、その結果を返します。

Regex に指定する文字列は、Perl の正規表現に準じます。*ICase* に **true** を指定した場合は、文字の大文字、小文字を区別しない検索を行います。

■match の返り値(配列 $m)

```
var $m = match( $str, $regex {, $icase } );
```

配列	格納値の説明
$m[0]	一致、不一致判定。真の場合、一致。偽の場合、不一致。
$m[1]	正規表現に一致した部分文字列。
$m[2]	正規表現に一致した部分以前の文字列。
$m[3]	正規表現に一致した部分以後の文字列。
$m[4～]	正規表現内の部分検索に一致した文字列。

なお、match()では以下の 2 点の制限があります。

・正規表現¥s に対して、全角スペースは一致しない。

・*ICase* に **true** を指定した場合でも、全角英字に関しては、大文字、小文字を区別しない検索を行うことができない。

記述例

■サンプル

```
var $m;
$m = match( "ABCDCEF", "[CD]+" );


―――――――――――――――――――――――――――――
```

```
[結果]  $m[0] は 「1」 です。
        $m[1] は 「CDC」 です。
        $m[2] は 「AB」 です。
        $m[3] は 「EF」 です。
```

■Q1000 の入力内容が 00～09 だったら、Q1200 番の入力内容に設定する。

```
var $m;

if( $m = match( Q1000.V, "^0[0-9]$" ) )
{
      Q1200.N = Q1000.N;
}
```

acode

書式	**acode(** *String* **)**
引数	*String*：文字列

説明

String の先頭 1 文字が U+0000-007F の場合のみ、その値のコードを数値で返します。

String が U+0000-007F 以外の場合は、-1 を返します。

achar

書式	**achar(** *Numeric* **)**
引数	*Numeric*：数字

説明

Numeric が 0x00 ～ 0x7f の場合のみ、対応する U+0000-007F の 1 文字を格納する文字列を返します。

Numeric が 0x00 ～ 0x7f 以外の場合は、-1 を返します。

to_num

書式	**to_num(** *String { , Radix }* **)**
引数	*String*：文字列(半角英数字) *Radix*：基数

説明

String を数値に変換して返します。

変換仕様は C 言語の strtol に準拠しています。

デフォルト(*Radix* を未指定の場合)は、10 進数として変換します。

記述例

■文字列「ffffff」を 16 進数として、数字に変換する

```
var $num = to_num( "ffffff", 16 );
----------------------------------------
[結果] $num は 「16777215」 です。
```

■文字列「0000011」を 2 進数として、数字に変換する

```
var $num = to_num( "0000011", 2 );
----------------------------------------
[結果] $num は 「3」 です。
```

format_array

書式	**format_array(** *FormatString*, *Array* ... **)**
引数	*FormatString*：文字列を括る文字列 *Array*：配列

説明

FormatString を第二引数の配列１つ１つに適用した文字列が格納された文字列を返します。

記述例

```
var $ret = format_array( "<%s>", [ "AAA", "BBB", "CCC"] );
----------------------------------------
[結果] $ret[0] は 「<AAA>」 です。
       $ret[1] は 「<BBB>」 です。
       $ret[2] は 「<CCC>」 です。
```

join_str

書式	**join_str(** *Separate, Array* **)**
引数	*Separate*：セパレータ *Array*：配列

説明

配列格納値を *Separate* でつなげた文字列を返します。

記述例

```
var $str = join_str( '-', [ 'A', 'B', 'C' ]);
----------------------------------------
[結果] $str は 「A-B-C」 です。
```

```
var $str = join_str(
      "," ,
      format_array( '<%s>', ['A','B','C'] )
 );
----------------------------------------
[結果] $str は 「<A>,<B>,<C>」 です。
```

md5sum

書式	**md5sum(** *String* **)**
引数	*String*：文字列

説明

String から UTF-８エンコードした文字列を作成し、その UTF-８バイナリイメージより、128 ビットの md5 ハッシュ値を算出した後、16 進文字列を生成し、返します。

記述例

```
var $val = md5sum( '20080101T000000+9000' );
----------------------------------------
[結果]$val は「a2a96139a2148af8debf0ab0fe25efc9」 です。
```

sha1sum

書式	**sha1sum(** *String* **)**
引数	*String*：文字列

説明

String から UTF-８エンコードした文字列を作成し、その UTF-８バイナリイメージより、160 ビットの sha1 ハッシュ値を算出した後、16 進文字列を生成し、返します。

記述例

```
var $value = sha1sum( '20080101T000000+9000' );
----------------------------------------
[結果]$val は「1c58ff2969cf1e06dfd57dcd03444d80b7ba5152」 です。
```

時間関数

時間関数は、時間に関する関数群です。

time

書式	**time()**
引数	なし

説明

システムの紀元（1970 年 1 月 1 日 00:00:00 UTC）からの経過時間を秒単位で返します。

記述例

```
var $time = time();
----------------------------------------
［結果例］$time は 「1262271676」 です。
※結果はシステムにアクセスする時刻で変化します
```

timestr

書式	**timestr(** *second* **)**
引数	*second*：秒数

説明

second を YYYYMMDDThhmmss±hhmm の形式にして返します。

フォーマットは ISO8601 の基本形式に準拠しています。

記述例

```
var $time = timestr( time() );
―――――――――――――――――――――――――――――――――
［結果例］$time は 「20080101T000820+0900」 です。
※結果は time()の値で変化します
```

timestr_gm

書式	**timestr_gm(** *Second* **)**
引数	*Second*：秒数

説明

Second を YYYYMMDDThhmmssZ の形式にして返します。

フォーマットは ISO8601 の基本形式に準拠しています。

記述例

```
var $time      = timestr_gm( time() );
----------------------------------------
[結果例] $time は 「20071231T150820Z」 です。
※結果は time()の値で変化します
```

timefromstr

書式	**timefromstr(** *time* **)**
引数	*time*：時刻(YYYYMMDDThhmmss±hhmm)

説明

time の紀元（1970 年 1 月 1 日 00:00:00 UTC）からの経過時間を秒単位で返します。

time が時刻フォーマットとして正しくない、もしくは 1970 年 1 月 1 日以前の場合は、-1 を返します。

記述例

```
var $time = timefromstr( '20080101T000820+0900');
----------------------------------------
[結果例] $time は 「1199113700」 です。
```

回答者情報関数

回答者情報関数は、回答者の環境やアクセスしてきた情報を取得する関数群です。

access_seq

書式	**access_seq()**
引数	なし

説明

回答者の回答ＵＲＬへのアクセス順番を返します。

記述例

```
var $seq = access_seq();
```

last_id

書式	**last_id()**
引数	なし

説明

回答者のアクセスした終了ページの終了番号を返します。

last_id は ORCA Core のみで使用します。スペック上では使用できません。

記述例

■終了番号 50 のページにアクセスした回答者

```
last_id == 50
```

category_id

書式	**category_id()**
引数	なし

説明

個人認証つきの回答ＵＲＬから対象者情報設定の対象テーブルのカテゴリ名を返します。

記述例

```
return category_id();
```

table_id

書式	**table_id()**
引数	なし

説明

個人認証つきの回答ＵＲＬから対象者情報設定の対象テーブル名を返します。

記述例

```
return table_id();
```

answer_id

書式	**answer_id()**
引数	なし

説明

回答者の対象者情報設定で設定された回答者ＩＤを返します。

回答者ＩＤを取得するには、個人認証を含む回答ＵＲＬにアクセスしている必要があります。

記述例

```
return answer_id();
```

mailkey

書式	**mailkey(** *Numeric* **)**
引数	*Numeric*：0～3

説明

回答ＵＲＬのパラメータに個人認証（メールキー）を持つ場合に、指定された *Numeric* のメールキー情報を返します。

回答ＵＲＬのパラメータに個人認証を持たない場合は、空値を返します。

■引数設定値と取得する内容

引数	説明
0	メールキーのバージョン
1	スペックＩＤ
2	回答者の「カテゴリの値」
3	回答者の「テーブル名.登録番号」

記述例

```
var $specid = mailkey(1);
```

start_time

書式	**start_time()**
引数	なし

説明

回答者の回答開始秒数を返します。

記述例

```
var $time = timestr( start_time() );
# $time は ans_info('TIME_START')と同値です;
```

proc_time

書式	**proc_time()**
引数	なし

説明

回答者の回答開始からの経過秒数を返します。

記述例

```
var $time = proc_time();
```

acc_param

書式	**acc_param(** *Parameter* **)**
引数	*Parameter*：アクセスパラメータ

説明

*Prameter*に指定された値をもとにアクセスした回答 URL のパラメータまたは対象者情報設定で定義した「項目ＩＤ」から値を返します。

項目ＩＤの方がＵＲＬパラメータより優先されます。

記述例

■アクセスした回答 URL のパラメータから uid の値を取得する

```
回答 URL http://www.example.com?uid=cyze001&cat=1

var $uid = acc_param('uid');
----------------------------------------
[結果] $uid は 「cyze001」 です。
```

■対象者情報設定の「age」の項目を取得する

```
var $age = acc_param('age');
```

ans_info

書式	**ans_info(** *Key* **)**
引数	*Key*：キー値

説明

回答データ(回答ファイル)の INFO 欄の値と同値を返します。

記述例

```
var $time = ans_info('TIME_START');
```

ans_info_param

書式	**ans_info_param(** *Class*, *ParamName* **)**
引数	*Class*：クラス *ParamNamer*：パラメータ名

説明

回答データ(回答ファイル)の INFO 欄の PARAM の値と同値を返します。

記述例

■ORCA の稼働サーバー名を取得する

```
return ans_info_param( 'ENV', 'SERVER_NAME' );
```

■回答者のアクセスしたブラウザの種類を取得する

```
return ans_info_param( 'ENV', 'HTTP_USER_AGENT' );
```

auth_uid

書式	**auth_uid(** *AnswererID*, *Password* **)**
引数	*AnswererID*：回答者ＩＤ *Password*：パスワード

説明

対象者情報設定で登録された回答者ＩＤとパスワードの組み合わせが一致しているかを **true**、または **false** で返します。

記述例

■Q1000 と Q1100 の入力内容からユーザー認証を行う

```
auth_pwd( Q1000.V, Q1100.V );
```

auth_pwd

書式	**auth_pwd(***Password* **)**
引数	*Password*：パスワード

説明

対象者情報設定で登録された回答者のパスワードが一致しているかを **true**、または **false** で返します。

記述例

```
auth_pwd( Q1000.V );
```

クォータ情報関数

関数名	ページ
quota_all	154
quota_cat	155
quota_que	156
quota_all_num	157
quota_cat_num	158
quota_que_num	159
quota_all_send	160
quota_cat_send	161
quota_all_plan	162
quota_cat_plan	163
quota_que_plan	164
quota_all_lim	165
quota_cat_lim	166
quota_que_lim	**エラー! ブックマークが定義されていません。**

クォータ情報関数は、設定されたクォータの情報を取得する関数群です。

quota_all

書式	**quota_all()**
引数	なし

説明

クォータ全体がクォータに達している場合は **true** を、達していない場合は **false** を返します。

記述例

```
quota_all();
```

quota_cat

書式	**quota_cat(***CategoryID* **)**
引数	*CategoryID*：カテゴリ ID

説明

指定された *CategoryID* を持つカテゴリクォータがクォータ数に達した場合は **true** を、達していない場合は **false** を返します。

記述例

```
quota_cat( 'C0001' );
```

quota_que

書式	**quota_que(** *QuotaID* **)**
引数	*QuestionID*：設問クォータ ID

説明

指定された *QuotaID* を持つ設問クォータがクォータ数に達した場合は **true** を、達していない場合は **false** を返します。

記述例

```
quota_que( 'C0001' );
```

quota_all_num

書式	**quota_all_num()**
引数	なし

説明

クォータ全体の回収数を返します。

記述例

```
var $num = quota_all_num();
```

quota_cat_num

書式	**quota_cat_num(***CategoryID* **)**
引数	*CategoryID*：カテゴリ ID

説明

指定された *CategoryID* を持つカテゴリクォータの回収数を返します。

記述例

```
var $num = quota_cat_num( 'C0001' );
```

quota_que_num

書式	**quota_que_num(** *QuestionID*)
引数	*QuotaID*：設問クォータ ID

説明

指定された *QuotaID* を持つ設問クォータの回収数を返します。

記述例

```
var $num = quota_que_num( 'Q0001' );
```

quota_all_send

書式	**quota_all_send()**
引数	なし

説明

クォータ全体の発送数を返します。

記述例

```
var $num = quota_all_send();
```

quota_cat_send

書式	**quota_cat_send(***CategoryID* **)**
引数	*CategoryID*：カテゴリ ID

説明

指定された *CategoryID* を持つカテゴリクォータの発送数を返します。

記述例

```
var $num = quota_cat_send( 'C0001' );
```

quota_all_plan

書式	**quota_all_plan()**
引数	なし

説明

クォータ全体の回収予定数を返します。

記述例

```
var $num = quota_all_plan();
```

quota_cat_plan

書式	**quota_cat_plan(***CategoryID* **)**
引数	*CategoryID*：カテゴリ ID

説明

指定された *CategoryID* を持つカテゴリクォータの回収予定数を返します。

記述例

```
var $num = quota_cat_plan( 'C0001' );
```

quota_que_plan

書式	**quota_que_plan(** *QuestionID*)
引数	*QuestionID*：設問クォータ ID

説明

指定された *QuestionID* を持つ設問クォータの回収予定数を返します。

記述例

```
var $num = quota_que_plan( 'Q0001' );
```

quota_all_lim

書式	**quota_all_lim()**
引数	なし

説明

クォータ全体のクォータ数を返します。

記述例

```
var $num = quota_all_lim();
```

quota_cat_lim

書式	**quota_cat_lim(***CategoryID* **)**
引数	*CategoryID*：カテゴリ ID

説明

指定された *CategoryID* を持つカテゴリクォータのクォータ数を返します。

記述例

```
var $num = quota_cat_lim( 'C0001' );
```

quota_que_lim

書式	**quota_que_lim(** *QuotaID* **)**
引数	*QuestionID*：設問クォータ ID

説明

指定された *QuotaID* を持つ設問クォータのクォータ数を返します。

記述例

```
var $num = quota_que_lim( 'Q0001' );
```

ファイルアップロード情報関数

ファイルアップロード情報関数は、設定されたクォータの情報を取得する関数群です。

fileexist

書式	**QuestionNumber.fileexist** **QuestionNumber.FE(省略書式)**
引数	なし

説明

回答者が QuestionNumber でファイルをアップロードしていれば true を、していなければ false を返します。

記述例

■回答者のファイルアップロードの有無を取得する（Q1000 がファイルアップロード設問）

```
var $result = Q1000.fileexist;
```

```
var $result = Q1000.FE;
```

filename

書式	**QuestionNumber.filename** **QuestionNumber.FN(省略書式)**
引数	なし

説明

回答者が QuestionNumber で指定したファイルのクライアントファイル名を取得します。(アップロードされたファイルは[ANSID_設問番号.拡張子]の形で保存されますが、.FN で元のファイル名を取得することができます。)

記述例

■回答者のクライアントファイル名を取得する（Q1000 がファイルアップロード設問）

```
var $str = Q1000.filename;
```

```
var $str = Q1000.FN;
```

filesize

書式	**QuestionNumber.filesize** **QuestionNumber.FS(省略書式)**
引数	なし

説明

回答者が QuestionNumber で選択したファイルのファイルサイズを取得します。

記述例

■ファイルサイズを取得する（Q1000 がファイルアップロード設問）

```
var $num = Q1000.filesize;
```

```
var $num = Q1000.FS;
```

filetype

書式	**QuestionNumber.filetype** **QuestionNumber.FT(省略書式)**
引数	なし

説明

回答者が QuestionNumber で指定したファイルのファイル種別(サーバー側での拡張子からの判定値)を取得します。

記述例

■ファイル種別(サーバー側での拡張子からの判定値）を取得する（Q1000 がファイルアップロード設問）

```
var $str = Q1000.filetype;
```

```
var $str = Q1000.FT;
```

filetypecInt

書式	**QuestionNumber.filetypecInt** **QuestionNumber.FTC(省略書式)**
引数	なし

説明

回答者が QuestionNumber で指定したファイルのファイル種別(ブラウザ側申告値) を取得します。

記述例

■ファイル種別(ブラウザ側申告値) を取得する（Q1000 がファイルアップロード設問）

```
var $str = Q1000.filetypecInt;
```

```
var $str = Q1000.FTC;
```

fileext

書式	**QuestionNumber.fileext** **QuestionNumber.FX(省略書式)**
引数	なし

説明

回答者が QuestionNumber で指定したファイルの拡張子を取得します。

記述例

■ファイル拡張子を取得する（Q1000 がファイルアップロード設問）

```
var $str = Q1000.fileext;
```

```
var $str = Q1000.FX;
```

fileproperty

書式	**QuestionNumber.fileproperty["param"]** **QuestionNumber.PR["param"](省略書式)**
引数	["EX:MAK"]：EXIF 情報のメーカー値（例値 ：KDDI-SA ） ["EX:MOD"]：EXIF 情報のモデル値（例値 ：W51SA ） ["EX:DAT"]：EXIF 情報の日付値、記録値（例値 ： 2014:07:04 21:01:52） ※メーカー、モデルごとに、日付フォーマットは異なる可能性あり ["EX:LAT"]：EXIF-GPS 情報の緯度（例値: 35.699777777777776） ※正数の場合は、北緯、負数の場合は、南緯 ["EX:LON"]：EXIF-GPS 情報の経度（例値: 139.77170000000001） ※正数の場合は、東経、負数の場合は、西経

説明

回答者が QuestionNumber で指定したファイルのプロパティを取得します。PR のみで使用することはできません。引数の指定が必須です。引数に指定可能なのは上記の 5 つのみです。（2014 年 12 月現在）

記述例

■ファイルのプロパティ値を取得する（Q1000 がファイルアップロード設問）

```
return Q1000.PR["EX:MAK"];
    →（例）「Apple」を返す
```

```
return Q1000.PR["EX:MOD"];
    →（例）「iPad mini 2」を返す;
```

索引

D

E

F

H

I

J

L

T

U

V

W

Y

か

き

さ

し

す

ORCA Script リファレンス

2008 年 5 月 29 日　初版版発行

2016 年 7 月 11 日　第二版版発行

発行者　株式会社サイズ

発行日　2016 年 7 月 11 日

連絡先　株式会社サイズ

〒150-0043

東京都渋谷区道玄坂 1-18-1 shibuyaINCS7A

電話　03-5459-3817

URL　http://www.cyze.jp/

E-mail　info@cyze.jp