What is there to learn from the Functions Chapter in Eloquent JS?

By Henry van Wagenberg
Based on Research & Learning from Jan 30 - Feb 22 2018

What's interesting about Javascript? What can I share with you that will help us be more effective developers when working with Javascript?

Eloquent JS 3 is out!
2 weeks old; early Feb 2018. Includes ES 6.
Made in Berlin

What is closure?

```
 96 var outerFrance = function() {
 97     var landscape = "Green fields";
 98     var innerParis = function() {
 99         var landscape = "City streets";
100         console.log(landscape);
101     }
102    innerParis();
103    console.log(landscape);
104 }
105
106 outerFrance()
107
```

What is closure? I've always heard this word thrown around; and there's the language "closure" itself.

I never knew what it meant. UNTIL NOW.

Closure is really simple: Closure refers to the enclosed locality of a variable inside a function.

A Closure is the function that encloses the local variable.

The feature of "being able to reference a specific instance of local variables in an enclosing function"—is called closure. A function that "closes over" some local variables is called a closure.

This behavior not only frees you from having to worry about lifetimes of variables but also allows for some creative use of function values.

How does JS handle too few or too many arguments to a function?

```
117
118 //How JS Handles Too Many or Too Few Arguments
119 function power(base, exponent) {
120   if (exponent == undefined)
121     exponent = 2;
122   var result = 1;
123   for (var count = 0; count < exponent; count++)
124     result *= base;
125   return result;
126 }
127
128 console.log(power());
129 // → NaN
130 console.log(power(4));
131 // → 16
132 console.log(power(4, 3));
133 // → 64
134 console.log(power(4, 3, 17));
135 // → 64
136 console.log(power(4, 3, 17, 179));
137 // → 64
138
```

It’s extremely broad-minded: it basically handles every function like the Ruby method(*); it’s always open to more and fewer arguments.

What's the difference between values and variables?

```
82
83 henrys_list = [1,2,3];
84 andreas_list = henrys_list;
85 andreas_list[0] = 4;
86 console.log(henrys_list[0]);
87 //==> ?
88
```

The moment you create - you can use a variable to add a value to state; but after that moment, you should think of the value as separate from the variable…

A variable is like a string to a Marionette (value)

What is the call stack and its significance?

```
function chicken() {
  return egg();
}
function egg() {
  return chicken();
}
console.log(chicken() + " came first.");
// → ??
```

Because a function has to jump back to the place of the call when it returns, the computer must remember the context from which the function was called. In one case, console.log has to jump back to the chicken function. So it's like, OK: jump here and execute this function; OK, now go here to egg — wait, I need to go to chicken; ok executing… ok saving that. Now, I need to go to egg… etc.

The place where the computer stores this context is the call stack. Every time a function is called, the current context is put on top of this "stack". When the function returns, it removes the top context from the stack and uses it to continue execution.

Storing this stack requires space in the computer's memory. When the stack grows too big, the computer will fail with a message like "out of stack space" or "too much recursion". "

Thus, a Call Stack is the context in memory that a computer stores as it works its way through a program and its constituent parts. Its relevance to programming and javascript is that its an easy place for the computer to get overwhelmed and thus to "blow the stack."

What is the difference between values, expressions and statements?

22
22;

Value: A value is a chunk of bits with a particular role.
There are six types of values with their roles: functions, booleans, strings, numbers, objects and undefined values.
Expression: fragment of code that produces a value; 22 or “psychoanylsis”
Statement: 1;

What is state?

Internal state is the variables that have been assigned. It is every value that the program remembers at a given time.

What are side-effects vs values in relation to function output?

```
alert("And Moses said: Burn the golden calf!");
```

```
function square(x) {
  return x * x
};
```

Javascript functions return either “side effects”, values, or both.

Alert() is an example of a function that returns a side effect.

Function square : returns a value. This value can be used in other functions and throughout the program. Side-effects are outputs of the program to, for example, the browser, that aren’t real values useful in the rest of the program.

Pure functions are functions that only return values — and only rely on / are built from functions that return values.

What are the two ways to create a function in Javascript - and why does the difference matter?

```
var square = function(x) {
        return x * x
        };

function square(x) {
        return x * x
        };
```

This 2nd definition method is called “declaring” a function. It will allow your function to work even on EARLIER lines of the program. It bumps the definition of the function up to the top of the program.