



SEDNOVE

Sncode/Extenso

pierre.Laplante@sednove.com

Version 1.2 : 2021-12-31

Course #1

- What you will see in this course:
 - Introduction about Extenso
 - Sednove
 - Extenso – Sncode
 - Architecture
 - Sncode programming

About this course

- Introduction to programming in Sncode
- Goal: Enable non-programmers to learn how to program, in particular with Sncode and Extenso
- What you will learn:
 - Sncode
 - Extenso
 - HTML
 - CSS
 - Javascript
 - JQuery
 - Ajax
 - Websocket
 - WebRTC

What you will need for this course

- HTML

- <https://www.youtube.com/watch?v=BvJYXI2ywUE>
- <https://www.youtube.com/watch?v=PypMN-yui4Y>
- https://www.youtube.com/watch?v=1rbo_HHt5nw
- <https://www.youtube.com/watch?v=bFvjE4ZRtSE>

- CSS / Bootstrap

- <https://www.youtube.com/watch?v=-qfEOE4vtxE>
- https://www.youtube.com/watch?v=1PnVor36_40

- Javascript

- <https://www.youtube.com/watch?v=cmIkfezTnBE&list=PL9dbBb7MI9bXwgPTH5STNGEQNQNeCDXdu>

What you will need for this course

- jQuery
 - <https://www.youtube.com/watch?v=hMxGhHNOKCU>
- JSON
 - <https://www.youtube.com/watch?v=iiADhChRriM>
- SQL MariaDB/Mysql
 - https://www.youtube.com/watch?v=p3qvj9hO_Bo
- REGEX : Regular Expression
 - <https://www.youtube.com/watch?v=rhzKDrUiJVk>
- Git
 - <https://www.youtube.com/watch?v=IHaTbJPdB-s>
- And programming experience in a language...

What you **may** need for this course

- Linux (CentOS). Basic Command line interface (CLI)
 - <https://www.youtube.com/watch?v=5jIIOkA0Npl>
- Apache configuration
 - <https://www.youtube.com/watch?v=rCr3-YIL5S8>
- C programming
 - cmake / make / gcc
- Websockets

About this course

- keep your personal question for later with me directly
- this is an introduction course not an advanced course.
- the course is recorded
- if you have a question, please raise your hand first
- you will need a headphones to speak

Sednove

- Founded in 1997 by Chantal Bilodeau and Pierre Laplante
- Web and mobile applications development
- Technological development
- Branding
- Design

Platinum

- Founded in 1995.
- Software developed in FoxPro
- Front Desk component only until 2000 (schedule, patient file, transactions and reports)
- EHR module in 2000 (doctor notes, patient flow with check in and calling patients to the room)
- in USA and around the world since 2002
- Need to move to the cloud to integrate new tools

Extenso / Sncode

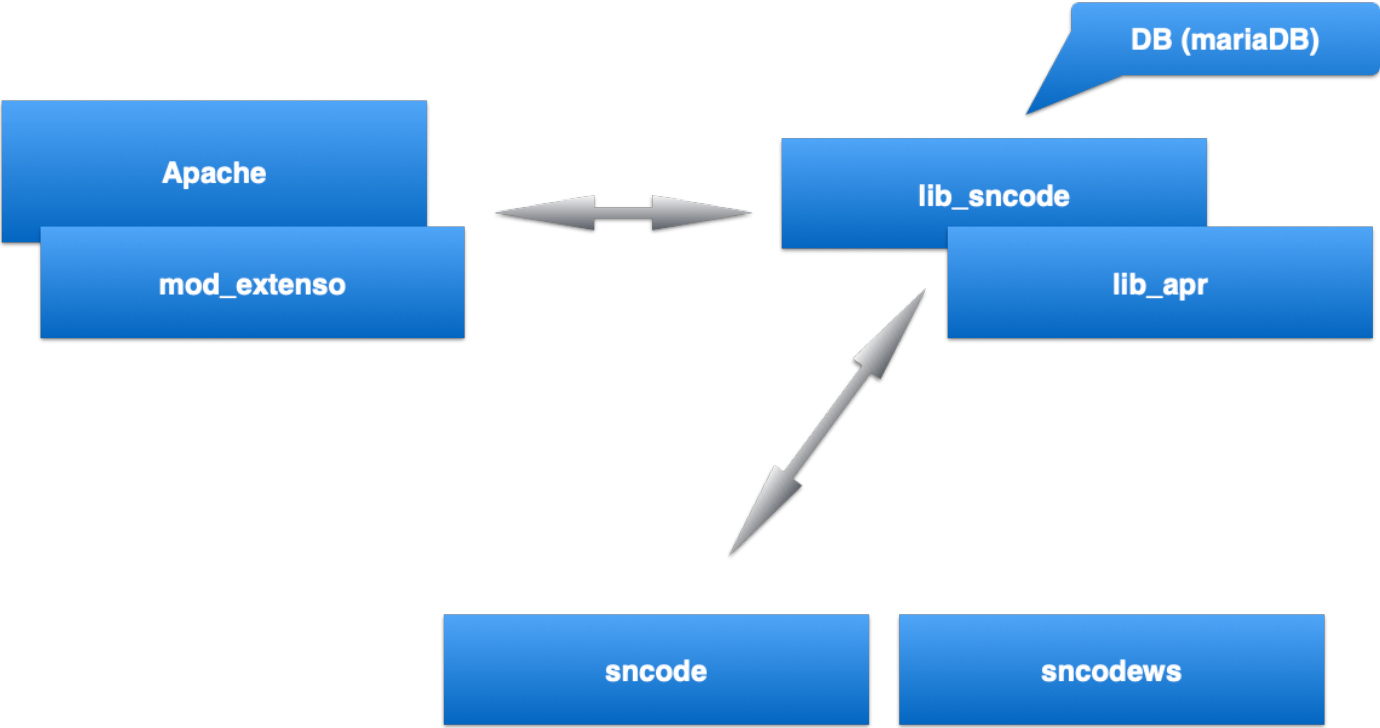
- Extenso : Clients and programmers interface
- Sncode : Programming language



Sednove's tools

- slack
- clickup
- uxpın
- gecko
- email
 - laplante@sednove.com
- Phone
 - 514-945-1779 (also whatsapp)

Architecture of Extenso



What is different about Extenso ?

- Dynamic / static
- Compile language
- Security with virtual machine
- Grid manager
- Style manager
- Modules manager
- Database manager
- Git/Gitlab management of modules

Sncode

- Key concepts:
 - Sncode is a compiled language
 - Uses a virtual machine to execute code
 - Rich and extensible library
 - External modules
 - Simple syntax for non programmer
 - Power-full for professional programmer

compile, virtual machine etc.

- Interpreted language, PHP, Ruby
- Compile language : C
- Virtual machine : Java, C#, Sncode
- JIT : Just in Time
- Assembler
- Machine code

Sncode concepts

- File with extension .sn are compile and execute
- File with extension .snc are file already compiled in binary
- Convention use in the naming of site:

<https://ssnode.sednove.com> for the staging version

<https://sncode.sednove.com> for the production version

Sncode concepts #2

- staging require a login to modify the site
- Production deployment (or publishing) is the process of copying the file from staging to production
- Directory for staging is /staging
- Directory for html is /html

Sncode

- In a page everything that is not between {{ and }} is print as-is
- {{ Start Sncode
- }} End Sncode
- All text outside of Sncode is returned to the browser without being parsed

IDE

- Introduction to IDE in Extenso
- How to execute a program:
 - In staging : <https://ssncode.sednove.com/ex.sn>
 - In html : <https://sncode.sednove.com/ex.sn>

Simple example

- ```
<html>
 <body>
 <h1>{{ "Date is "; datetime(); }}</h1>
 </body>
</html>
```

# Exercices

- Exercice #1 : try to reproduce the previous example.
  - Create a file in /staging/ex1.sn
  - Execute it with the URL <https://ssncode.sednove.com/ex1>
- Exercice #2 Only display the time not the date

PS : <https://extenso.live>

- PS #2 <https://getbootstrap.com/docs/4.5/getting-started/introduction/>

# Sncode's documentation

- All documentation under <https://extenso.live>
- <https://module.sednove.com>
- Man pages under Linux:

`man sql`

# Sncode's types

- Integer : `{{ a = 5 ; }}`
- Double : `{{ a = 3.1415 ; }}`
- Array : `{{ a = [ 1, 2.0, "3.1415" ] ; }}`
- Boolean : `{{ a = true ; }}`
- Null : `{{ a = null ; }}`
- Undefined : `{{ a = undefined ; }}`
- Associative Array / Hash array / Context :  
`{{ a = { "x" : 1.5, "y" : 2.0 } ; }}`
- `a.type()` will return the type of variable a

# Integer

- Try This program:

```
{ {
 a = 5;
 b = a / 2;
 "b = "; b;
 type (b) ;
} }
```



# Operators

- By order of priority:

- +, -,
- \*, /,
- \*\* (power), % (modulo)

- Try:

$$2 + 3 * 4 ** 2 \Rightarrow 50$$

- Use () to modify the order:

$$((2 + 3) * 4) ** 2 \Rightarrow 400$$



SEDNOVE

Sncode/Extenso

[pierre.Laplante@sednove.com](mailto:pierre.Laplante@sednove.com)

Version 1.4 : 2020-07-03

# Course #2

- What we have seen so far:
  - Extenso presentation
  - How to use IDE
  - Sncode's type
  - Structure of directory in Extenso

# Integer / float

- Try this program:

```
{ {
 a = 5;
 b = a / 2.0;
 "b = "; b;
 type (b) ;
} }
```

# Function printf

- `printf` : print formatted
- `printf("Number = %05d", 10);`
- `d` : use to print integer
- Try
  - `printf("%7d", 10); a=printf("%x", 10); a;`
  - `printf("%-10s", a); printf("%10.4f", 10.2);`
  - `printf("%010.4f", 10.2); printf("%+010.4f", 10.2);`
  - `printf("%+10.4f", 10.2); printf("%10.1f", 5.17);`
- `f` : float, `s` : space, `x` : hexadecimal

# Floating point number

- Example:

```
a = 1.123456789;
a;
```

- By default snprintf use: `printf("%.8f", number)`
- According to Wikipedia:

"In [computing](#), **floating-point arithmetic (FP)** is arithmetic using formulaic representation of [real numbers](#) as an approximation to support a [trade-off](#) between range and [precision](#)."

# Floating point number

- Floating point number are represented as double in C
- Try:

```
{ { printf("%.20f", 0.1+0.2); } }
```

- Check:

<https://docs.python.org/3/tutorial/floatingpoint.html>

[https://doc.lagout.org/science/0\\_Computer%20Science/3\\_Theory/Handbook%20of%20Floating%20Point%20Arithmetic.pdf](https://doc.lagout.org/science/0_Computer%20Science/3_Theory/Handbook%20of%20Floating%20Point%20Arithmetic.pdf)

# Floating point number

- Try this program:

```
a = 1/5;
b = 1/5.0;
c = 1.0/5;
"a="; a; ", b="; b; ", c ="; c;
```



# Floating point number

- Try:

```
a = 48.0 * atan(1.0/49.0) + 128.0 *
 atan(1.0/57.0) -
 20.0 * atan(1.0/239.0) + 48.0 *
 atan(1.0/110443.0);

printf("%.25f", a);
```

# Floting point number comparaison

- Floating point comparaison : operator ==
- try:

```
a=0.15+0.15; // 0.3000000000000155
b=0.1+0.2; // 0.3000000000000144
a==b;
```

- return !

```
false
```

# Floating point number

```
function compare(a,b)
 //!code Minimal function to compare FPN to 0.0001
 if a==b then
 return true;
 endif

 if abs(abs(a) - abs(b)) < 0.001 then
 return true;
 endif
 return false;
endf
a = 0.15+0.15; b = 0.1 + 0.2; a==b; compare(a,b);
```

# Comparison operators

- $<$  : less than
- $>$  : greater than
- $\leq$  : less or equal
- $\geq$  : greater or uqual
- $\Leftrightarrow$  : compare  $1 \Leftrightarrow 2$ ;  $2 \Leftrightarrow 1$ ;
- $\neq$  : not equal

## FPN example

```
if 1 == 2 then
 "Oh la la something is wrong here";
else
 "Ok 1 is not equal to 2";
endif
```

# FPN compare with string

- String are automatically convert to double

```
a = "0.0001";
```

```
b = 0.0001;
```

```
if a == b then
```

```
 "a is equal to b";
```

```
else
```

```
 "a is not equal to b";
```

```
endif
```

- Test:

```
if "0" == 0 then "true"; else "false"; endif
```

# Comparison operator

- If we do:

```
a = 5 < 6; a;
```

- a is a boolean : true or false

```
if a then "true part"; else "false part"; endif
```

- If we do:

```
a = 1;
```

```
if a then "true part"; else "false part"; endif
```

- 1 is true and 0 is false

# The art of programming... part #1

- Good indentation
- Use good meaningful variable name :english, lower case, no plural \_
- Use comment:
  - `/* ... */`
  - `/*`
  - `...`
  - `...`
  - `*/`
  - `// comment`
  - `# comment`



# Comparison operator : <=>

- Use to compare 2 numbers
- Return
  - -1 if left then
  - 0 if equal
  - 1 if greater than
- Example using a new statement : switch

```
a = 60;
```

```
b = 5;
```

```
switch a <=> b do
```

# Comparison operator : <=>

- Use to compare 2 numbers
- Return
  - -1 if left then
  - 0 if equal
  - 1 if greater than
- Example using a new statement : switch

```
a = 60;
```

```
b = 5;
```

```
switch a <=> b do
```

# Comparison operator : <=>

- Use to compare 2 numbers
- Return
  - -1 if left then
  - 0 if equal
  - 1 if greater than
- Example using a new statement : switch

```
a = 60;
```

```
b = 5;
```

```
switch a <=> b do
```

## Comparaison operator : <=>

**case** -1:

"a is lower than b";

**endc**

**case** 0:

"a is equal to b";

endc

**case** 1:

"a is greater than b";

**endc**

**default:**

"Ohh la system error";

endc

**ends**



SEDNOVE

Sncode/Extenso

[pierre.Laplante@sednove.com](mailto:pierre.Laplante@sednove.com)

Version 1.4 : 2020-07-03

# Course #3

- What we have seen in course #2
  - Floating point number
  - Comparison operators
  - Comments

# Boolean

- 2 values: true or false
- Examples:

```
a = true; b = false;
```

```
a != b;
```

```
a == b;
```

```
a < b;
```

```
a > b;
```

# String

- A string variable is created using quote or double quote
- Example:

```
a = "Pierre";
a;
a= 'Pierre';
a;
```

- Why single quote or double quote ?



# String : double vs single quote

- Double quote support escape sequence
- Ascii characters : `a = "Pier\x43\x44"; return PierCD;`
  - 43 is hex value of character C
  - 44 is hex value of character D
- UTF-8 characters: `a = "\ucf80 = 3.1415"; a;`  
`return  $\pi$  = 3.1415`
- Complete list of utf-8 character:

<http://www.fileformat.info/info/charset/UTF-8/list.htm?start=1024>

# String : escape sequences

- `\\` : display `\`
- `\n` : newline
- `\a` : alert beep bell
- `\b` : backspace
- `\t` : tab
- `\r` : carriage return
- `\v` : vertical tab
- `\o` : octal number
- `\f` : Formfeed Page Break
- `\'` ou `\"` : Display ' or "

## String : when " and ' are not enough

- `q( ... )` : quote is ( and )
  - `q( x=' "\t );`
- `dq(( ... ))` : quotes are 2 characters (( and ))
  - `dq([ x=' "\t ]);`
- `qq( ... )` : double quote are ( and )
  - `qq( x=' "\t );`
- `dqq(( ... ))` : double quotes are 2 characters (( and ))
  - `dqq(( x=""\t ));`

# String : quote and double quotes

- You can also use the following characters:
- /, #, @, !, \$, %, ?, &, \
  - q/abc/;
- ( and ), [ and ], { and }, < and >
  - q(abc);
- /, #, @, !, \$, %, ?, &, \
  - dq/edghe/;

# String comparison

- `a = "001"; b = "1";`

`a == b; // will convert a and b to double before doing the comparison`

or try

`a eq b; // string are compare case sensitive`

- Sncode knows that a is a string and b is string or convert them
- `type(a); type(b);`

# String comparison operators

- eq : equal(==)
- ne: not equal (!=)
- lt : less than (<)
- le : less or equal (<=)
- gt : greater than (>)
- ge : greater or equal (ge)
- st : start
- ns : not start
- cmp : compare ( $\Leftrightarrow$ )

# String comparison st and ns

- Can you figure out what is the use of st and ns ?

# String comparison st and ns

- st stands for start with
- ns stands for not start with
- Example:

```
a = "/usr/local/website/plv1/staging/tmp";
a st "/usr/local"; // true
a st "usr/local"; // false
a ns "usr/local"; // true
```



# String comparison and date

- SQL usually return date in the military format:
  - YYYY-MM-DD HH:MM:SS
- String comparison can then be used to compare date and time

```
a = "2020-12-14 18:32:33";
a < "2020-12-15 19:19:19"; // return true
```

# String and sub-string

- You can use the [] operator to get substring of a string:

```
a = "Pierre Laplante";
```

```
a[0:5]; // return Pierr
a[1:5]; // return ierr
a[:5]; // return Pierr
a[5]; // return e
a[7:]; // return Laplante
a[:-2]; // return Pierre Laplan
a[-4:-2]; // return an
```

# String operator

- To concatenate 2 string use the operator .+

```
a = "pierre " .+ "laplante";
```

```
a; // return pierre laplante
```

```
b = "pierre " .+ 35;
```

```
b; // return pierre 35
```

## String exercise

- Write a program to reverse the string `a="pierre"`; `b`
- `length(string)` return the length of a string  
`name = "Etienne"`;

```
b = name[6:7];
```

```
b = b .+ name[5:6]; // b .+= name[5:6] a = 7; a+= 7;
```

```
b = b .+ name[4:5];
```

```
b = b .+ name[3:4];
```

```
b = b .+ name[2:3];
```

```
b = b .+ name[1:2];
```

```
b = b .+ name[0:1];
```

## String reverse

```
a = "pierre laplante";
b = "";
len = length(a);
for(i=0; i<len; ++i) do
 b = a[i:i+1] .+ b;
endfor
b;
```

## String functions : **strnatcmp**

- **strnatcmp** - Case insensitive string comparisons using a "natural order" algorithm.
- Natural order means that rather than solely comparing single character code values, strings are ordered in a natural way. For example, the string "hello10" is considered greater than the string "hello2" even though the first numeric character in "hello10" actually has a smaller character value than the corresponding character in "hello2". However, since 10 is greater than 2, strnatcmp will put "hello10" after "hello2".
- `strnatcmp("hello10", "hello2"); // return 1`

## String functions : addslashes

- Escape the following characters:  
single or double quote, the backslash and the null character.
- `addslashes ("\\ \" ' " ); // return \\ \\ \" \'`

## String functions : base64\_encode/decode

- **base64\_encode**(data,modified) Encodes data with MIME base64
- if modified is false then the character set supported are ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789:-
- If modified is true then the character set supported are ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/  
• `base64_encode("Pierre Laplante",true);`
  - `return UGllcnJlIExhcGxhbnRl`





SEDNOVE

Sncode/Extenso

[pierre.Laplante@sednove.com](mailto:pierre.Laplante@sednove.com)

Version 1.4 : 2020-07-03

# Course #4

- What we have seen in course #3
  - Boolean
  - String
  - String 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](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](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]

a[0] = "value";
```

# 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

- **Assignment:**

```
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 assignment**

- `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
last(a);
```

- 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 join(a, ";")1
```

- 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)
```

```
 return new_str;
```

```
endf
```

```
reverse_str("pierre"); // erreip
```

# 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]
```

```
for i in a do
```

```
 i;
```

```
endfor
```

## Array functions : array\_search

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

```
a = [1,2,3,4.0,5, "abc", true, [1,2]];
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](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
```



SEDNOVE

Sncode/Extenso

[pierre.Laplante@sednove.com](mailto:pierre.Laplante@sednove.com)

Version 1.4 : 2020-07-03

# Course #5

- What we have seen in course #4
  - POST vs GET
  - Function cgidata
  - Array and array functions



# Hash / associative / context array

- In [computer science](#), an **associative array**, **map**, **symbol table**, or **dictionary** is an [abstract data type](#) composed of a [collection](#) of [\(key, value\) pairs](#), such that each possible key appears at most once in the collection.
- Based on JSON structure
- <https://www.json.org/json-en.html>
- Example:  

```
a = { "x" : [1,2], "pi" : 3.1415 };
```

# Hash array

- If we have:

```
a = { "x" : [1,2], "pi" : 3.1415 };
```

```
a.x; // return [1,2]
```

```
a.pi; // return 3.1415
```

```
a{"x"}; //return [1,2]
```

```
a{"p" .+ "i"}; // return 3.1415
```

```
a.x[0]; // return 1 a{"x"}[0];
```

```
a.x[1]; // return 2 a{"x"}[1];
```

# Hash array

- If we have:

```
a = { "x & y" : [1,2], "pi" : 3.1415 };
```

```
a.x & y; does not work but
```

```
a{"x & y"}; will return [1,2]
```

```
a{"p & l"} = { "p" : 1, "l" : 56 };
```

```
a; will return
```

```
{"x & y":[1,2], "pi":3.1415, "p & l":{"l":56, "p":1}}
```

## JSON / Hash array

- Hash array and JSON are completely compatible in Sncode
- With javascript you can do:

```
{{ a = { "x" : [1, 2] }; }}
```

```
<script>
```

```
 let a = {{ a }};
```

```
 console.log(a);
```

```
</script>
```

## Hash functions : clearctx

- clearctx : Clears everything in a context.

```
a = { "x" : 1 };
```

```
clearctx(a);
```

```
a; // return {}
```

## hash functions : deletectx

- Delete an entry in a context.

```
a = { "x":1, "y":2};
deletectx(a, "x");
a; // return {"y":2}
```

## Hash functions : **exist**

- Check if an element exist in a context.

```
a = { "pi" : 1 };
```

```
a.exist("pi"); // return true
```

```
a.exist("PI"); // return false
```

```
a.exist('pi') exist(a, 'pi')
```

```
if a.pierre != undefined then "ok"; endif
```

```
a;
```

## hash functions : keys

- Return an array of the keys of the hash array

```
a = { "x":1, "y":2};
keys(a); // return ["x", "y"];
```

- This can be use to loop in the hash array:

```
a = { "x":1, "y":2};
for k in sort(keys(a)) do
 k; "="; a{k}; " ";
endfor
// return y=2 x=1
```



## Hash array : Note about the order

- Keys in hash array do not have an order.

```
a = { "x" : 1, "y" : 2 };
```

```
keys(a); might return
```

```
["x", "y"] or ["y", "x"]
```

## Hash functions : values

- **values** - Returns the list of values of a context.

```
a = { "x" :1, "y" : 2};
```

```
values(a); // return [1,2]
```

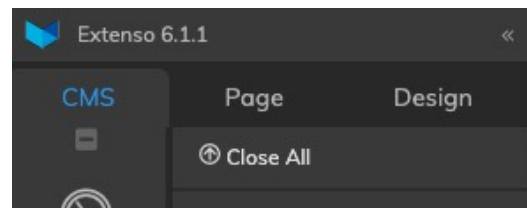
## Hash functions : both

- **both** - Returns the list of keys and values of a context.

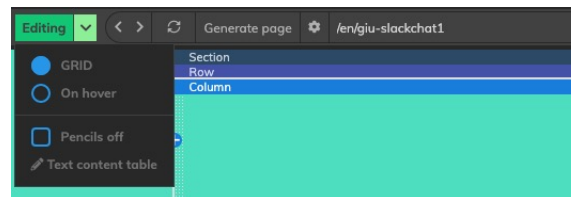
```
a = {"x" : 1, "y" : 2};
both(a); // return
[{"key":"y", "value":2}, {"key":"x", "value":1}]
for i in both(a) do
 i.key; "="; i.value;
endfor
// return x=1y=2
```

# Extenso : a brief introduction

- The interface
- The left menu
  - CMS
  - Page
  - Design

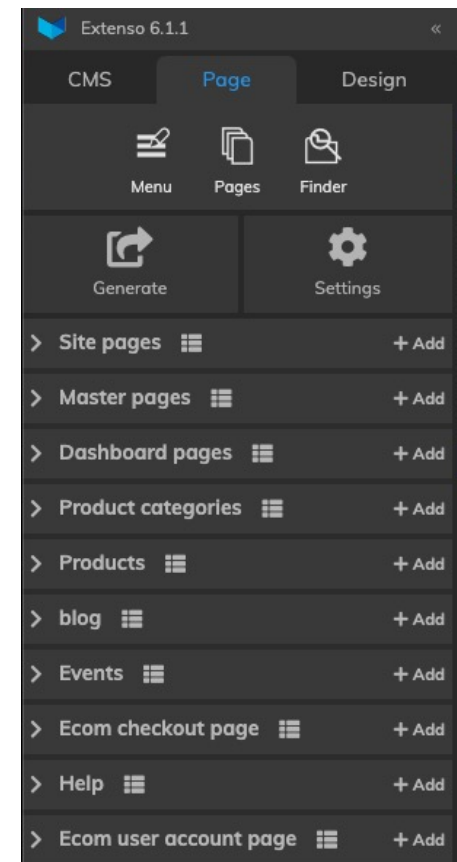


- The right menu (displayed in Page or Design)
- The top menu



# Extenso : A brief introduction

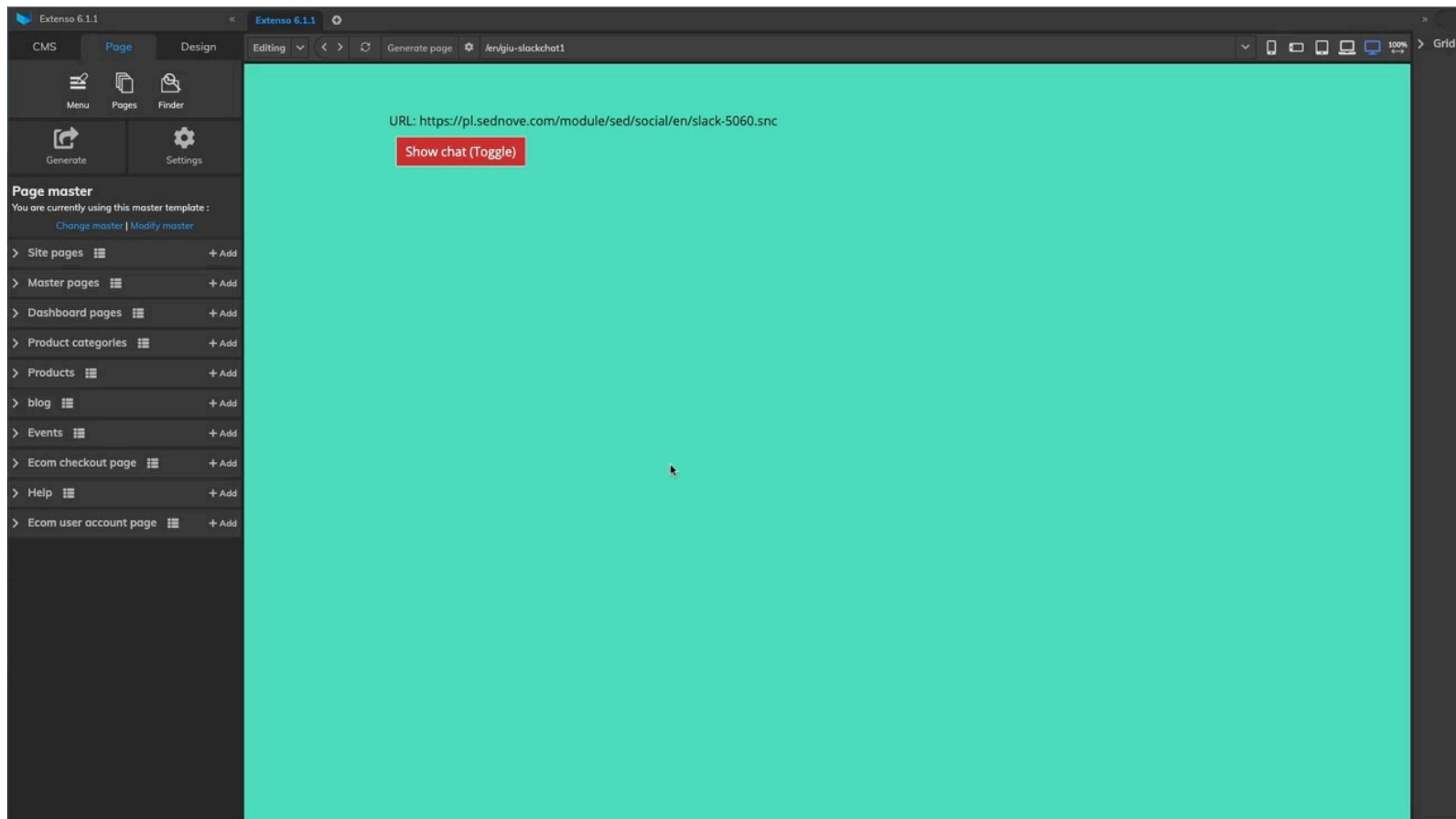
- Master Page
- Site Pages
- Other pages



Extenso : A brief introduction : create a new master page

- [https://extenso.live/sn\\_uploads/cours/create-blank-master.mp4](https://extenso.live/sn_uploads/cours/create-blank-master.mp4)

# Extenso : Create a blank master



Enregistrement automatique ACTIVE | Sncodev3 — Enregistré

Nagios | gecko | Merge Requests - Dashboard | Empêcher le blocage ou le pla... | Extenso Sednove | https://extenso.live/sn\_upload... | Nagios

spl.sednove.com/extenso/secure/module/sed/core/en/index.snc

Applications | Manage DNS | Sednove | Nagios | OpenSRS | GURU - Login | Facebook | Sednove | Google Agenda | Gecko | The W3C Markup... | MetroOptic Login | QuickBooks | Bookmarks | Plesk 12.0.18 | SSL Analyzer and... | Alibaba | Autres favoris

Extenso 6.1.1 | Extenso 6.1.1 | Editing | Generate page | /extenso/secure/master/en/blank1

CMS | Page | Design

Menu | Pages | Finder

Generate | Settings

- > Site pages + Add
- > Master pages + Add
- > Dashboard pages + Add
- > Product categories + Add
- > Products + Add
- > blog + Add
- > Events + Add
- > Ecom checkout page + Add
- > Help + Add
- > Ecom user account page + Add

Section

Row

Column

Use to instantiate a grid in a master page

# Widget grid

WIDGET LIST

Add a section

Grid explorer

Body

- Section
  - Row
    - Column
      - Use to instantiate a grid in a master p...
        - page

Add a section

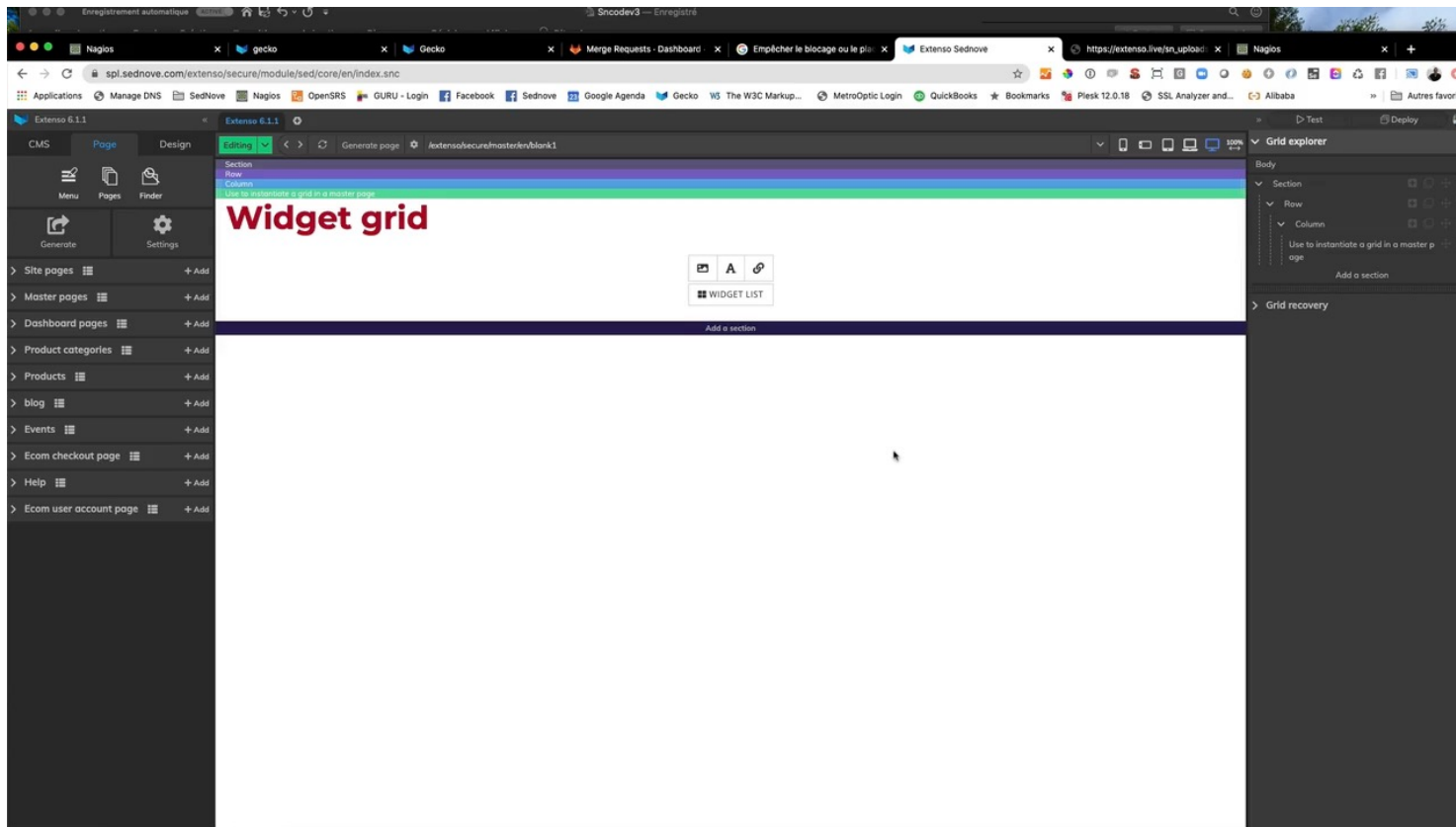
Grid recovery



## Extenso : create a site page

- We create a new page using the new master blank page
- [https://extenso.live/sn\\_uploads/cours/create-site-page.mp4](https://extenso.live/sn_uploads/cours/create-site-page.mp4)

# Create a site page



# Exercise

- Create a blank master page with grid widget
- Create a new site page based on this master page
- Add a text widget to your site page

## Sncode : IF ELSEIF THEN ENDIF

```
if x == 1 then
 ...
elseif y eq "string" then
 ...
else
 ...
endif
```

## Sncode : logical operator && || and !

- && for and
- || for or
- ! for not

```
if x == 5 && y == 5 then
 "Both x and y are equal to 5";
endif
```

```
if (! x == 5) && y == 5 then
 "x is not equal to 5 and y is equal to 5";
endif
```

# NO

- `if x == 5 && y == 5 then`
- `if x && y == 5 then`

## Sncode : Priority of operators and grouping with ()

```
if ! (x == 5 && y == 5) then
 "x or y are not equal to 5";
endif
```

() can be used to change the order or priority

```
if x == 5 then
if x eq "5" then
```

---



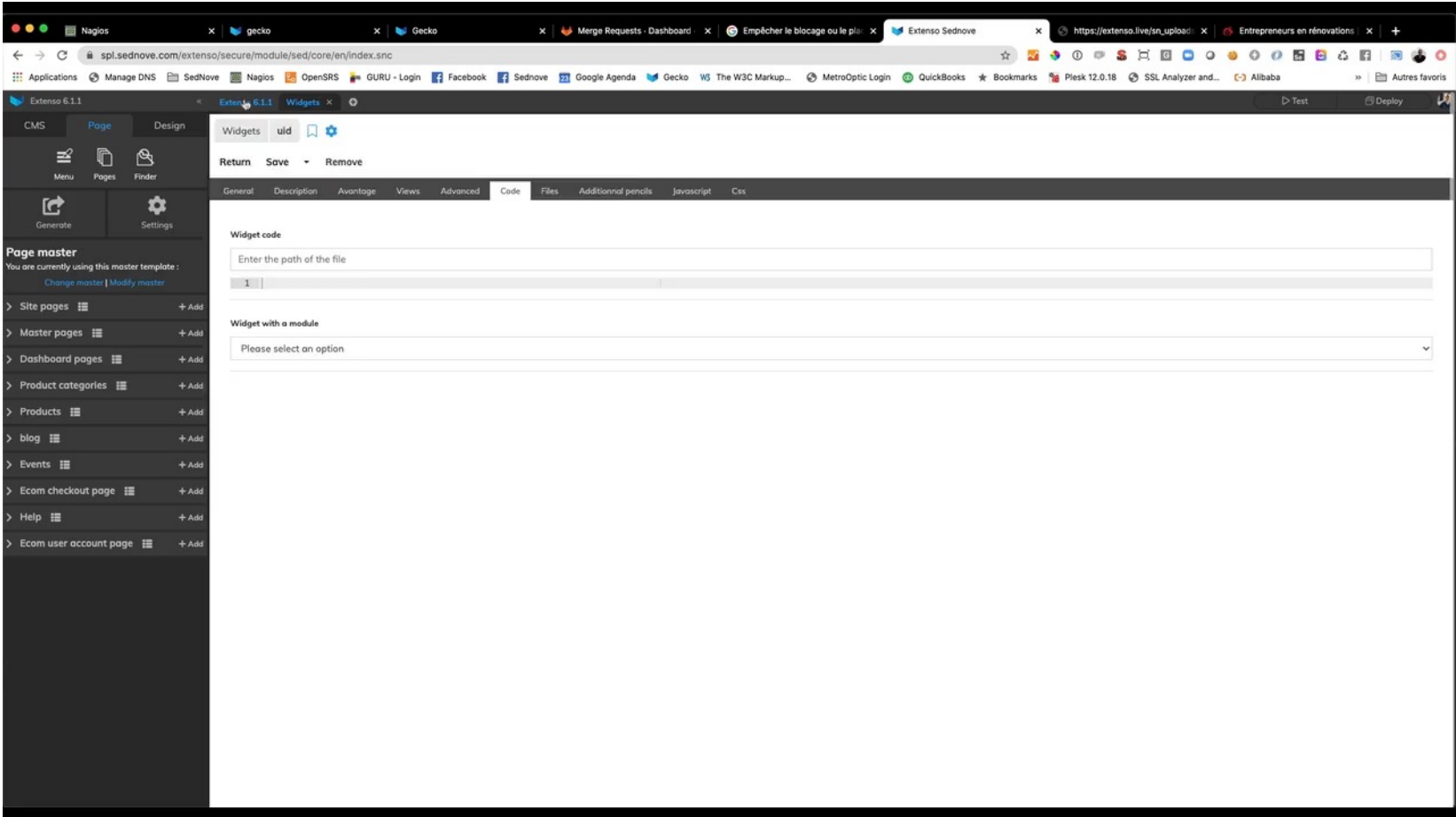
Extenso : Create a  
widget



# Extenso : Widget

- CMS / DEV / Module Managment / Widgets
- Click add
- Click Active
- Enter code name : w1
- Select category : Custom made
- Specify a widget name in fr and en

# Video



# Extenso : Widget

- Goto tab code
  - Type /site/widget/w1.sn
  - Type `<h2>Widget W1</h2>`
- Click Save and return
- Add the widget to your page
- Video: [https://extenso.live/sn\\_uploads/cours/add-widget.mp4](https://extenso.live/sn_uploads/cours/add-widget.mp4)



SEDNOVE

Sncode/Extenso

[pierre.Laplante@sednove.com](mailto:pierre.Laplante@sednove.com)

Version 1.4 : 2020-07-03

# Course #6

- What we have seen so far:
  - How to use IDE in Extenso
  - Sncode's type
    - Integer
    - Float
    - String
    - Array
    - Associative Array / hash array / context / JSON
  - Extenso introduction
  - Creating a widget

# Exercise

- Create your widget W1
- Create another widget W2 that will display the current datetime()
- When you reload the widget, does it display the current time ?
- Why
- Dynamic / static widget
- When you generate a widget the code of the widget is executed and the resulting widget is compiled
- So how to you make the code dynamic: use \ in front of {{
- \{{ datetime() }}

# Exercice

- Create another widget w3:
  - Display date & time when the widget is generated
  - Display current date & time
- Create a widget w4 that will display:  
<h1>H1</h1>....<h6>...</h6> with a loop in Sncode
  - A loop can be done with  
while i<6 do ... endw
  - Or  
for(i=1;i<=6;++i) do .... endfor

# Exercice

- Create a widget that will display in a table:

username firstname lastname email number of login (right align)

- To get the data:

```
data = sql("select username, firstname, lastname, email, nb_login from
sed_login_user order by username");
```

- A table can be build in bootstrap:

<https://getbootstrap.com/docs/4.0/content/tables/>



# Exercice

- Create a new widget widget2.sn
- Display a form with the following text field:
  - firstname
  - lastname
  - submit
- Save the information in a new table call widget2
  - firstname
  - lastname

```
sql("insert into widget2 (firstname,lastname) values('?', '?')",
cgidata.firstname, cgidata.lastname);
```
- Display the content of the table widget2 dynamically