THROTTLING

INTERVIEW QUESTIONS-48

JS

Follow on in

@Duvvuru Kishore

Questions on the topic of throttling

1) How can you restrict the large number of api calls?

2) How throttling is different from debouncing?

3)Can you provide an example of a real-world scenario where throttling is essential for a web application's performance?

A simple example to help you understand what’s throttling and how it’s helpful :

Imagine you have a water tap in your home 🤣

Consider three scenarios

- Normal flow : No Throttling)
- Partially Open Tap: Throttling Begins
- Closed tap :Complete Throttling

Normal Flow (No Throttling)

- When you fully open the tap, water flows quickly and fills your glass in seconds.
- This is similar to a service or system responding promptly to requests without any limitations.

Partially Open Tap (Throttling Begins):

- First, think of slowing down the flow of water by slightly turning the tap.

- In a digital context, throttling similarly involves intentionally limiting incoming requests to prevent system overload and ensure a controlled response

Closed tap (Complete Throttling):

- If you further close the tap, water trickles out very slowly or stops completely.
- Similarly, in the digital world, complete throttling means the service stops responding to requests altogether or responds extremely slowly. It's like telling the system, "I can't handle any more requests right now; please wait."

With the three points you got the idea on how throttling behaves.

let’s understand each of three steps with real time example to take deep into the topic

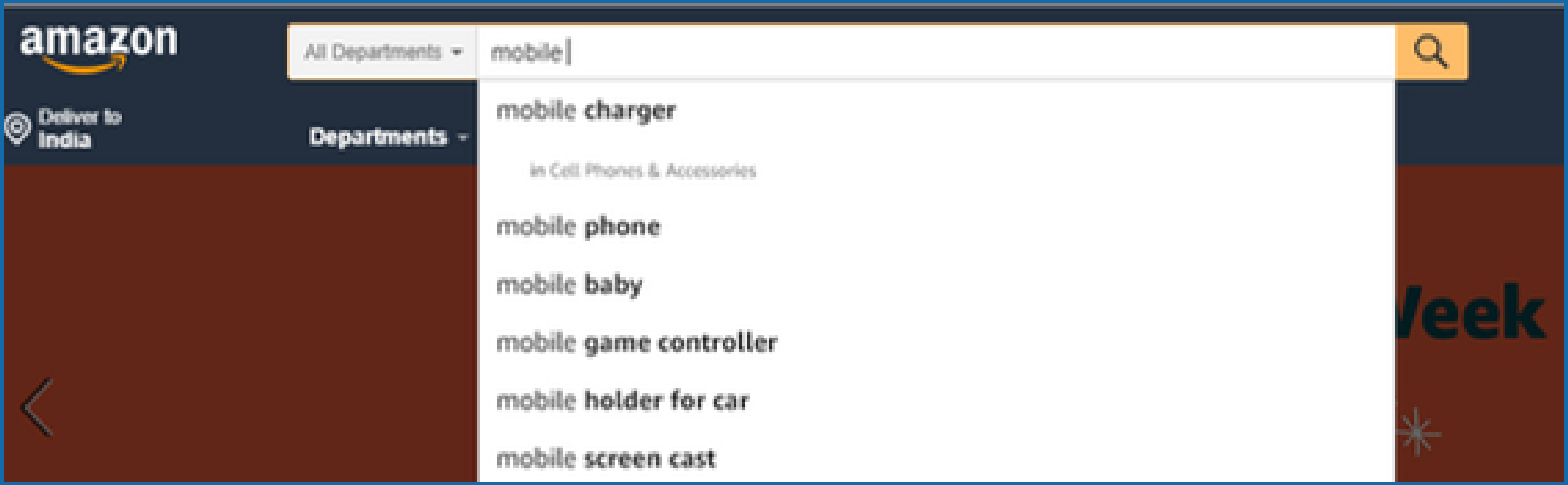
1)No throttling : real time scenario

In a chat application, when users send messages and receive responses, there's typically no throttling involved. Messages are delivered instantly, and there's no delay in communication.

Before understanding pratial throttling and closed throttling. we will understand throttling and how it’s going to help.

suppose a amazon ,big billion sale is happening and you want to buy a phone

- Now you are heading to search bar to search for different phone and their offers



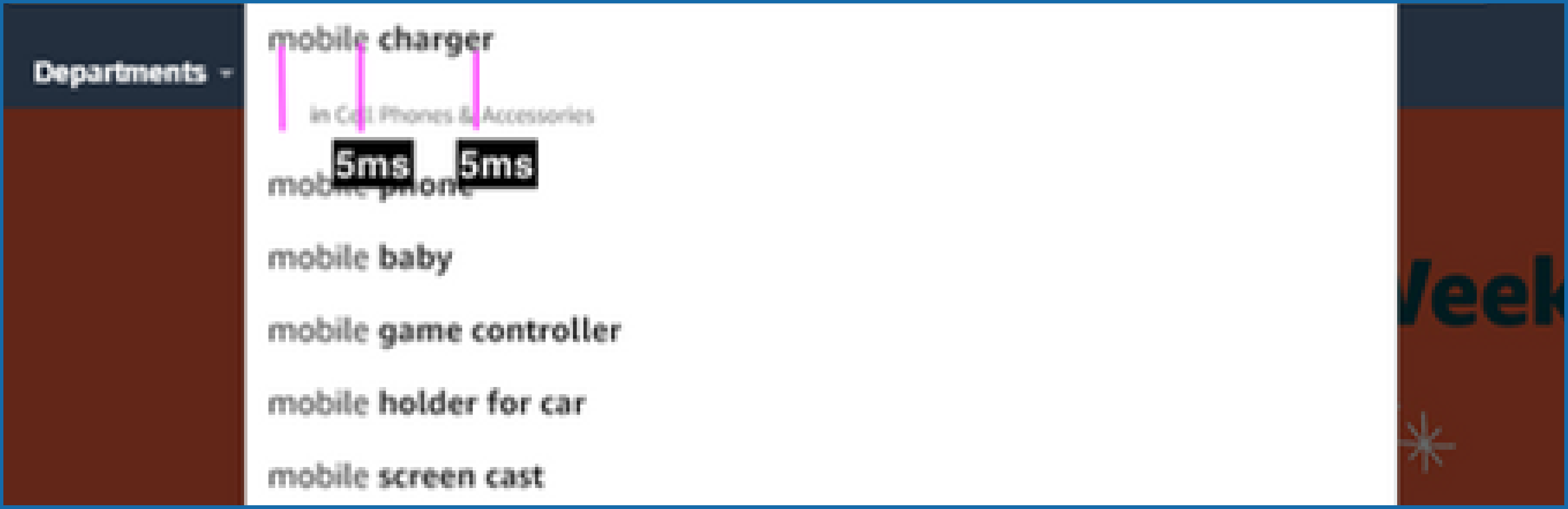
- you are typing name of the phone you want and it will give the results.

- Generally without throttling or debouncing the api is called on each key press. which means api is called 6 times while typing mobile.

- To restrict to many api calls we have throttling and debouncing

Definition

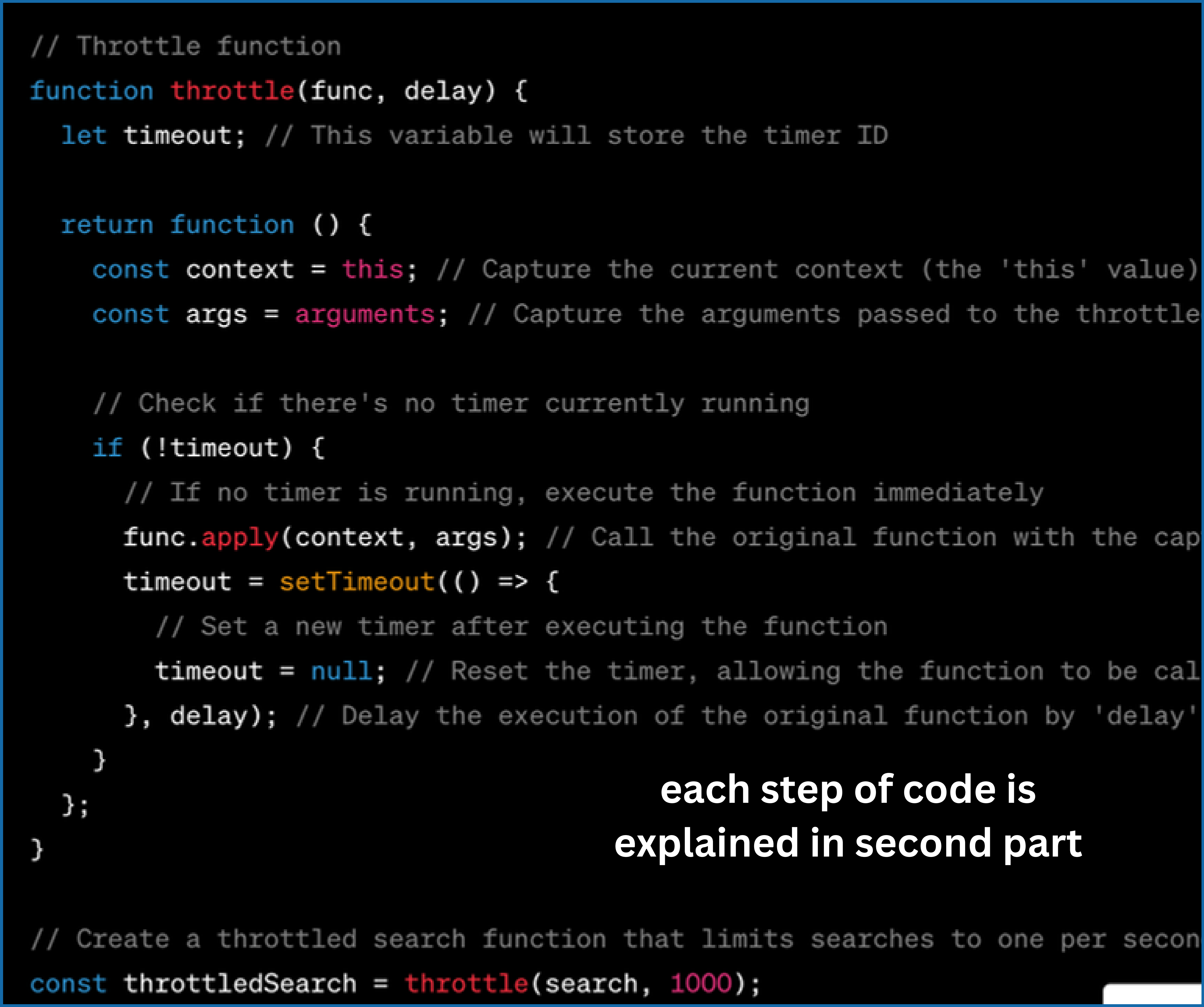
Throttling restricts how often a specific action or operation can occur within a set time frame in software development.



mobile charger
in Cell Phones & Accessories
mobile phone
mobile baby
mobile game controller
mobile holder for car
mobile screen cast

- **With throttling, api is called after a time period. we have set it as 5ms .**
- **first call is made on keyprers M and second api call is called only after 5ms if the key is pressed.**

- Instead of calling 13 api calls on a key press of mobile charger. it’s only called 3 times while using throttling
- And it’s a partial throttling

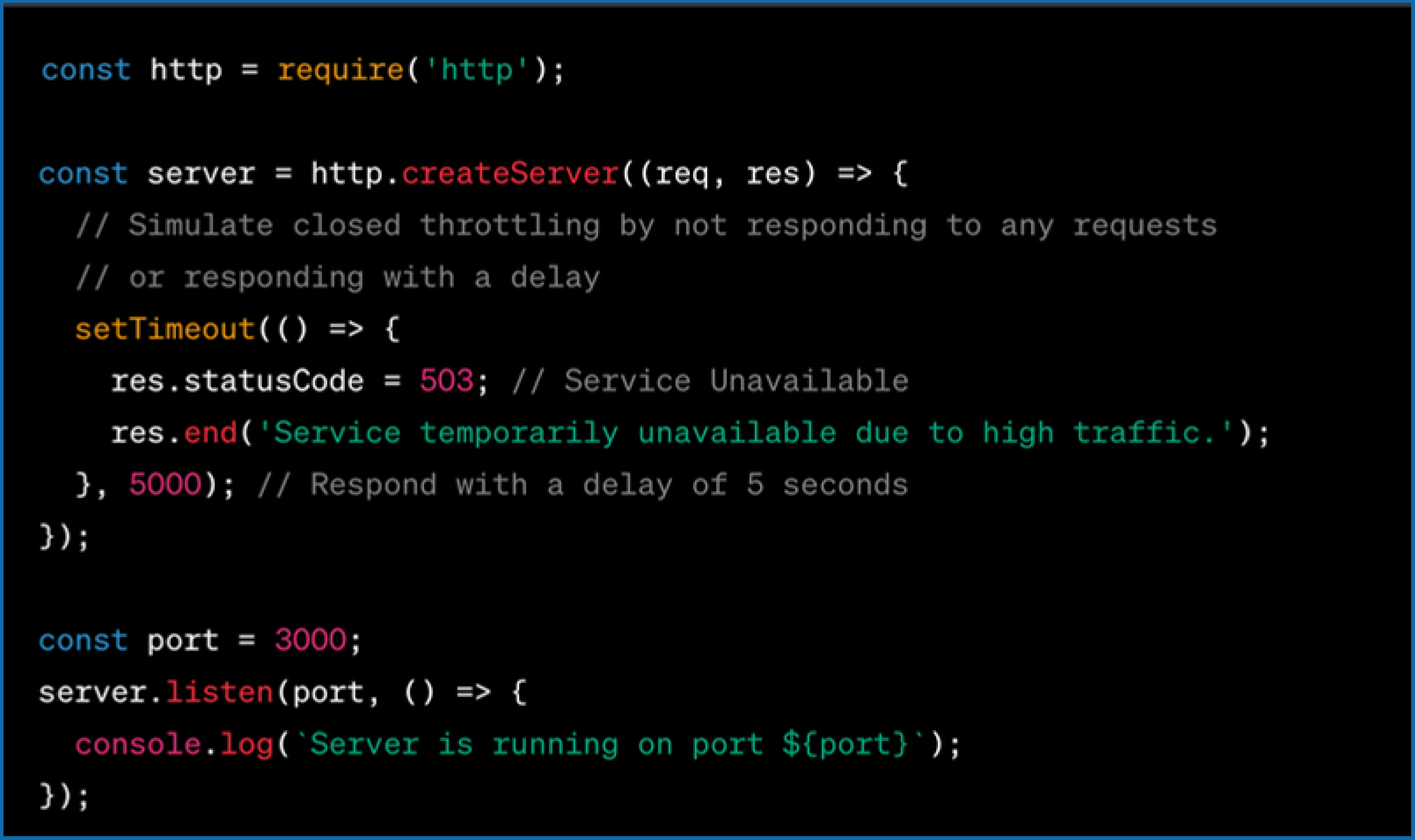


```
// Throttle function
function throttle(func, delay) {
  let timeout; // This variable will store the timer ID

  return function () {
    const context = this; // Capture the current context (the 'this' value)
    const args = arguments; // Capture the arguments passed to the throttle

    // Check if there's no timer currently running
    if (!timeout) {
      // If no timer is running, execute the function immediately
      func.apply(context, args); // Call the original function with the cap
      timeout = setTimeout(() => {
        // Set a new timer after executing the function
        timeout = null; // Reset the timer, allowing the function to be cal
      }, delay); // Delay the execution of the original function by 'delay'
    }
  };
}

// Create a throttled search function that limits searches to one per secon
const throttledSearch = throttle(search, 1000);
```

each step of code is explained in second part

Closed throttling

A real-world example of closed throttling could be a web server that temporarily shuts down access to its website due to high traffic or a service outage



```
const http = require('http');

const server = http.createServer((req, res) => {
  // Simulate closed throttling by not responding to any requests
  // or responding with a delay
  setTimeout(() => {
    res.statusCode = 503; // Service Unavailable
    res.end('Service temporarily unavailable due to high traffic.');
  }, 5000); // Respond with a delay of 5 seconds
});

const port = 3000;
server.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```

Throttling code will be explained in second part with different examples

Follow on

@Duvvuru Kishore