SEDNOVE

Sncode/Extenso

pierre.Laplante@sednove.com

Version 1.4 : 2020-07-03

Course #4

- What we have seen in course #3
  - Boolean
  - String
  - Sring functions
  - String operators

String function : cgidata parse POST/GET and command line arguments

- **cgidata**(parse_get : true|false, parse_post : true|false, conflict : replace|array|keep|join, join : "join string", callback : "…", postmax : integer, disable_upload : true|false, directory : "directory upload", fileconflict : "overwrite | rename | error", maxsize : 456, extention : "jpg,png,gif", ct : "format/gif, image/jpeg", "file parameter" : { directory : "directory upload to overwrite default directory upload", url : "url….", fileconflict : "overwrite|rename|error", maxsize : 456, ct : …, extention : …}, esc_cgidata:bool)

POST VS GET

- POST: data enclosed in the body of the request message
  - Usually use in a form

```
<form method="POST">
     <input name="pierre">
</form>
```

- GET: data is within the query string

https://sncode.sednove.com/index.sn?a=b&c=d

HTTP : methods

- From https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
    - GET
    - HEAD
    - POST
    - PUT
    - DELETE
    - TRACE
    - CONNECT
    - PATCH

Form Example for cgidata

- https://sncode.sednove.com/method.sn
- Source : https://sncode.sednove.com/method_src.html
- Form example:

```
<form method="POST" action="?">
    <input type="text name="email">
    <button type="submit">Submit</button>
</form>
```

Form example for CGIDATA

- Method specify how the data will be transfert
  - POST
  - GET
- Action specify which page will be called ? stand for the current page
- input specify an input of some sort
  - type="text" specify a text input
  - name="email" specify the name of the field
- button of type submit specify that when the user click, the form is submitted to the action using the specified method

Using cgidata in Sncode

- the function cgidata() read the data from the form and put it in a JSON format
- JSON format ? Who is JSON ?

JSON : Javascript Object Notation

- https://www.json.org/json-en.html
- **JSON** (JavaScript Object Notation) is a lightweight data-interchange format
- It is also 2 types in Sncode:
  - array : list of elements
  - associative array, hashor context: list of name elements
  - element can be any types

Sncode array

- An array is initialise with the following code:

```
a = [
   true,                          // A boolean
   "string",                      // A string
   1,                             // An integer
   1.1,                           // A float
   [1,2],                         // An array
   { "x" :1, "y" : [1, 2] }    // An hash array
]; a;
[true,"string",1,1.1,[1,2],{"x":1,"y":[1,2]}]
```

Array

- Another way to initialize an array:

```
a = [];              // empty array
a = array(5);        // array with 5 elements null
a;
[null,null,null,null,null]
```

Array

- Accessing elements of an array

```
 a = [true,"string",1,1,[1,2],{"x":1,"y":[1,2]}];
 a[0];                // return true, the first element
 a[length(a)-1];      // return the last element
a[-2]; " ";           // return [1,2]
a[1:3]; " ";          // return ["string",1]
a[-3:-2]; " ";        // return [1]
```

array update

- Assignation:

```
a = [1,2,3,4,5];
a[2] = 5;
a[-1] = 10;
a;
// return [1,2,5,4,10]
```

- But you cannot use range in assignation
  - a[2:3] = 6; // will generate an error at execution
- How can you do this? There is a function call splice for that

array function : splice

- splice(array, start, length, elements...);
- Example:

```
a = [1,2,3,4,5];
splice(a,1,2,6,8,10);     // return [1,6,8,10,4,5]
splice(a, 2, 2);          // return [1,2,5]
splice(a,-3, 2);          // return [1,2,5
]
```

Array functions : first / top

- Return first/top element of an array

```
a = [1,2,3,4,5];
top(a);                         // return 1
first(a);                       // return 1
```

array functions : last/tail and pop

- Return the last element of an array
- Function last and tail are the same

```
a = [1,2,3,4,5];
a.last();             // return 5
```

- Remove the last element of an array and return it

```
a.pop(); a;           // return 5 [1,2,3,4]
```

array functions : join

- Join elements of an array and return a string
- You have to specify the separator
- Example:

```
a = [1,2,3,4,5];
a.join(","); // return 1,2,3,4,5
```

- split can be use to split a string in a array:

```
split("1,2,3,4,5",","); // return [1,2,3,4,5]
split("pierre",""); // return ["p","i","e","r","r","e"]
```

Array function : reverse

- reverse an array
- Exercice:
  - Using the functions you have learn, write a small function to reverse a string

```
function reverse_str(str)
  new_str = join(reverse(split(str,"")), "");
  return new_str;
endf

reverse_str("pierre");
```

Array functions : push

- Push an element on top of array

```
a = [1,2,3];
push(a,5); a; // return [1,2,3,5];
```

- push will increase the size of the array (double the memory allocated)
- array have 2 sizes:
  - Real size
  - Memory allocated
- Increasing the size of an array imply the copy of the array

Array function : array

- array(10) will create an array with 10 position initialize with null
- [null,null,null,null,null,null,null,null,null,null];
- It's better to create the initial array with the correct size than to increment the array.
- array in sncode are real array
- indexing an array in sncode is really fast
- It's not an hash array which is less performant
- Adding an element to an array is like creating a new array with size+1

Array functions : range

- Generate an integer array with the range provided. Containing arithmetic progressions.
- **range(**stop);
- **range(**start,stop[, step]);
- range(10); **return** [0,1,2,3,4,5,6,7,8,9]
- range(10,20,2); **return** [10,12,14,16,18]

```
for i in range(10,20,2) do
     i; " ";
endfor // return 10 12 14 16 18
```

Array functions : shift

- Remove first element from array and return it.
- **shift**(array)

```
a = [1,2,3,4,5];
shift(a); a; // return 1 [2,3,4,5]
```

Array functions : array_search

- Searches the array for a given value and returns the first corresponding key if successful
- **array_search(**array, needle);
- The comparaison is based on the type of needle

```
a = [1,2,3,4.0,5];
a.array_search(3);       // return 2
a.array_search(33);      // return -1
a.array_search(4);       // return -1
a.array_search(4.0);     // return 3
```

Array functions : array_search

```
a = [1,2,[3,2],{ "x" :1, "y" : "abc" } ,5];

a.array_search([3,2]);                    // return 2
a.array_search([3,2.0]);                  // return -1
a.array_search({"x":1,"y":"abc"});        // return 3
a.array_search({"x":1.0,"y":"abc"});      // return -1
```

```
a.array_search({"y":"abc", "x" : 1});
```

Array functions : contains

- Same as array_search but return true or false

```
a = [1,2,[3,2],{ "x" :1, "y" : "abc" } ,5];

a.contains([3,2]);                    // return true
a.contains([3,2.0]);                  // return false
a.contains({"x":1,"y":"abc"});        // return true
a.contains({"x":1.0,"y":"abc"});      // return false
```

Array functions : sort

- Sort an array using quick sort

```
sort([1.5,5,2,98,32,7,2,5]); // return
[1.5,2,2,5,5,7,32,98]
a = [1.5,5,2,98,32,7,2,5];
sort(a); a; // return a new array but a is not affected
// return [1.5,2,2,5,5,7,32,98][1.5,5,2,98,32,7,2,5]
```

- Sort float: `sort(sort:3, [1.1, 0.1, 1.3]);`
- Sort string: `sort(sort:1,['gt','ab', 'ba', 'cd']);`

sort functions

- bubble sort
- merge sort
- quick sort
- heap sort
- shell sort
- intro sort
- ....
- https://en.wikipedia.org/wiki/Sorting_algorithm

Array function : sort

- If parameter is not specify sort will check the type of element 0 of the array and use it to select function

```
 a = [1,2,3];
 function f(a,b)
     usera = sql(single: true, "select username from sn_users where uid = '?'", a);
     userb = sql(single: true, "select username from sn_users where uid = '?'", b);
     return usera.rows.username cmp userb.rows.username;
endf

sort(sort:2, fname: "f", a); // return [2,3,1]
```

Array function : exercice

```
function reverse_string(str)
  new_str = join(reverse(split(str, "")),"");
  return new_str;
endf

reverse_string("pierre");
```

Array function : exercise

- Write a sort function:

```
function mysort(arr)
  ...
  return new_arr;
endf


mysort([93,8,2,6]);
a = build_random_array();
a;
mysort(a);
```

```
function build_random_array()
  a = random(min:1,max:10,
    init:true);
  arr = array(a);

  for(i=0;i<a;++i) do
    arr[i] = 50 -
       random(min:0,max:100);
  endfor

  return arr;
endf
```