ALIENS PROGRAMMING LANGUAGE

Documentation

Version 1.0

Friday, August 28, 2020

Chapiter 0: What is ALENS PL ?

1- Introduction:

we have made this language to make programming as easy as possible for new learners, who have no experience in coding.

we coined this language as ‘Aliens ' programming language and the abbreviation is "Als” which we will use a lot in our programming journey.

we also designed this language to be familiar to the user especially those who already have an experience in coding, we got inspired the structure and the syntax from the famous and successful languages such as (dart, python, C#, JavaScript ...), so it inherits many of the same statements and expressions form those.

2- The reason why we name our language "Aliens Programming Language":

before talking extensively about the syntax and the structure of the language we should put you in the ground and reasons behind choosing this name for our language:

On 10 August 2020, our team participated in one of the biggest events that Repl.it organized with a prize of 10000 $ for the winning team.

our team started the brainstorming on how our language should look like after hours of negotiation and discussion, we finally agreed that the language should present something that we all know, even those how never used a computer before. the idea is space.

you must be wondering what is the relevance of "space" with computer language, alright we will explain.

the first thing you should know is the Aliens language hierarchy, well.

imagine yourself you are cruising space; in your way you may discover new planets or new galaxies. how knows everything is possible over there, so the thing is our language converts all this word or space jargon if we could say to names that have a meaning in our language.

eg: a user could create a planet with its moons and of course inside a space which contains many galaxies, it is simple isn't it?

we think now it's become more obvious for you where the name of "Aliens programming language” came from.

3- Pre-requirements:

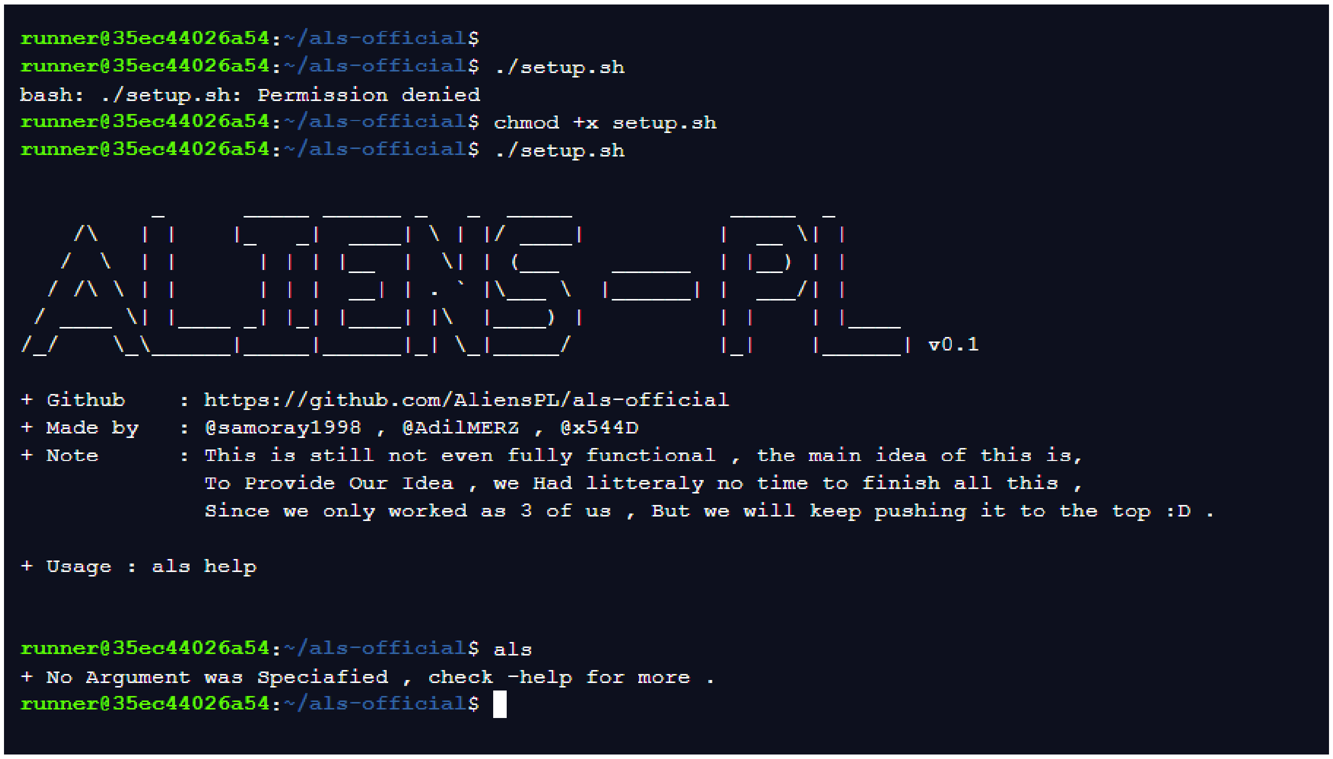
All you need to make this language work is https://www.python.org/

, and Linux.

“Unfortunately, this language only works in Linux for the moment.“

4- Setup Aliens PL

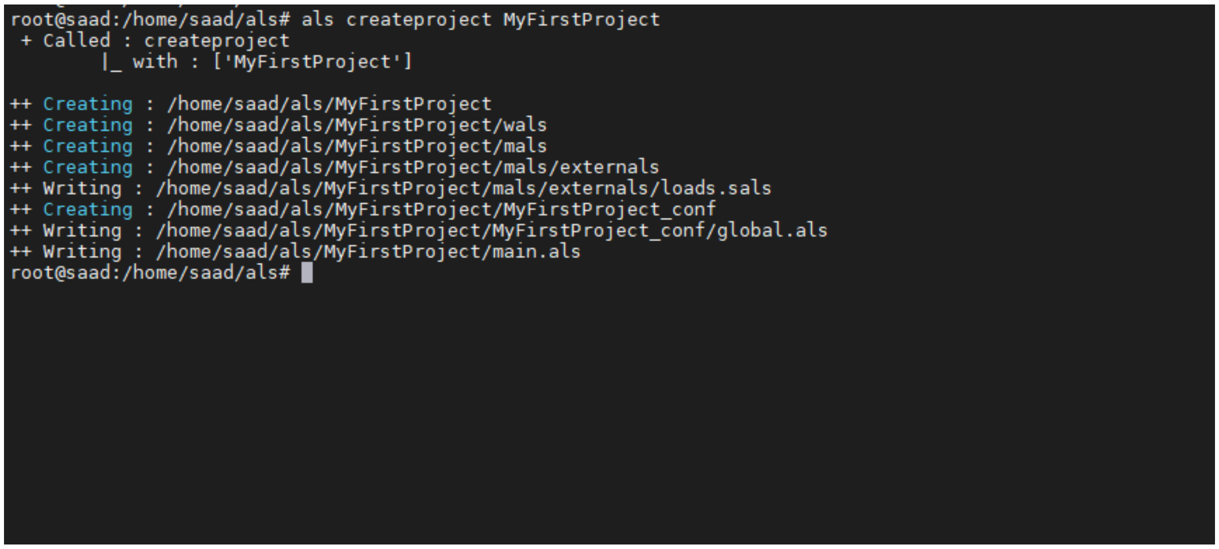
Now let create our first project which we will name as myfirstapp:



```
runner@35ec44026a54:~/als-official$
runner@35ec44026a54:~/als-official$ ./setup.sh
bash: ./setup.sh: Permission denied
runner@35ec44026a54:~/als-official$ chmod +x setup.sh
runner@35ec44026a54:~/als-official$ ./setup.sh

          _      _____ ______ _   _  _____            _____  _
    /\   | |    |_   _|  ____| \ | |/ ____|          |  __ \| |
   /  \  | |      | | | |__  |  \| | (___    ______  | |__) | |
  / /\ \ | |      | | |  __| | . ` |\___ \  |______| |  ___/| |
 / ____ \| |____ _| |_| |____| |\  |____) |          | |    | |____
/_/    \_\______|_____|______|_| \_|_____/           |_|    |______| v0.1

+ Github     : https://github.com/AliensPL/als-official
+ Made by    : @samoray1998 , @AdilMERZ , @x544D
+ Note       : This is still not even fully functional , the main idea of this is,
               To Provide Our Idea , we Had litteraly no time to finish all this ,
               Since we only worked as 3 of us , But we will keep pushing it to the top :D .

+ Usage : als help


runner@35ec44026a54:~/als-official$ als
+ No Argument was Speciafied , check -help for more .
runner@35ec44026a54:~/als-official$ 
```

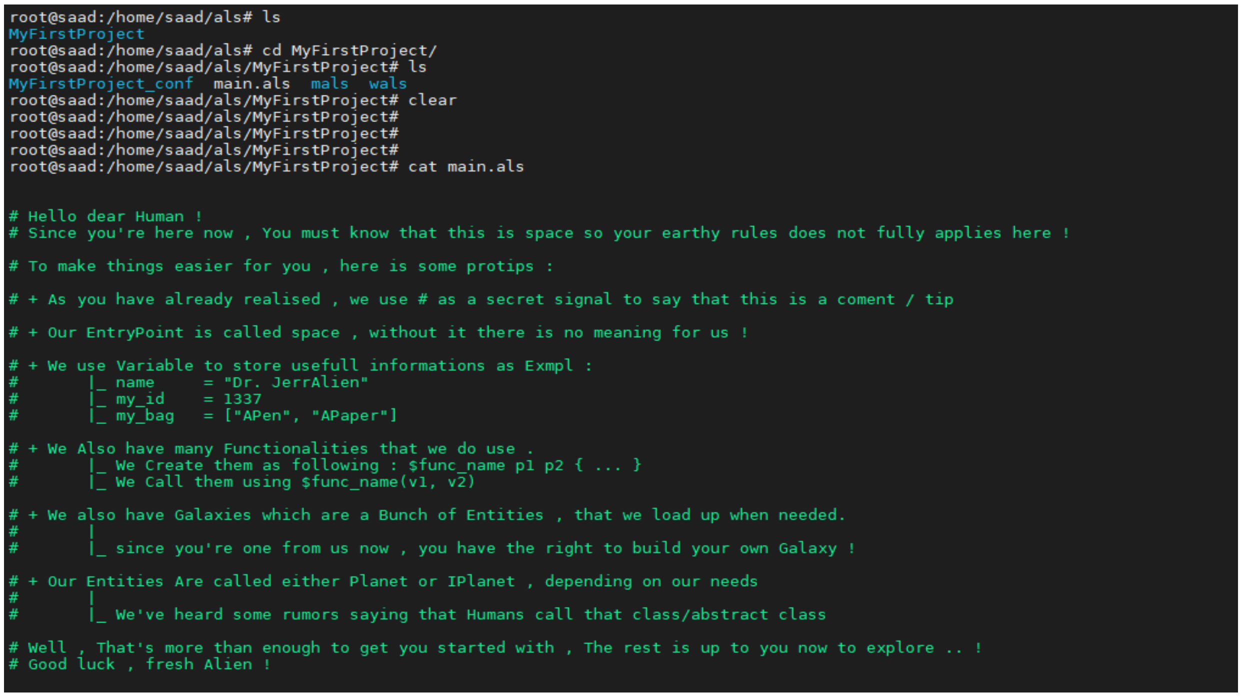
5- Create new project:

Now let create our first project which we will name as myfirstapp:



```
root@saad:/home/saad/als# als createproject MyFirstProject
 + Called : createproject
        |_ with : ['MyFirstProject']

++ Creating : /home/saad/als/MyFirstProject
++ Creating : /home/saad/als/MyFirstProject/wals
++ Creating : /home/saad/als/MyFirstProject/mals
++ Creating : /home/saad/als/MyFirstProject/mals/externals
++ Writing : /home/saad/als/MyFirstProject/mals/externals/loads.sals
++ Creating : /home/saad/als/MyFirstProject/MyFirstProject_conf
++ Writing : /home/saad/als/MyFirstProject/MyFirstProject_conf/global.als
++ Writing : /home/saad/als/MyFirstProject/main.als
root@saad:/home/saad/als#
```

As you see in the example above, we created our "myfirstapp" project successfully, inside Test directory we have the file main.als as well. you noticed the keyword space. this is our entry point which will be responsible for displaying this:



```
root@saad:/home/saad/als# ls
MyFirstProject
root@saad:/home/saad/als# cd MyFirstProject/
root@saad:/home/saad/als/MyFirstProject# ls
MyFirstProject_conf  main.als  mals  wals
root@saad:/home/saad/als/MyFirstProject# clear
root@saad:/home/saad/als/MyFirstProject#
root@saad:/home/saad/als/MyFirstProject#
root@saad:/home/saad/als/MyFirstProject#
root@saad:/home/saad/als/MyFirstProject# cat main.als


# Hello dear Human !
# Since you're here now , You must know that this is space so your earthy rules does not fully applies here !

# To make things easier for you , here is some protips :

# + As you have already realised , we use # as a secret signal to say that this is a coment / tip

# + Our EntryPoint is called space , without it there is no meaning for us !

# + We use Variable to store usefull informations as Exmpl :
#       |_ name     = "Dr. JerrAlien"
#       |_ my_id    = 1337
#       |_ my_bag   = ["APen", "APaper"]

# + We Also have many Functionalities that we do use .
#       |_ We Create them as following : $func_name p1 p2 { ... }
#       |_ We Call them using $func_name(v1, v2)

# + We also have Galaxies which are a Bunch of Entities , that we load up when needed.
#       |
#       |_ since you're one from us now , you have the right to build your own Galaxy !

# + Our Entities Are called either Planet or IPlanet , depending on our needs
#       |
#       |_ We've heard some rumors saying that Humans call that class/abstract class

# Well , That's more than enough to get you started with , The rest is up to you now to explore .. !
# Good luck , fresh Alien !
```

1- Input & output

```
name=$in()  #$in() takes a input from the user

$out(‘Hello  ’+name)   #$out  print value

#the result will be something like this:

---------------------------------------------------------------------------------------------------------------------------------------

Aliens

Hello Aliens
```

2- Variables in Aliens

variables are important to store your information temporarily in the computer ram in order to use them again, there is a lot of types:

a.Numbers

we use this type of variables when we want to represent both integer and floating-point numbers.

```
a=12+20

$out(‘the result of this addition is ’+a)

#the result will be something like this:

---------------------------------------------------------------------------------------------------------------------------------------

The result of the addition is 32
```

As you see in the example above we declare a variable which we called a to get the addition result of two integers, after that we used show function to display the result to the user.

Note:

We use **#** this symbol to say this is only comment.

b.Strings

we use this type of variables when we want to represent any kind of text, but it must be surrounded by quotes. Or double quotes.

```
Myname='Adam'

$out('my name is '+Myname)

#the result will be something like this.

----------------------------------------------------------------------------------------------------------------------------------------

My name is Adam
```

What we did here is no difference from what we did in the example above of number so we declare a variable of type string, after that we display it .

c. Lists:

A list is a type of object used for storing multiple values in single variable ,Each value (also called an element or item) in a list has a numeric position, known as its index, and it may contain data of any data type-numbers, strings, Booleans … and even other lists or dictionaries . The List index starts from 0, so that the first array element is lis[0] not lis[1].

```
Countries = [‘morocco’,‘usa’,’Canada’, ‘France’]    #this is a list of countries

$out(“I’m from “+Countries[0])

----------------------------------------------------------------------------------------------------------------------------------------

I’m from morocco
```

As you see the result of this example is “Morocco”, this because we choose the first element .

Chapter 3: More Control Flow Tools

1- Conditions

A condition is an expression that evaluates whether something is true or false. When the value of a condition is true, we say that this condition is satisfied.

a.If condition statement

```
If (true) {

$out( “hello I\’m from earth”)

}




-----------------------------------------------------------------------------------------------------------------------------------------

Hello i’m from earth
```

In this case the condition is true, so it displays” hello I’m from earth” as a result .

b.If/else conditions statements

```
If (false) {

$out( «hello I’m from earth »)

} :{

$out(« hey I’m from mars »)

}

-----------------------------------------------------------------------------------------------------------------------------------------

hey I’m from mars
```

in this example we have two conditions, the first one is if (false) which is not true the second one represents “else” of the “if” condition which we symbolized it with “:” .and this means if the first condition not true than the second one will be true.

The result of this example is “hey I’m from mars”.

c.If/else if …./else conditions statement

we use usually this kind of conditions to check a chain of conditions if they are true or not.

It is very important to understand that once a condition is found to be true, no other if statements are evaluated and once the code block for the true statement is completed, the program continues from the end of the if/else if statement.

```
_name=”mars”

if(_name == "earth"){

 _name = “EARTH"

} :(_name == "pluto") {

 _name = "PLUTO"

}:(_name == “mars") {

 _name = "MARS"

} :{

 _name = "UNKNOWN"

}
-------------------------------------------------------------------------------------------------------------------------------------------

MARS
```

Note: in this case “else if” equivalent to :(condition){} and else is equivalent to :{} this how we know the difference between them.

2- Operators (&&,| |,<,>,<=,>=,…..)

operators are used to assign values, compare values, perform arithmetic operations, and more.

Operator	Description	Example
+	Addition	X=2+3
-	Subtraction	Y=2-7
*	Multiplication	A=7*9
/	Division	D=11/2
%	Modulus (division remainder)	M=9%3
++	Increment	X++
--	Decrement	Y--
=	equal	Y=5
==	equal to	X==4 it will return true
!=	Not equal to	X!=3 it will return true
>	Greater than	x>2
<	Less than	X<3
>=	Greater than or equal	x>=2
<=	Less than or equal	X<=11
&&	and	(x>3 && y<=4)
||	or	(x>3 || y<=4)
!	not	!(x==y)

3- Loop Statements

Loops are handy, if you want to run the same code over and over again, each time with a different value.

a. @loop(start=0,end) as n{...}

We will use this loop to repeat this sentence “I’m from earth” 10 times .

See the example in the next page.

```
i=0

while(i<3){

$out(« i'm alien « )

i++ ;

}

------------------------------------------------------------------------------------------------------------------------------------

I'm alien

I'm alien

I'm alien


I'm from earth 7

I'm from earth 8

I'm from earth 9

I'm from earth 10
```

As you see above the result of our loop we repeat the sentence 10 time ,all you have to do is to call the loop and give it a start number and end number.

b. @while(condition){..}

This type of Loops can execute a block of code as long as a specified condition is true.

this loop will only stop when the condition is true, In this case we got “I’m alien “ 3 times.

c. Enumerable.loop(item){…}

Loop(item) method executes a provided function once for each array element.

```
Countries=[‘morocco’,‘usa’,’Canada’, ‘France’]

Countries.loop(item){

If(item==’morocco’){

$out(“I’m alien from ”+item)

}

}

---------------------------------------------------------------------------------------------------------------------------------------

I’m alien from morocco
```

after we declared the list of countries, we wanted to check first if there is an item has a value of 'morocco’. The result of the condition is true, therefore, it displays the sentence of "I'm alien from morocco'.

d.enumerable.loop(e,item){..}

this function takes two parameters the first one is where should the loop start ,and the second one is the name of the enumerable object.

See the example in the next page.

```
Countries=[‘morocco’,‘usa’,’Canada’, ‘France’]

Countries.loop(1,item){

If(item==’morocco’){

$out(“I’m alien from ”+item)

}

}

---------------------------------------------------------------------------------------------------------------------------------
```

In this case it will display nothing because, the loop starts from the second item of the list which is “usa”, therefore the condition will not recognize the first item which is morocco.

4- Defining Functions

In Aliens programming language functions are defined with $ keyword

There are two types of functions the first one is without parameters and the second one with the parameters.

let’s create our first functions:

```
#function without parameters

$Sayhey{

$out(‘hey ‘)

}

#function with parameters

$SayHello firstname lastname {

$out(“hello mr :”+firstname+” “+lastname)

}
```

1- How to import a Model?

In this language, we changed a little bit the flavor to make it easier for all of us to learn from and contribute to each other's code. this way we define a new ecosystem which we call models or galaxies.

models are a set of planets and functions that a program can use in order to make it easy for developers.

a.Import a model to your project

By default, you will have all the models in your project in case you don’t just check that nothing is missing.

```
from https://www.site.com/wem.mals load Saad

 from @base load @convert, @math   # for use function Ex: $rand () ➔ @math.$rand(0,10)

 load humanModule

load @base from ../../file.mals load ThisPlanetA

using @math   # After write this now for using function $rand() ➔ $rand(0,10)

$Space(){

#Statements

}
```

2- Standard library

a.Type Checking Functions

Function	Mining	example

$isnumber	return true if value is number	num= 10
		test = $isnumber(num)
		$out(test)
		>>>1

$isalpha	return true If all characters in the string are alphabet.	Text= “Aliens”
		test = $isalpha(Text)
		$out(test)
		>>>1

$isnumber	return true if value is number	num= 10
		test = $isnumber(num)
		$out(test)
		>>>1

$isequal	return true if two values are equals	Text1= “mars”
		Text2= “earth”
		test = $isequal (Text1, Text2)
		$out(test)
		>>>0

$ismatch	return true if value respect regex expression	Text1= “Aliens”
		test = $ismatch(text, “[A-z]*”)
		$out(test)
		>>>1

b.Strings Functions

Function	**Mining**	**example**

[str].lower()	return text to lowercase	Text= “ALIENS”
		newText = Text.lower()
		$out(newText)
		>>>aliens

[str].upper()	return text to uppercase	Text= “Aliens”
		newText = Text.upper()
		$out(newText)
		>>>ALIENS

[str].split(character/text)	return splitting text by character or text	Text= “The Aliens Language”
		newText = Text.split(“ ”)
		$out(newText[1])
		>>>The Aliens

[str].replace(text1,text2)	replaces a given text within the text	Text= “The PHP Language”
		newText = Text.
		replace(“PHP”,“Aliens”)
		$out(newText)
		>>> The Aliens Language

[str].len()	return length of text	Text= “Aliens”
		length = Text.len()
		$out(length)
		>>>6

[str].count(value)	return number of times the text is present	Text= “The Aliens Language”
		times = Text.count (“e”)
		$out(times)
		>>> 3

c. List basic Functions

Function	Mining	example

[list].add(object)	add object to list	listPlanet= [“Earth”, “Mars”,“Neptune”]
		newList = listPlanet.add(“Pluto”)
		newList.loop(p) {show(p+ “ - ”)}
		>>>Earth – Mars – Neptune – Pluto –

[list].insert(object,index)	insert object into list	listPlanet= [“Earth”, “Mars”, “Neptune”]
		newList = listPlanet.insert(“Pluto”,1)
		newList.loop(p) {show(p+ “ - ”)}
		>>>Earth – Pluto – Mars – Neptune –

[list].remove(object)	remove object from list	listPlanet= [“Earth”, “Mars”, “Neptune”]
		newList = listPlanet.remove(“Mars”)
		newList.loop(p) {show(p+ “ - ”)}
		>>>Earth – Neptune –

[list].sort()	return list sorted ascending	listPlanet= [“Pluto”, “Mars”, “Neptune”]
		newList = listPlanet.sort()
		newList.loop(p) {show(p+ “ - ”)}
		>>>Mars – Neptune – Pluto –

[list].reverse()	return list reverse	listPlanet= [“Earth”, “Mars”, “Neptune”]
		newList = listPlanet.reverse()
		newList.loop(p) {show(p+ “ - ”)}
		>>>Neptune – Mars – Earth –

[list].size()	return size of list	listPlanet= [“Earth”, “Mars”, “Neptune”]
		size = listPlanet.size()
		show(size)
		>>>3

[list].clear()	Delete all element of list	listPlanet= [“Earth”, “Mars”, “Neptune”]
		newList = listPlanet.clear()
		size = newList.size()
		show(size)
		>>>0

d. Global Use functions

Function	**Mining**	**example**
$system(command)	Execute the command (a string) in a subshell.	command = “date” $system(command) >>> Sat Aug 29 16:44:18 UTC 2020

3- Built in galaxies (@base)

a.Converting Functions(@convert)

Function	Mining	example

$tonumber	convert value to number	Text= “2019”
		number = $tonumber(Text)
		$out(number + 1)
		>>>2020

$totext	convert value to text(string)	Number= 5432
		Text = $totext(Number)
		$out(Text + “1”)
		>>>54321

$toxml	convert list to xml	listPlanet= [“Pluto”, “Mars”, “Neptune”]
		filexml = $toxml(listPlanet)

$tojson	convert list to json	listPlanet= [“Pluto”, “Mars”, “Neptune”]
		filejson = $toxml(listPlanet)

$todict	convert a list to dictionary	listPlanet= [“Pluto”,1, “Mars”,2, “Neptune”,3]
		filedict = $todict(listPlanet)
		>>> {'Pluto':1, 'Mars':2, 'Neptune':3}

$toascii	return number of times the text is present	Text= “The Aliens Language”
		times = Text.count (“e”)
		$out(times)
		>>> 3

b.Files Functions(@io)

Function	Mining
$f_read(path)	return a file content with format text
$f_write(path ,text)	write text in file
$f_copy(filePath ,toPath ,true|false)	copy file to directory
$f_move(filePath , toPath)	move to another path
$f_delete(path)	delete file
$f_name(path)	return a name of file
$f_directory(path)	calculate exponential
$f_extension(path)	return a extension of file
$ f_size(path)	return file's bytes number
$f_append(path , "text")	Add new text to file
$f_loop_bytes(item){... }	item takes each byte/char
$f_loop_lines(line){ ... }	line takes each line

c.Math Functions(@math)

Function	**Mining**	**example**
$sqrt	calculate square root	number = $sqrt(81) $out(number) >>>9

$fabs	calculate absolute value	Number= 5-10
		absNumber = $fabs(Number)
		$out(absNumber)
		>>>5

$pow	calculate power	number = $pow(9,2) $out(number) >>>81
$fact	calculate factorial	number = $fact(6) $out(number) >>>720
$log	calculate logarithm	number = $log(100) $out(number) >>>2
$exp	calculate exponential	number = $exp(0) $out(number) >>>1
$sin	calculate sinus	number = $sin(0) $out(number) >>>0
$cos	calculate cosine	number = $cos(0) $out(number) >>>1
$tan	calculate tangent	number = $tan(0) $out(number) >>>0
$asin	calculate arcsinus	number = $arcsin(0) $out(number) >>>0
$acos	calculate arccosine	number = $arccos(1) $out(number)

		>>>0
$atan	calculate arctangent	number = $arctan(0) $out(number) >>>0
$rand	return random number	number _1 = $rand(10) number _2 = $rand(0,1) $out(number _1+"<= 10 \n") $out("0<=")+number _2 +"<= 1") >>>4 >>>0,34526984

4- Function ALS Web

Function	Description	Example

$alspage	Main function for create page html in the specified path	$alspage(
		"D://mypage/index.html",
		$html(
		 $head(
		    $title("title of page"),
		    $meta("charset="UTF-8"),
		    $style("", ""),
		    $script("", "")
		)
		
		 $body(
		    "id ='page' style='margin:100px 160px'",
		    $section(
		        "class='sec'",
		        $div(
		        "style='background-color:#43cbbb;'",
		            $p(
		                "id='txt'",
		                "SAAD"
		            ),
		            $button(
		                "",
		                "onclick='show()'"
		            )
		        )
		    ),
		    $script(
		        "type='text/javascript'",
		        "function show(){
		            var name =
		document.getElementById('name').innerText();
		            alert('Hello '+ name);
		            }
		        ")
		)
		)
		)

$texthtml	show code html in the page.	$p("class='codeHtml'", $texthtml("<html> <head></head> <body><body> </html>"))

$addstyle	Add a style css and save to file(styleAls.css)	$addstyle("#idphoto", "border :2px solid blue ; margin-top :100px ;")
$addscript	Add a code JavaScript or jQuery on the page and save to file (scriptAls.js).	$ addscript("var name = $('#txtName').value ; alert('Hello '+name); ")
$addevent	Add event in the element(s) html and save to file (scriptAls.js).	$addevent("#idbutton" , "click", "var name = $('#txtName').value ; alert('Hello '+name); ")

5- Structural Tags

Function	**Description**	**Tag**
$a	Defines a hyperlink.	<a>…</a>
$article	Defines an article.	<article>…</article>
$aside	Defines some content loosely related to the page content.	< aside>…</aside>
$body	Defines the document's body.	<body>…</body>
$br	Produces a single line break.	…<br/>…
$details	Represents a widget from which the user can obtain additional information or controls on-demand.	<details>…</details>
$div	Specifies a division or a section in a document.	<div>…</div>
$h1 to $h6	Defines HTML headings.	<h1>…</h1>……<h6>…</h6>
$head	Defines the head portion of the document that contains information about the document.	<head>…</head>
$header	Represents the header of a document or a section.	<header>…</header>
$hgroup	Defines a group of headings.	<hgroup>…</hgroup>
$hr	Produce a horizontal line.	…<hr/>…

$html	Defines the root of an HTML document.	<html>…</html>
$footer	Represents the footer of a document or a section.	<footer>…</footer>
$nav	Defines a section of navigation links.	<nav>…</nav>
$p	Defines a paragraph.	<p>…</p>
$section	Defines a section of a document, such as header, footer etc.	<section>…</section>
$span	Defines an inline style less section in a document.	<span>…</span>
$summary	Defines a summary for the $details element.	<summary>…</summary>

6- Metadata Tags

Tag	Description	
$base	Defines the base URL for all linked objects on a page.	<base attr= '…' />
$link	Defines the relationship between the current document and an external resource.	<link attr= '…' />
$meta	Provides structured metadata about the document content.	<meta attr= '…' />
$style	Inserts style information (commonly CSS) into the head of a document.	<style>…</style>
$title	Defines a title for the document.	<title>…</title>

7- Form Tags

Tag	Description	
$button	Creates a clickable button.	<button>…</button>
$datalist	Represents a set of pre-defined options for an $input element.	<datalist>…</datalist>

$fieldset	Specifies a set of related form fields.	<fieldset>…</fieldset>
$form	Defines an HTML form for user input.	<form>…</form>
$input	Defines an input control.	<input attr= '…' />
$keygen	Represents a control for generating a public-private key pair.	<keygen attr= '…' />
$label	Defines a label for an $input control.	<label>…</label>
$legend	Defines a caption for a $fieldset element.	<legend>…</legend>
$meter	Represents a scalar measurement within a known range.	<meter>…</meter>
$optgroup	Defines a group of related options in a selection list.	<optgroup>…</optgroup>
$option	Defines an option in a selection list.	<option>…</option>
$select	Defines a selection list within a form.	<select>…</select>
$textarea	Defines a multi-line text input control (text area).	<textarea>…</textarea>

8- Formatting Tags

Tag	Description	
$abbr	Defines an abbreviated form of a longer word or phrase.	<abbr>…</abbr>
$acronym	Defines an acronym.	<acronym>…</acronym>
$address	Specifies the author's contact information.	<address>…</address>
$b	Displays text in a bold style.	<b>…</b>
$bdi	Represents text that is isolated from its surrounding for the purposes of bidirectional text formatting.	<bdi>…</bdi>
$bdo	Overrides the current text direction.	<bdo>…</bdo>
$big	displays text in a large size.	<big>…</big>
$blockquote	Defines a long quotation.	<blockquote>…</blockquote>
$cite	Indicates a citation or reference to another source.	<cite>…</cite>
$code	Specifies text as computer code.	<code>…</code>
$del	Specifies a block of deleted text.	<del>…</del>
$dfn	Specifies a definition.	<dfn>…</dfn>
$em	Specifies emphasized text.	<em>…</em>
$i	Displays text in an italic style.	<i>…</i>
$ins	Defines a block of text that has been inserted into a document.	<ins>…</ins>
$kbd	Specifies text as keyboard input.	<kbd>…</kbd>
$mark	Represents text highlighted for reference purposes.	<mark>…</mark>
$output	Represents the result of a calculation.	<output>…</output>

$pre	Defines a block of preformatted text.	<pre>…</pre>
$progress	Represents the completion progress of a task.	<progress>…</progress>
$q	Defines a short inline quotation.	<q>…</q>
$rp	Provides fall-back parenthesis for browsers that don't support ruby annotations.	<rp>…</rp>
$rt	Defines the pronunciation of character presented in a ruby annotations.	<rt>…</rt>
$ruby	Represents a ruby annotation.	<ruby>…</ruby>
$samp	Specifies text as sample output from a computer program.	<samp>…</samp>
$small	Displays text in a smaller size.	<small>…</small>
$strong	Indicate strongly emphasized text.	<strong>…</strong>
$sub	Defines subscripted text.	<title>…</title>
$sup	Defines superscripted text.	<sup>…</sup>
$tt	Displays text in a teletype style.	<tt>…</tt>
$var	Defines a variable.	<var>…</var>
$wbr	Represents a line break opportunity.	<wbr attr= '…' />

9- List Tags

Tag	**Description**	
$dd	Specifies a definition for a term in a definition list.	<dd>…</dd>
$dl	Defines a definition list.	<dl>…</dl>
$dt	Defines a term (an item) in a definition list.	<dt>…</dt>
$li	Defines a list item.	<li>…</li>

$ol	Defines an ordered list.	<ol>…</ol>
$menu	Represents a list of commands.	<menu>…</menu>
$ul	Defines an unordered list.	<ul>…</ul>

10- Table Tags

Tag	**Description**	
$caption	Defines the title of a table.	<caption>…</caption>
$col	Defines attribute values for one or more columns in a table.	< col attr= '…' />
$colgroup	Specifies attributes for multiple columns in a table.	<colgroup>…</colgroup>
$table	Defines a data table.	<table>…</table>
$tbody	Groups a set of rows defining the main body of the table data.	<tbody>…</tbody>
$td	Defines a cell in a table.	<td>…</td>
$tfoot	Groups a set of rows summarizing the columns of the table.	<tfoot>…</tfoot>
$thead	Groups a set of rows that describes the column labels of a table.	<thead>…</thead>
$th	Defines a header cell in a table.	<th>…</th>
$tr	Defines a row of cells in a table.	<tr>…</tr>

11- Scripting Tags

Tag	**Description**	
$noscript	Defines alternative content to display when the browser doesn't support scripting.	<noscript>…</noscript>
$script	Places script in the document for client-side processing.	<script>…</script>

12- Embedded Content Tags

Tag	Description	
$area	Defines a specific area within an image map.	<area attr= '…' />
$audio	Embeds a sound, or an audio stream in an HTML document.	<audio>…</audio>
$canvas	Defines a region in the document, which can be used to draw graphics on the fly via scripting (usually JavaScript).	<canvas>…</canvas>
$embed	Embeds external application, typically multimedia content like audio or video into an HTML document.	<embed attr= '…' />
$figcaption	Defines a caption or legend for a figure.	<figcaption>…</figcaption>
$figure	Represents a figure illustrated as part of the document.	<figure>…</figure>
$frame	Defines a single frame within a frameset.	<frame>…</frame>
$frameset	Defines a collection of frames or other frameset.	<frameset>…</frameset>
$iframe	Displays a URL in an inline frame.	<iframe>…</iframe>
$img	Displays an inline image.	<img attr= '…' />
$map	Defines a client-side image-map.	<map>…</map>
$noframes	Defines an alternate content that displays in browsers that do not support frames.	<noframes>…</noframes>
$object	Defines an embedded object.	<object>…</object>
$param	Defines a parameter for an object or applet element.	<param attr= '…' />
$source	Defines alternative media resources for the media	<source attr= '…' />

	elements like $audio or $video .	
$time	Represents a time and/or date.	<time>…</time>
$video	Embeds video content in an HTML document.	<video>…</video>