1. Introduction

How to measure the speed of the code. JS Event loop.

1. Add Page Content Efficiently

The standard way to measure how long it takes code to run is by using performance.now().

The browser's performance.now() method is slightly different in that it starts measuring from the time the page loaded.

These are the steps to use performance.now() to measure the speed of your code:

1. use performance.now() to get the an initial start time for the code
2. run the code you want to test
3. execute performance.now() to get another time measurement
4. subtract the initial time from the final time

const startingTime = performance.now();

for (let i = 1; i <= 100; i++) {

for (let j = 1; j <= 100; j++) {

console.log('i and j are ', i, j);

}

}

const endingTime = performance.now();

console.log('This code took ' + (endingTime - startingTime) + ' milliseconds.');

The browser is constantly working to make the screen match the DOM. When we add a new element, the browser has to run through a reflow calculation (to determine the new screen layout) and then repaint the screen. This takes time.

This is exactly why we have the DocumentFragment!

const fragment = document.createDocumentFragment(); // ← uses a DocumentFragment instead of a <div>

for (let i = 0; i < 200; i++) {

const newElement = document.createElement('p');

newElement.innerText = 'This is paragraph number ' + i;

fragment.appendChild(newElement);

}

document.body.appendChild(fragment); // reflow and repaint here -- once!

1. Reflow &amp; Repaint

Reflow is the process of the browser laying out the page. (slow)

Repaint happens after reflow as the browser draws the new layout to the screen. (faster but still want to reduce occurrences)

Reflow is the process of calculating the dimensions and position of page elements. This is a computationally intensive (slow) tasks. Repaint is the process of drawing the pixels to the screen. This is faster than reflow, but is still not a fast process. You want to make sure that your code causes the fewest number of reflows as possible.

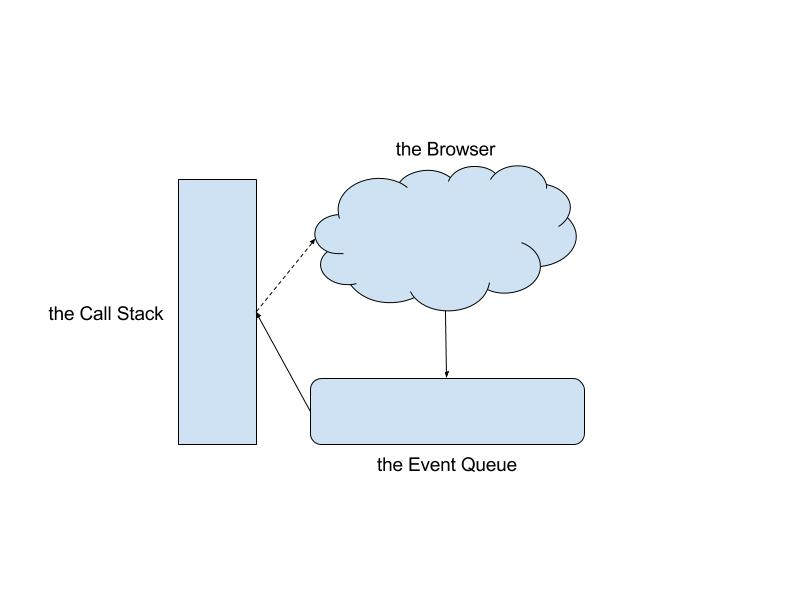
1. The Call Stack

JavaScript is single-threaded meaning it can process one command at a time.

Call Stack filo.

1. The Event Loop

* the Call Stack
* Web APIs/the browser
* an Event Queue



1. setTimeout

The setTimeout() function takes:

* a function to run at some later time
* the number of milliseconds the code should wait before running the function

You might think that since it has a delay of 0 milliseconds, that the sayHi function would run immediately. However, it still goes through the JavaScript Event Loop. So the function is handed off to the browser where the browser starts a timer for 0 milliseconds. Since the timer ends immediately, the sayHi function will move to the Queue, and then to the Call Stack once the Call Stack has finished executing any currently-running tasks.

1. Lesson summary

Speed, reflow and repaint, JS Event Loop.

1. Course summary