Execution context is the concept for describing the internal working of a code. In JavaScript, the environment that enables the [JavaScript](https://www.javatpoint.com/javascript-tutorial) code to get executed is what we call **JavaScript Execution Context**. It is the execution context that decides which code section has access to the functions, variables, and objects used in the code. During the execution context, the specific code gets parsed line by line then the variables and functions are stored in the memory. An execution context is similar to a container that stores variables, and the code gets evaluated and then executed. Thus, it is the execution context that provides an environment for the specific code to get executed.

**Phases of the JavaScript Execution Context**

There are two phases of JavaScript execution context:

1. **Creation phase**: In this phase, the JavaScript engine creates the execution context and sets up the script's environment. It determines the values of variables and functions and sets up the scope chain for the execution context.
2. **Execution phase**: In this phase, the JavaScript engine executes the code in the execution context. It processes any statements or expressions in the script and evaluates any function calls.

Everything

### Creation Phase

|  |  |
| --- | --- |
| Memory | code |
| Variable:undefined  Function:{….} | Each line of code is executed line by line from top to bottom |

Execution Context

Let's take this example:

var n = 5;

function square(n) {

var ans = n \* n;

return ans;

}

var square1 = square(n);

var square2 = square(8);

console.log(square1)

console.log(square2)

At the very beginning, the JavaScript engine executes the entire source code, creates a global execution context, and then does the following things:

1. Creates a global object that is**window** in the browser and **global** in NodeJs.s
2. Sets up a memory for storing variables and functions.
3. Stores the variables with values as undefined and function references.

|  |  |
| --- | --- |
| Memory | code |
| n:undefined  square:{….}  square1:undefined  square1:undefined |  |

Execution Context

### Execution Phase

Now, in this phase, it starts going through the entire code line by line from top to bottom. As soon as it encounters **n = 5**, it assigns the value 5 to 'n' in memory. Until now, the value of 'n' was undefined by default.

Then we get to the 'square' function. As the function has been allocated in memory, it directly jumps into the line **var square1 = square(n);**. square() will be invoked and JavaScript once again will create a new function execution context.

|  |  |
| --- | --- |
| Memory | code |
| n:5  square:{….}  square1:undefined  square1:undefined | |  |  | | --- | --- | | Memory | code | | n:undefined  ans:undefined |  | |

Code Execution Phase