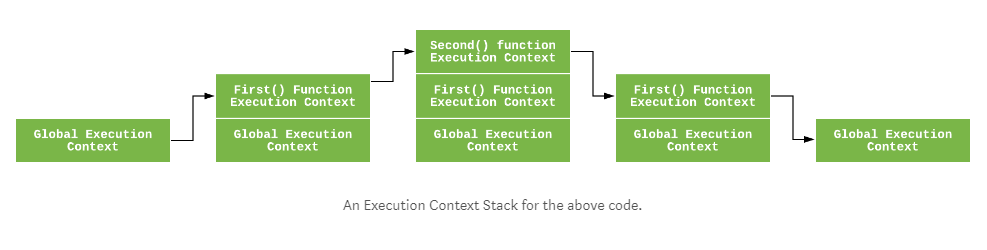
Execution Stack in JavaScript

function first() {  
 console.log('Inside first function');  
 second();  
 console.log('Again inside first function');  
}

function second() {  
 console.log('Inside second function');  
}

first();  
console.log('Inside Global Execution Context');



**The Creation Phase**

Before any JavaScript code is executed, the execution context goes through the creation phase. Three things happen during the creation phase:

1. Value of this is determined, also known as This Binding.
2. LexicalEnvironment component is created.
3. VariableEnvironment component is created.

So the execution context can be conceptually represented as follows:

ExecutionContext = {  
 ThisBinding = <this value>,  
 LexicalEnvironment = { ... },  
 VariableEnvironment = { ... },  
}

#### **This Binding:**

In the global execution context, the value of this refers to the global object. (in browsers, this refers to the Window Object).

In the function execution context, the value of this depends on how the function is called. If it is called by an object reference, then the value of thisis set to that object, otherwise the value of this is set to the global object or undefined(in strict mode). For example:

let person = {  
 name: 'peter',  
 birthYear: 1994,  
 calcAge: function() {  
 console.log(2018 - this.birthYear);  
 }  
}

person.calcAge();   
// 'this' refers to 'person', because 'calcAge' was called with //'person' object reference

**let calculateAge = person.calcAge;**

calculateAge();  
// 'this' refers to the global window object, because no object reference was given