← Notes

Understanding code execution in Javascript

0 JavaScript Code Review

If you are doing any form of programming in Javascript then understanding its eventloop is very important.

Any javascript engine has three kinds of memory models:

1. **Stack**, which has the current function pointer an gets executed sequentially.
2. **Heap**, this stores all the objects, functions basically anything that is initiated is stored here.
3. **Queue**, all things to be executed is stored here and stack picks up tasks to do from the queue.

So, to understand this further, when a function is getting executed its loaded in the stack and if it encounters any setTimeouts then it would be added in the queue and when the current stack gets empty then the new function to be executed from the queue.

This code run will explain the process:

```
console.log("Add this code to queue");

setTimeout(function() {
//This goes and sits in the queue.
                              console.log("Running next event from the
queue.");
                   },0);

function a(x) {
     console.log("Function a added to the stack!");
     b(x);
     console.log("Function a removed from the stack!");
}

function b(x) {
      console.log("Function b is added to the stack.");
      console.log("Value passed is "+x);
```

```
        console.log("Function b is removed from the stack.");
}

console.log("starting work for this stack");
a(22);
console.log("Stopping work for this stack. stack would be empty after this.");
```

The output of this code is self explanatory:

Add this code to queue starting work for this stack Function a added to the stack! Function b is added to the stack. Value passed is 22 Function b is removed from the stack. Function a removed from the stack! Stopping work for this stack. stack would be empty after this. Running next event from the queue.

So what happened here? Stack is initialized with current code. Only thing to notice is that at setTimeout call anonymous function is added to the queue. //Remember closures. Then the function a is added to the heap. Then function b is added to the heap. a is called that means from heap function a is loaded on the stack. b is called that means from heap function b is loaded on the stack. b is removed from the stack. a is removed from the stack. current code execution gets over, stack becomes empty. next execution is loaded on the stack from the queue.

This should help one understand how JS memory model works

From my blog: http://www.censore.blogspot.in/2014/11/understanding-code-execution-in.html

AUTHOR

Biplav
Senior Software Engineer a...
Bangalore
1 note

TRENDING NOTES

Python Diaries Chapter 3 Map | Filter | For-else | List Comprehension
written by Divyanshu Bansal

Bokeh | Interactive Visualization Library | Use Graph with Django Template
written by Prateek Kumar

Bokeh | Interactive Visualization Library | Graph Plotting
written by Prateek Kumar

Python Diaries chapter 2
written by Divyanshu Bansal

Python Diaries chapter 1
written by Divyanshu Bansal

more ...

6
LIVE EVENTS

About Us
Innovation Management
Technical Recruitment
University Program
Developers Wiki
Blog
Press
Careers
Reach Us