**someone else or other people built programs that convert your JavaScript into something the computer can understand.**

**those programs, interpreters, and compilers do the process, the work of actually reading your code character by character. And determining if the syntax is valid and then implementing that syntax in a way the computer can understand.**



lexical environment of something in the code we're talking about where it's written and what surrounds it.



# **The Global Environment(Global Execution Context) and Global Variable**

**Whenever code run in JavaScript, it's run inside an execution context.**

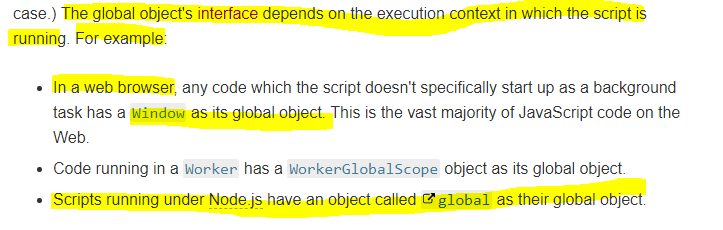
Meaning a wrapper that the JavaScript engine, the program that other people wrote that's parsing and looking at and verifying and executing your code. That wraps that up, that code that you've written, it wraps the currently executing code in an execution context.

**The base execution context, the global execution context meaning code not inside any function it means its in global execution context, and when we say global, we're talking about the thing that's accessible everywhere to everything in you're code, it's global.**

**so the global execution context creates two things for you, you don't have in your code.**

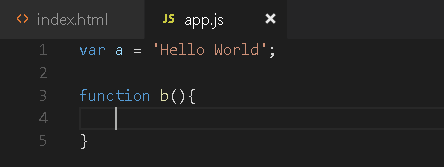
**It creates ‘a Global Object’ and a special variable for you, called 'this'.**

**The JavaScript engine is creating** these two things for you whenever your code is run, because your code is wrapped inside an execution context.



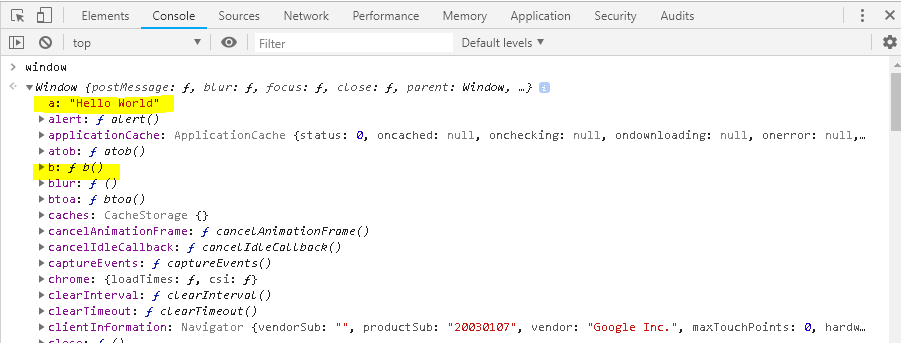
**But there is always a global object when you're running JavaScript. In the case of browsers it's the window object.**

**if you're using node js your running java script on the server it won't be the window object there's a different global object called ‘global’.**

**Lets see an example -** 

The execution context is created.

All this code is not inside a function, so it's global.



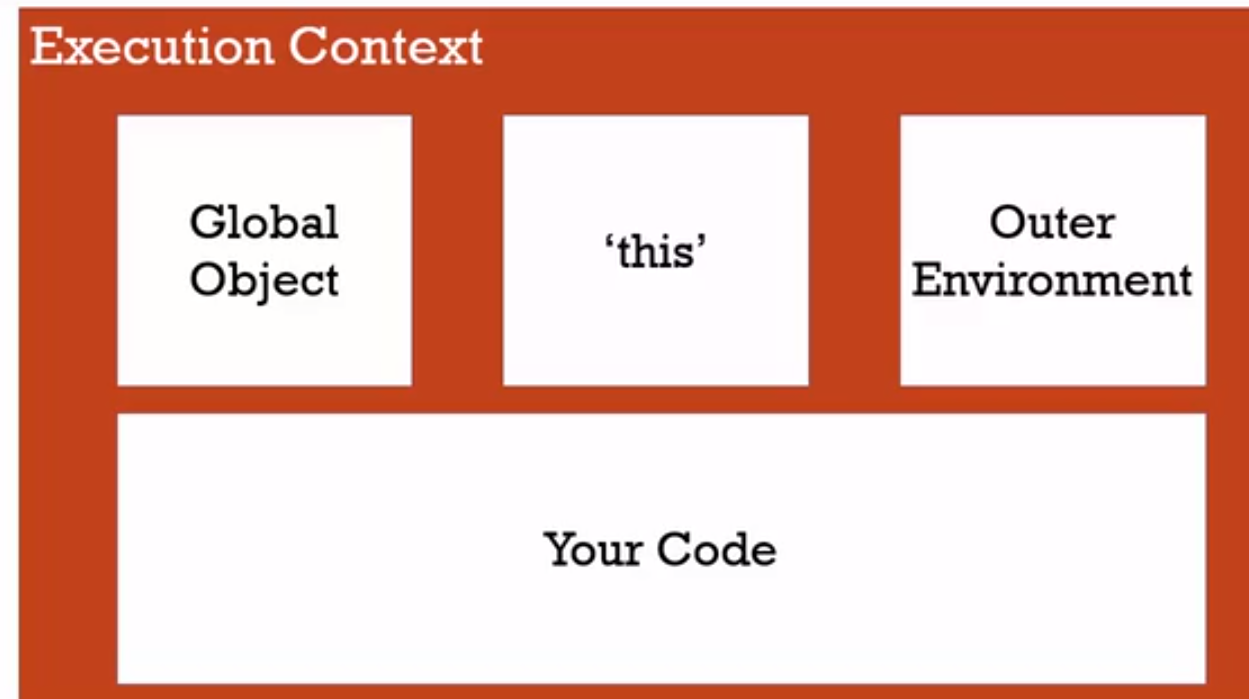
**In JavaScript, when you create variables and functions, and you're not inside a function, those variables and functions get attached to the global object.**

It’s the same thing



There's also something that is something we haven't talked about yet and we will shortly, a **link to the outer environment.**

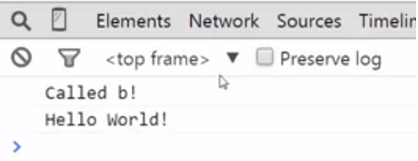
**Now when you're running code inside a function, that means the code that's outside the function is outer environment. But, when you're running at the global level, meaning when you're not inside a function, well there is nothing outside, you're as outside as you can get. There is no outer environment so that's just null, it's nothing at the global level.**



# **The Execution Context(Creation Phase) and Hoisting**

**let's look at a phenomenon that happens in JavaScript that people find surprising, and perhaps a bit confusing.**



**Output** - 

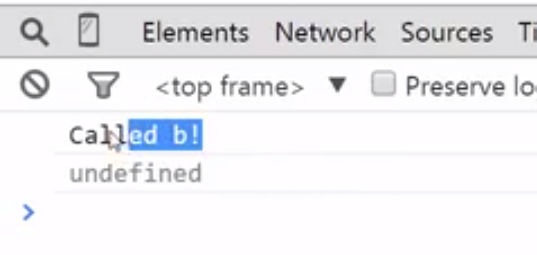
**Now let's do something a little different, that in other programing languages you would expect to not work, at all.**



Now what would you expect to have happen?

**Well, in most programming languages, you would expect an error because programming languages execute their code one line at a time, and since we haven't gotten to the b function, I can't use it yet.**

But JavaScript is not like that.



**It ran the function, and instead of throwing an error, It gave me a value but not the hello world value that I saved. But this thing called undefined.**

**So even though the function was below where it was executed, it still ran. And in fact this a variable was available to me even though it wasn't the proper value.**

Now, I actually get an error a is not defined.



