AtomScript Basics
By Leandro Yabut

What is AtomScript? AtomScript is a programming language designed to make Web Development more versatile and more convenient. It allows you to create interactive sites with a complex or simple structure. It transforms JavaScript from a client-side programming language to a server-side as well.

Introduction
AtomScript uses a transcompiler. A transcompiler converts source code into source code. In this case, it converts AtomScript into JavaScript. It can be used to create dynamic web applications because of the languages structure and built-in library.

Setup
Environment: You're going to need to setup a development environment to get started using AtomScript. The first thing I recommend doing is getting a web server software. I advise you to use Fenix but you can use the software of your choice. I also recommend using Notepad++ to write your AtomScript code. You can get the language XML file for Notepad++ in the Github source or in any releases. Afterwards, you should now be able to get into setting up your web page to use AtomScript.

AtomScript Setup: AtomScript is like a JavaScript framework. The compiler is written in JavaScript. To implement AtomScript into your webpage just create a script element with the source being the path/url to the 'AtomScript.js' file. Next you need to define your main AtomScript source file. You can do so by creating a new script element and inputting the following code:

```
AtomScript.src = “path/to/script/scriptname.atom”;
```

You should now be done with implementing AtomScript into your web page! Let's get into learning the basic syntax!

Syntax
AtomScript's syntax is almost a mixture of C/C++ and JavaScript. Let's start learning the basics!

Comments are completely ignored by the compiler and the browser. Comments can be used to keep code organized or to delete unwanted code but keep it for later:

```
# This is a comment.
```

```
        # This code will be ignored and removed when the code is compiled.
```

Methods are important in AtomScript, as it is what allows your program/site to run. To call a method, simply call it as you would in JavaScript:

```
        methodName();
```

You can also create your own methods as well! Use the $ symbol to define a method.

```
        $methodName(){

                # Code goes here

        }
```

A method literal can also be defined:

```
        $(){

                # Code goes here

        }
```

A method can also contain parameters/arguments:

```
        $methodName(arg1, arg2, arg3){

                # Code goes here

        }
```

It can then be called like so:

```
        methodName("arg1", "arg2", "arg3");
```

Variables are used to store data to be used later. It has a huge importance in programming. You can define a variable like so:

```
        @variableName = “SomeString”;
```

Variable and methods are both **objects**. Object hold data that can be accessed. Some objects include ‘Doc’ which holds data (methods and variables) that help with creating dynamic web pages. An example of an object being defined would be:

```
@Person = $(){      # Defines an object called Person

       this -> name = “”; # Defines a name property for Person
       this -> age = 0; # Defines an age property

}
```

After you’ve defined an object, you can create a new instance of that object. Let’s make an object called Bob:

```
@Bob = *Person(); # The asterisk is important when defining an object instance
```

Now that Bob is defined as a new person, you can edit his properties which were defined when the object was created. You can define object properties using two methods:

```
# Using namespace callers
Bob<name> = “Robert”; # Bob’s name is set to Robert
Bob<age> = 18; # Bob’s age is set to 18

# Using namespace splitters
Bob::name = “Robert”;
Bob::age = 18;
```

Now that you’ve learned some of the basics of AtomScript, let’s learn to use what you’ve learned!

Creating A Simple Program

When I think about simple programs, I think about programs like ‘Hello, World’ or a simple calculator. We are going to be learning how to make both! Let’s get started!

Hello World:

Hello World can be done in many different ways. In this, we are going to access the HTML document through AtomScript and write to it.

First, we learn about the ‘main’ method. The ‘main’ method is the first method that ever runs in a program and what it does is defined by you:

```
$main(){

        # Your code goes here

}
```

This is the method we'll be using to write 'Hello, World' onto the HTML document.

To access the document, you can use the built in namespace: 'Doc'. Afterwards, you should access the 'write' method in the Doc object like so:

```
Doc<write>("Hello, world!");
```

Now that we have our basic tools, let's start writing the actual program by creating a new variable:

```
@helloworld = "Hello, world!";
```

You can use this variable to pass into the write method:

```
Doc<write>(helloworld);
```

The written program should now look like:

```
@helloworld = "Hello, world!";

$main(){

        Doc<write>(helloworld);

}
```

This is usually the first program a programmer must learn to write. Next, we'll learn to create a simple calculator.

Simple Calculator:
Now, a simple calculator program is a program that takes two numbers the user inputs and applies an operation to them (addition, subtraction, multiplication, etc…). First we learn the arithmetic operators and comparing operators.

Operators:

- Addition   +

Subtraction	-
Multiplication	*
Division	/
Equal to	==
Equal to (+type)	===
Less than	<
Greater than	>
Less or equal	<=
Greater or equal	>=
And	&&
Or	||

Those operators are very important. Now let's get into programming.

We're going to want to create two variables (first number and second number). Set both to equal to 0:

```
@fnum = 0;
@snum = 0;
```

After that, we want to create a new method called 'add' and it will have two arguments. In the method, you will add both passed arguments and return the result:

```
$add(a, b){

        return a + b;

}
```

Once you've done that, go into your main method and prompt the user for each number and parse the integer that they entered as well. Then, create a variable called result and set it equal to the add method:

```
$main(){

        fnum = parseInt(prompt("Enter first number:"));
        snum = parseInt(prompt("Enter second number:"));

        @result = add(fnum, snum);

}
```

You can now display the result by using the alert method:

```
alert(“Your result is: ” + result);
```

Call that method in the main method. As you see above, you can also concatenate strings using the + sign.

The completed program should look like this:

```
@fnum = 0;
@snum = 0;

$main(){

        fnum = parseInt(prompt(“Enter first number:”));
        snum = parseInt(prompt(“Enter second number:”));

        @result = add(fnum, snum);

        alert(“Your result is: ” + result);

}

$add(a, b){

        return a + b;

}
```

Conclusion

Congratulations! You’ve learned the basics of AtomScript! As you can see, AtomScript can be very useful. I hope you have fun creating and writing your own programs! You can learn more about AtomScript by reading through the documentation.

AtomScript

Language created by Leandro Yabut