Noowow Community

Javascript Bootcamp / Async code

Mr Stone, 30 mai 2023

Mr Stone

Séance 8

Async code

Javascript bootcamp

Async code

- Single thread
- Synchronous execution
  - Execution context
  - Call stack
- Asynchronous execution
  - Callback
  - Promise
  - async / await

Javascript is a Synchronous single Threaded Language

Javascript bootcamp

Async code

- Thread
  - Container(Place) where code is executed sequentially
- Single thread
  - Only one statement is executed at a time.
  - It has only one call stack

Javascript bootcamp

Async code

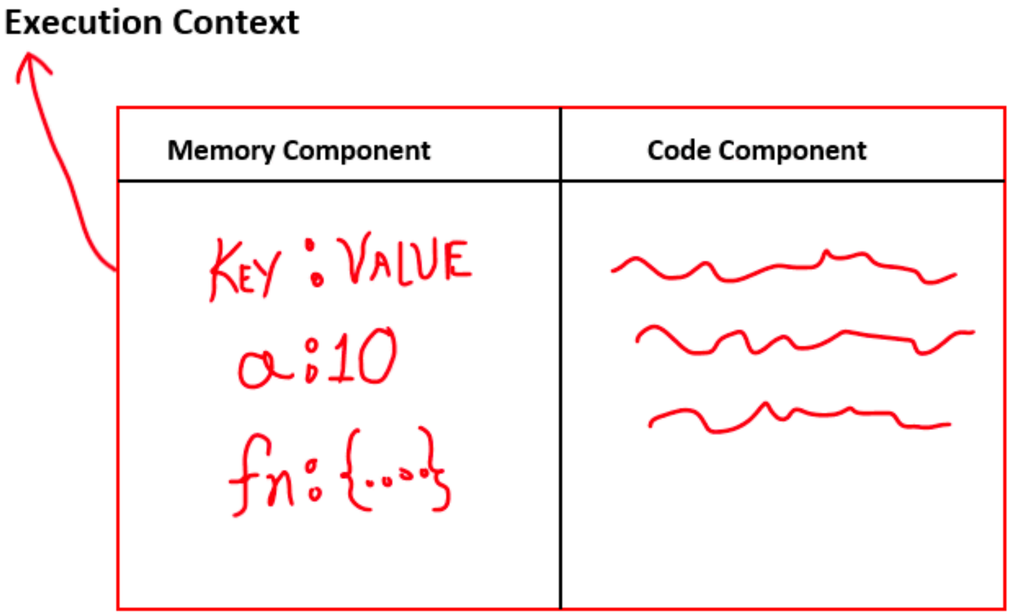
- Synchronous execution
  - Execution of code sequentially
  - Program is executed in order, line by line, one line at a time
  - Each time a function is called, the program execution waits until that function returns before continuing to the next line of code.

Everything in Javascript happens inside an execution context.

Javascript bootcamp

Async code

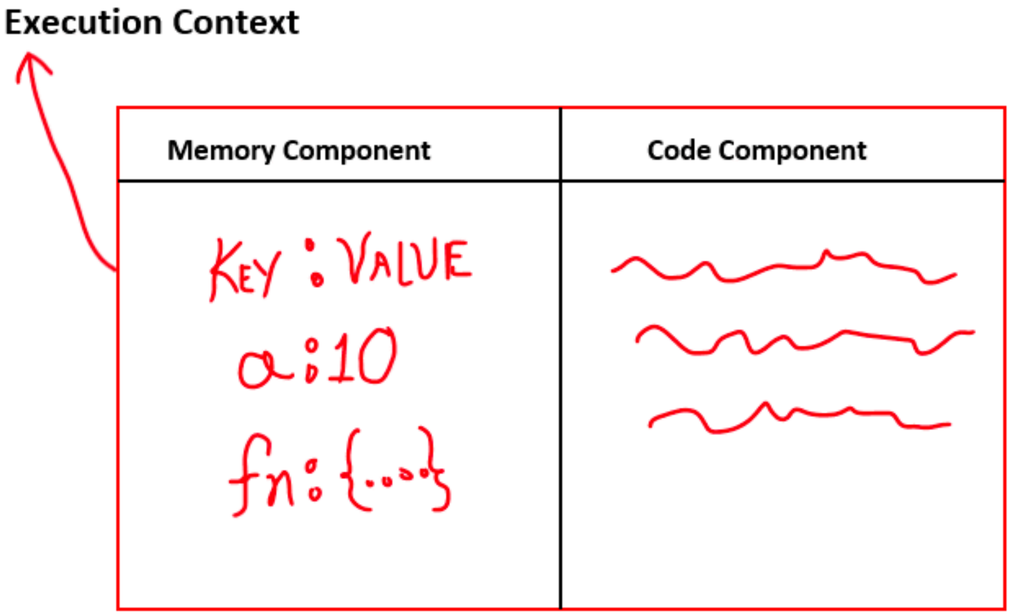
- Execution context
  - Container where whole javascript code is executed.
- Execution context components
  - Memory component / variable environment
  - Code component / Thread of execution



Javascript bootcamp

Async code

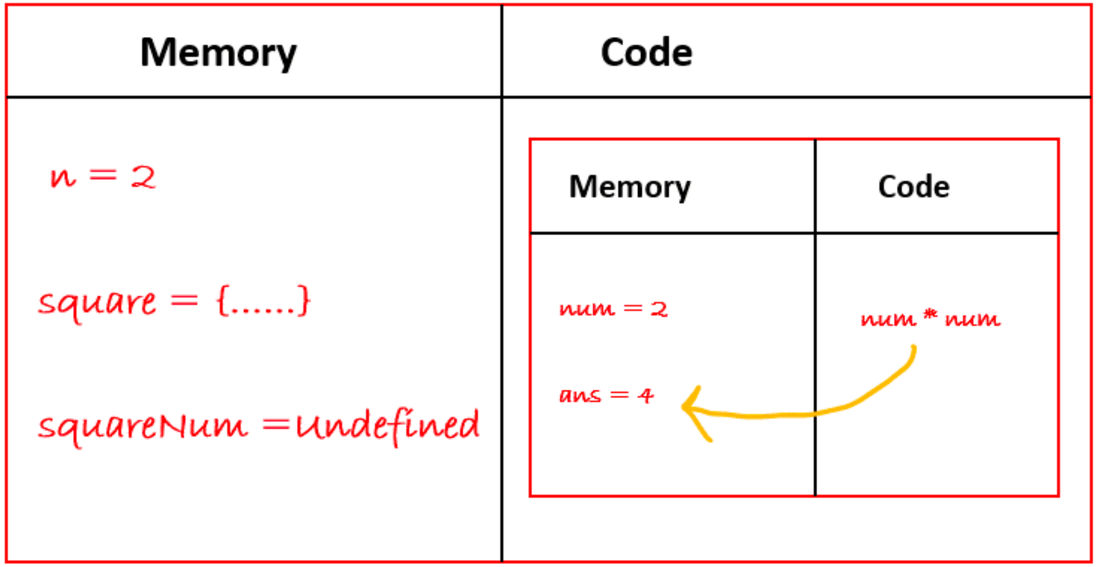
- Memory component
  - All the variables and functions are stored as a key-value pairs
- Code component / Thread of execution
  - Code are executed one line at a time.



Javascript bootcamp

Async code

- Two types of execution context
  - Global execution context
  - Function execution context
- Function call
  - New execution context in created for each function call
  - For each execution context, there is a scope chain coupled with it.



Javascript bootcamp

Async code

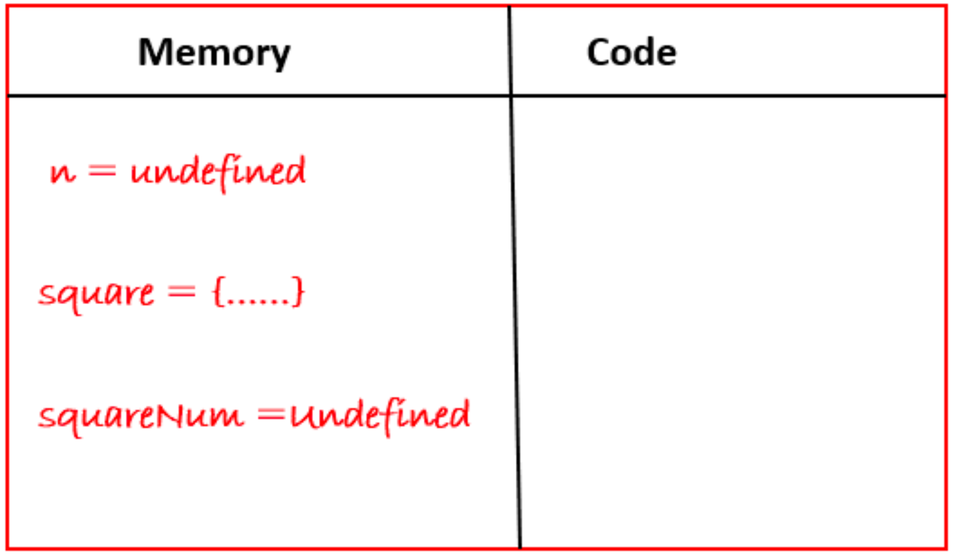
- How Javascript Code is Executed
  - When the javascript code is run, a global execution context is created.
  - Execution is created in two phases
    - Memory Creation phase
    - Code execution phase

```
1  var n=2;
2  function square(num)
3  {
4      var ans=num*num;
5      return ans;
6  }
7  var squareNumber=square(n);
8
```

Javascript bootcamp

Async code

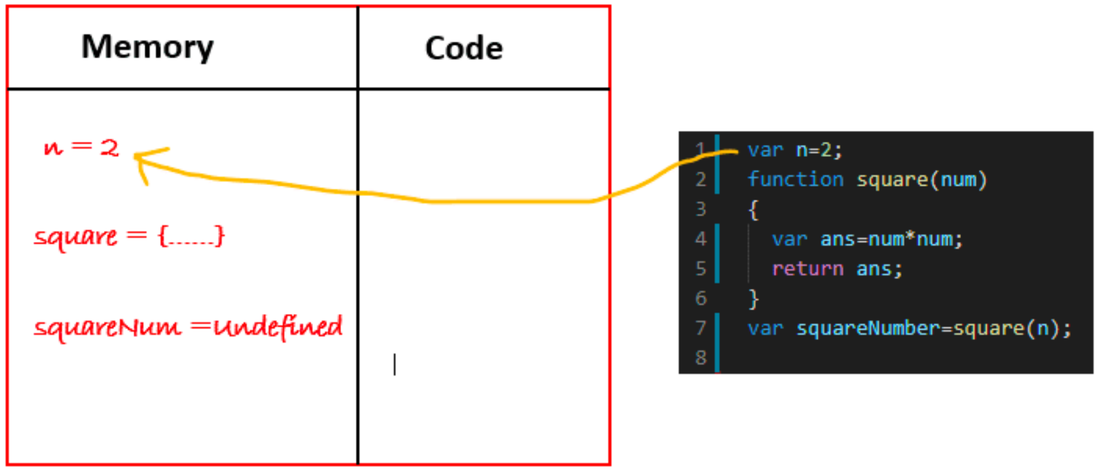
- Memory Creation phase
  - Allocate memory to all the variable and functions
  - Set undefined as variable value
  - For function it stores whole code of the function in memory space.



Javascript bootcamp

Async code

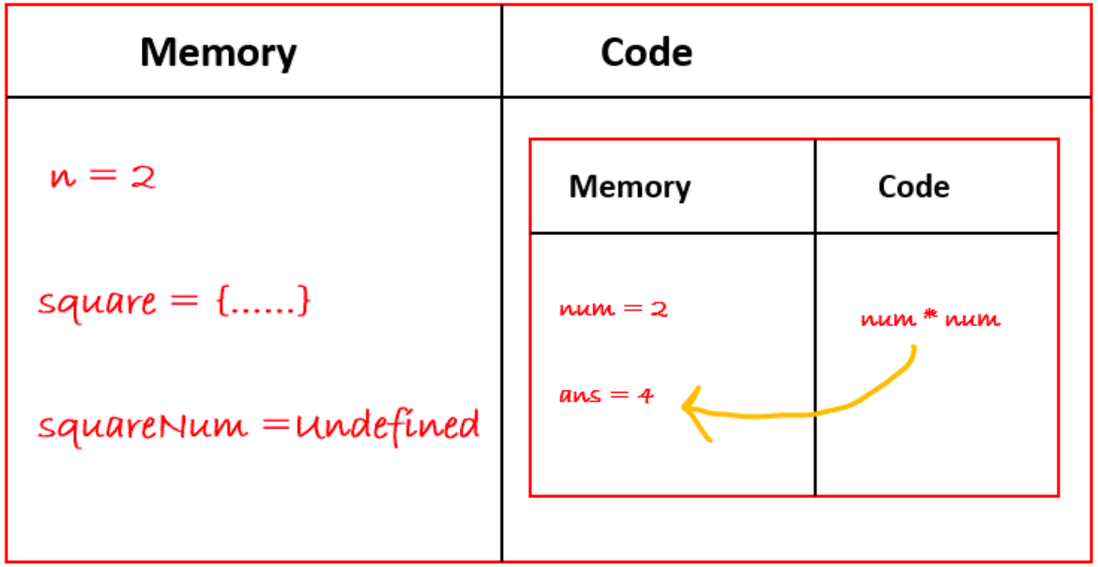
- Code execution phase
  - Make variables affectation
  - Make computation



Javascript bootcamp

Async code

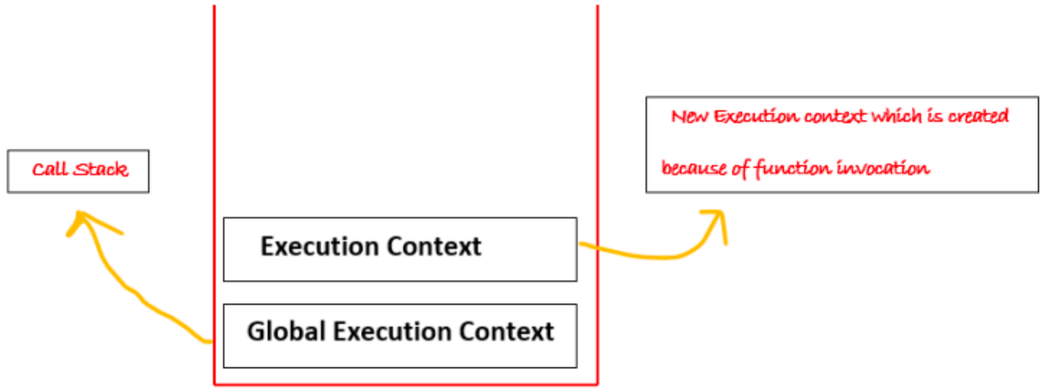
- Function invocation
  - Each function invocation in a code execution phase create a new context related to the function being invoked
  - Because functions are like mini programs
  - And goes through all the phases



Javascript bootcamp

Async code

- Call stack / Pile d’execution
  - Tracks the current function in execution
  - Maintains the order of execution of Execution Context
  - LIFO: Last in first out
  - Add function to the stack when invoked
  - Remove function from the stack when completed



Javascript bootcamp

Async code

- Call stack / Pile d’execution
  - Is empty before running the code
  - Is empty after running the code
  - Other names (Execution context stack, Program Stack, Control Stack, Runtime Stack, Machine Stack)

Javascript bootcamp

Async code

- In summary
  - Codes in javascript are executed in order, one at a time
  - When you run a code, a global execution context is created
  - The global execution context is added to the call stack to keep track of it and to execute it in a sequential order.
  - A new execution context is created for each new function invocation and this context is added to the stack for the same purpose
  - When a function execution is complete the context is removed from the stack
  - Each context as a scope chain coupled with it
  - At the end, after all codes are executed, the stack becomes empty

Asynchronous code

Javascript bootcamp

Async code

- Asynchronous
  - Code execution not occurring at the same time or sequentially
  - As single thread language how does it handle asynchronous code?
  - Two group of async operation
    - **Browser API/Web API** events or functions. These include methods like setTimeout or event handlers like click, mouse over, scroll, and many more.
    - **Promises**. A unique JavaScript object that allows us to perform asynchronous operations.

Javascript bootcamp

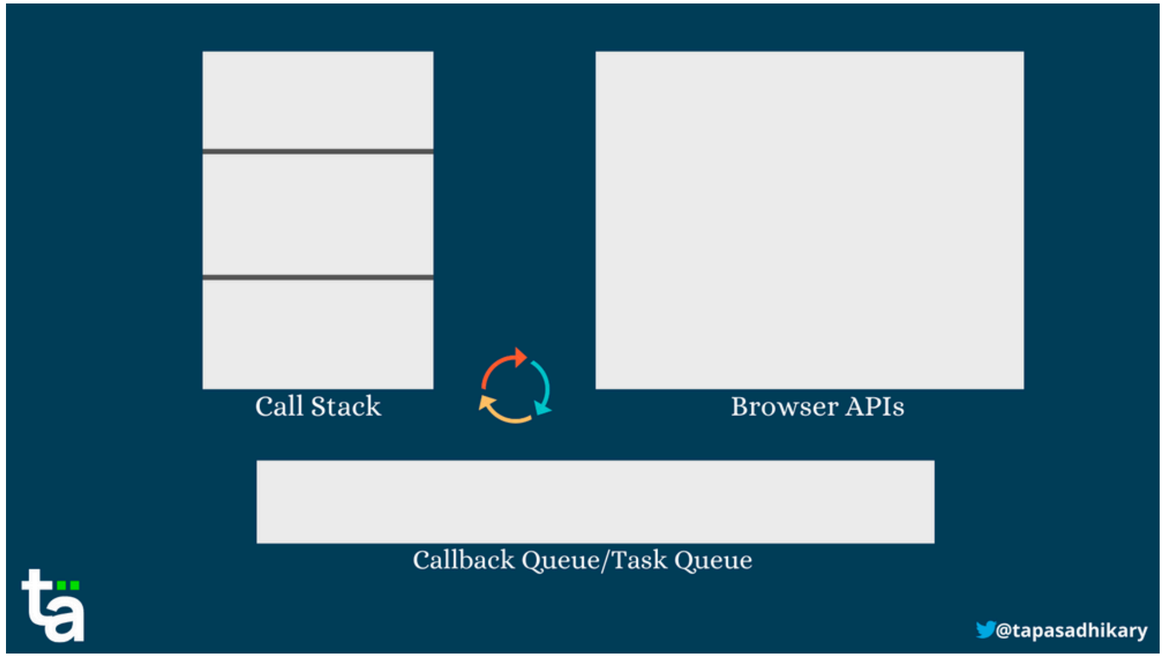
Async code

- Callback
  - A function passed as argument to another that will be execute in the futur
  - Browser APIs like setTimeout, setInterval and event handlers rely on callback functions. A callback function executes when an asynchronous operation completes.

Javascript bootcamp

Async code

- Callback Queue / Task Queue
  - FIFO
  - Stock all the callback functions invoked by the Browser API
- Event Loop
  - Created by the engine to look into the queue periodically to find what it needs to pull from there. It pulls a callback function from the queue to the call stack when the stack is empty.



Javascript bootcamp

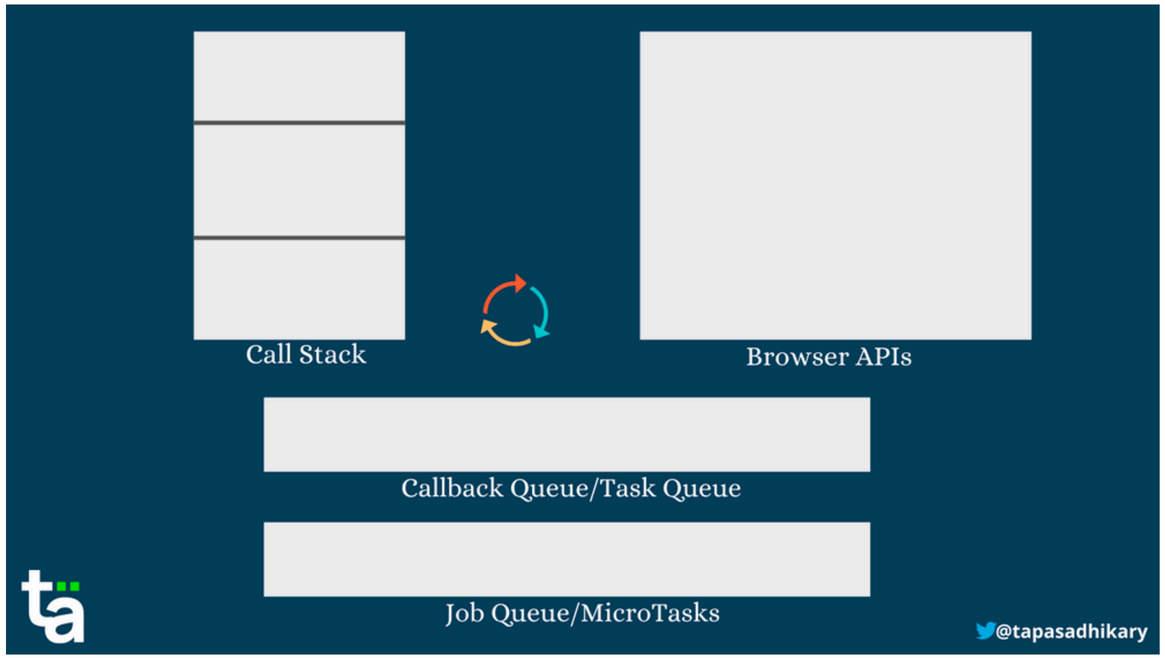
Async code

- **Promise**
  - Special objects that help you perform asynchronous operations.
  - Executor function
    - Resolve
    - Reject
  - Thenable
    - Then
    - Catch
    - Finally

Javascript bootcamp

Async code

- Promise / Job Queue
  - FIFO
  - Stock all the executor functions each time a promise occurs
  - The event loop gives priority to the job queue items over the callback queue items when the stack is free
- Item in the callback queue is called a macro task, whereas the item in the job queue is called a micro task.



Javascript bootcamp

Async code

- **Async - await**
  - Async
    - Declares an asynchronous function
    - Automatically transforms a regular function into a Promise.
    - Async functions enable the use of await.

Javascript bootcamp

Async code

- **Async - await**
  - Await
    - Pauses the execution of async functions.
    - When placed in front of a Promise call, await forces the rest of the code to wait until that Promise finishes and returns a result.
    - Await works only with Promises, it does not work with callbacks.
    - Await can only be used inside async functions.

Javascript bootcamp

Async code

- **Async - await**
  - Handing error