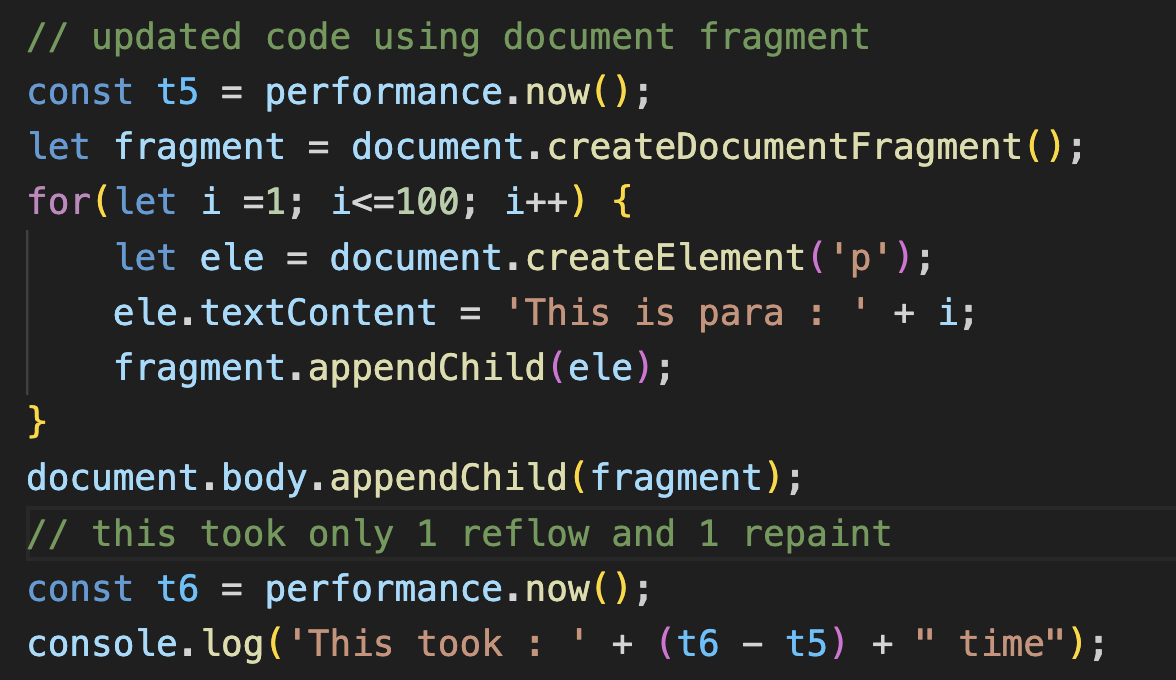
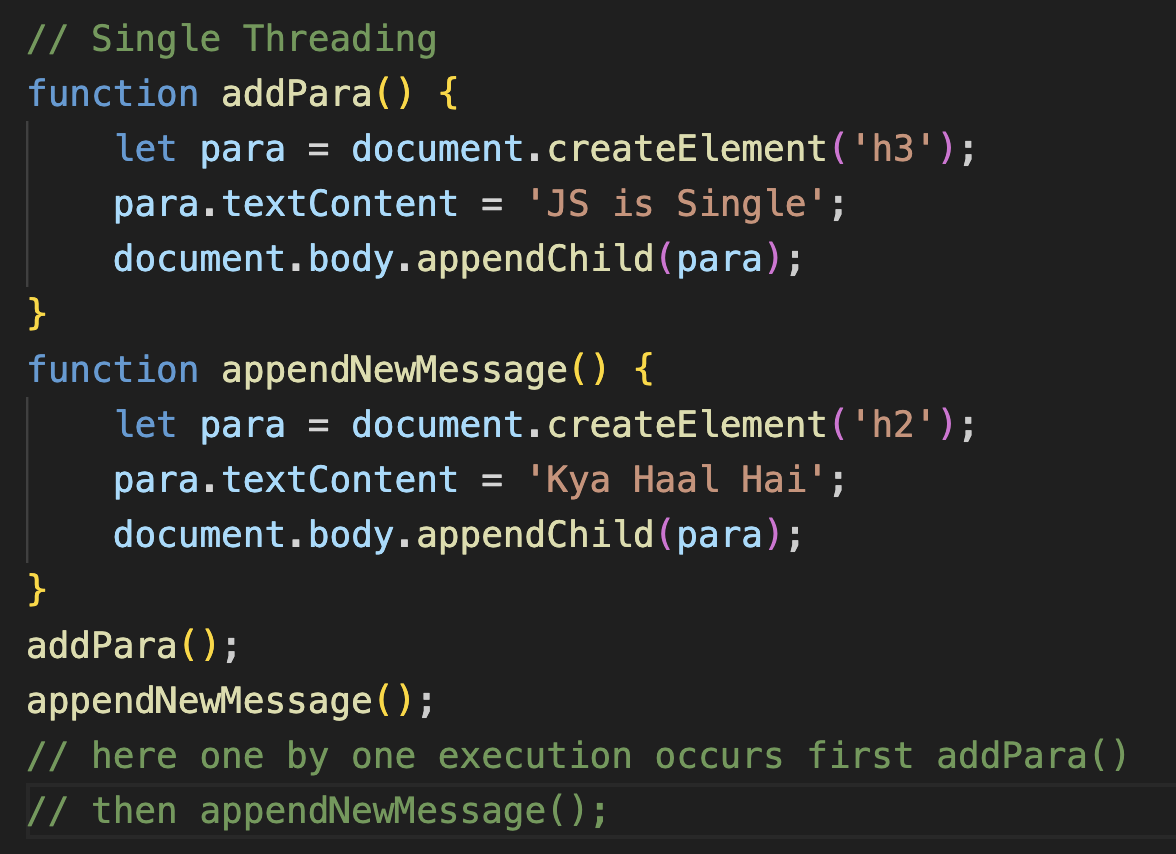
**JavaScript DOM + Modern JS**

**Performance**

* Measure speed of code:
  + Standard way to measure how long your code took to run:
  + Using performance.now(): very accurate function.
  + 
  + On which these performances are dependent.
  + Reflow: When new element is added or deleted some mathematics calculation happens these calculations are known as reflow. Like dimension of paragraph, where to put that paragraph etc.
  + Repaint: to show that element (screen layout) pixel by pixel is known as repaint.
* How to write efficient and performing code
  + Document Fragment: lightweight document object
  + Whatever we add / delete element on this document fragment no reflow and repaint occurs only 1 reflow and repaint occur when we add this document fragment to main document.
  + 
  + When we want to update DOM too many times there we use createDocumentFragment.
* The Call Stack:
  + Single-Threading: At a time only one command is running like JAVASCRIPT.
    - Processing of one command at a time.
    - Line by line execution happens
    - 
    - Run to completion nature of code.
    - JS does not execute multiple lines/function at the same time.
    - JS find how to execute function and all they use call stack.
    - Call stack talks about flow of execution. When a function is called its entry is made in call stack and when it gets executed completely then it gets removed from the call stack.

**Event loop**

* Synchronous: Occurring at the same time. Like VLC media player video audio plays at the same time.
* Async: Event Listener are kind of asynchronous.
* Three things to learn so that to get understanding of Event Loop:

1. Call stack
2. Browser
3. Event Queue

// Event Loop

console.log('Hi');

document.body.addEventListener('click', function() {

console.log('123');

});

console.log('Hello');

// flow will be first 'Hi' prints (goes into call stack)

//then an entry will be made in browser

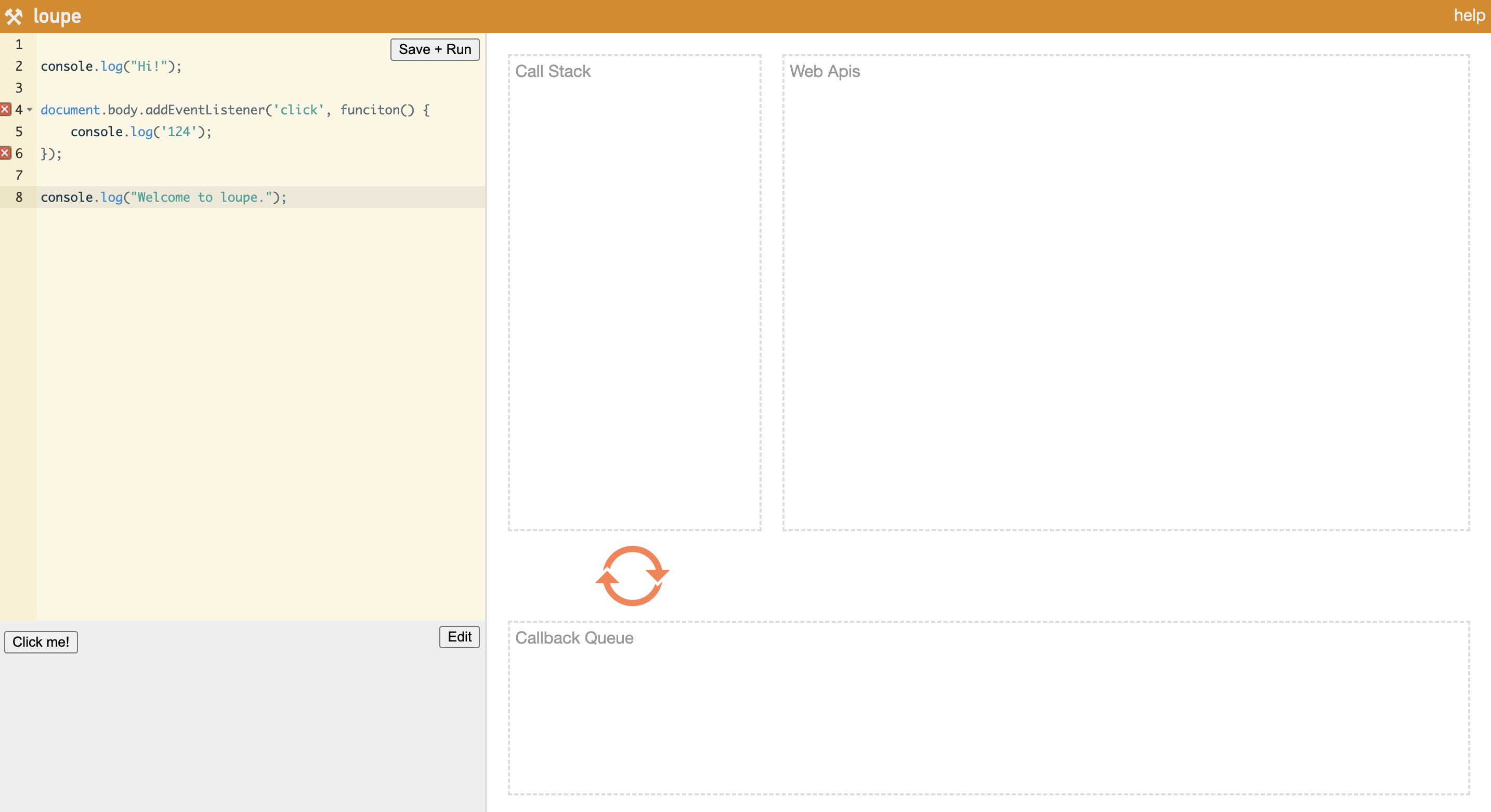
// regarding that a eventListener is there for click event

// after that 'Hello' will be made

// the entry which was made in browser for eventListener it will go

// went to Event Queue when someone click and this queue will start // its execution

// when call stack become empty

* It does not mean when you click then 123 will be printed at the same time it may happen that some kind of synchronous function task was running so it may take time to print 123.
* 
* You can check out the above website to know more.
* Asynchronous:
  + All asynchronous code is being executed in form of Event Loop.
  + Handling of Asynchrony code is done by Browser
* SetTimeout():
  + Is a method which takes two arguments first one is any kind of function and second one in milli seconds which states that we need to wait for this milli seconds to execute the function but it is not guaranteed that it will run after that milli seconds because this async code will be run after call stack become empty.
  + setTimeout(function() {

console.log(‘Hi’);}, 4000);

It will run after 4000ms.

* Every async function is being called via Event Loop.