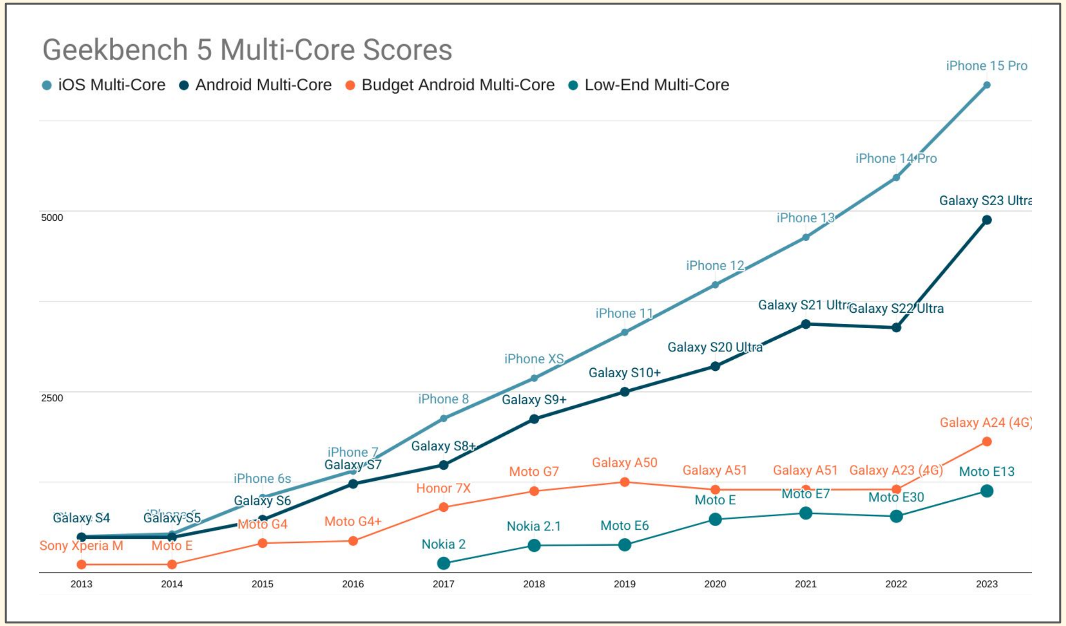
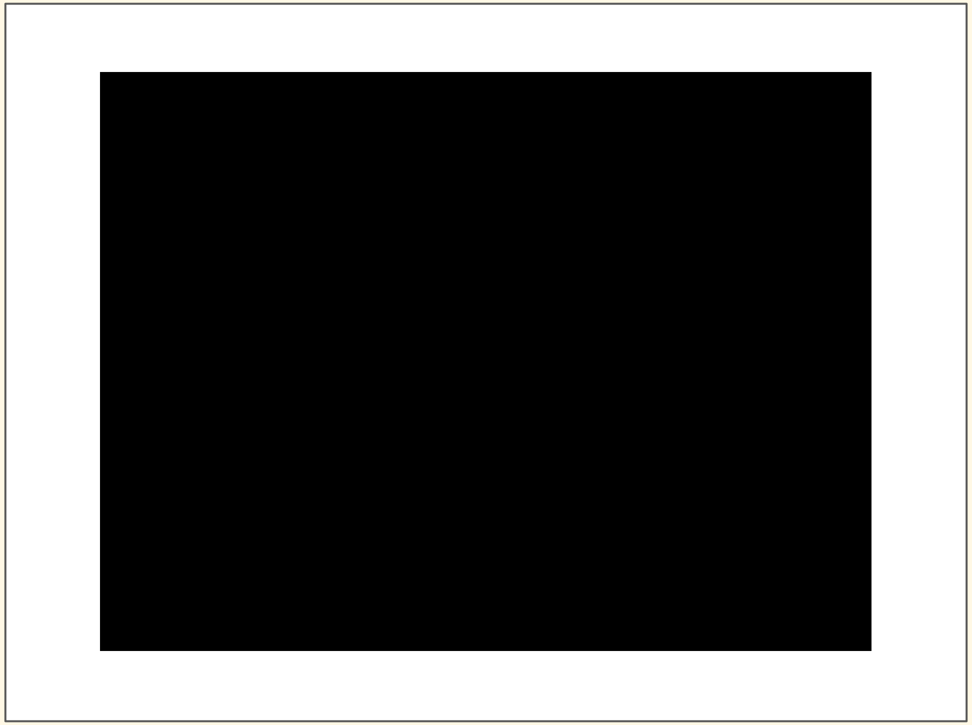
Scheduling work on the main thread to improve INP

Dec 06 2023



Source: https://toot.cafe/@slightlyoff/111527384063250418

Interaction to Next Paint (INP) will become a Core Web Vital in March 2024



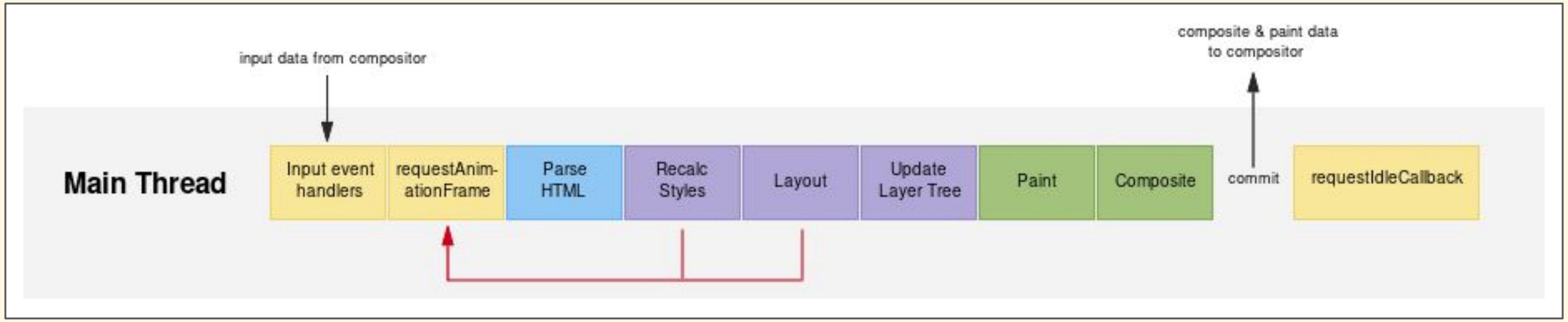
Source: https://imkev.dev/inp

Hi, I’m Kevin.

Learn Performance!

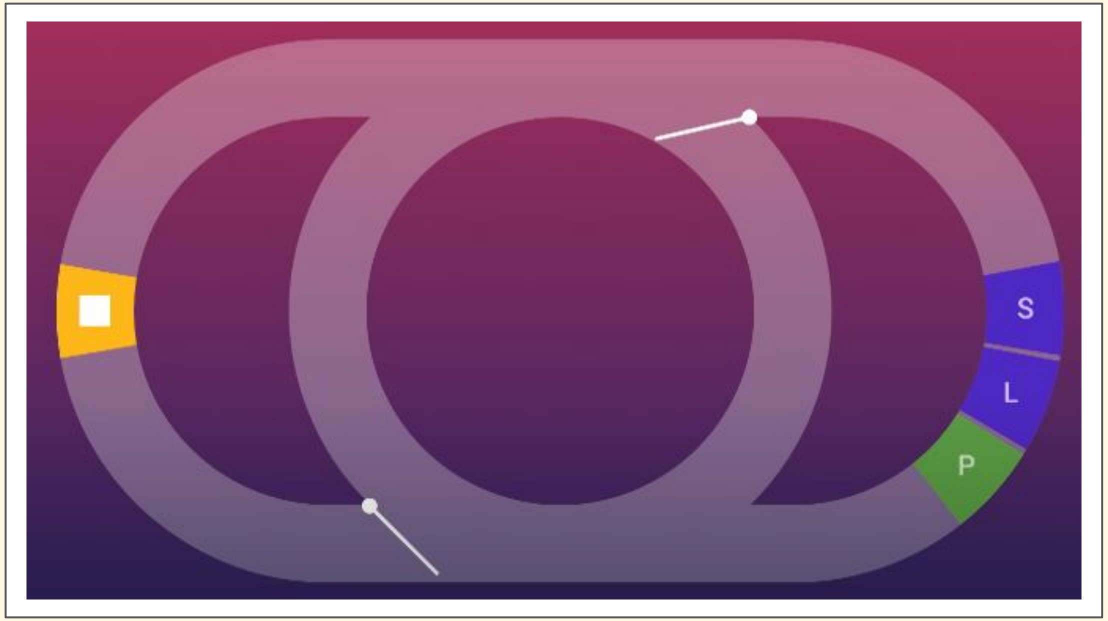
web.dev

The main thread

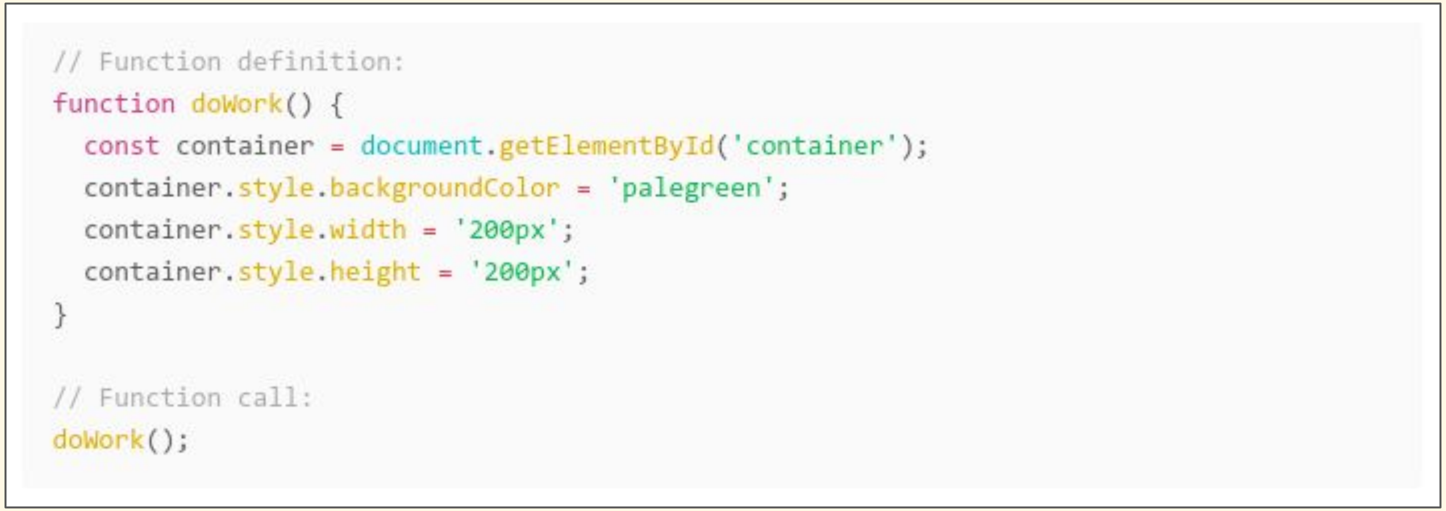


Source: https://aerotwist.com/blog/the-anatomy-of-a-frame/

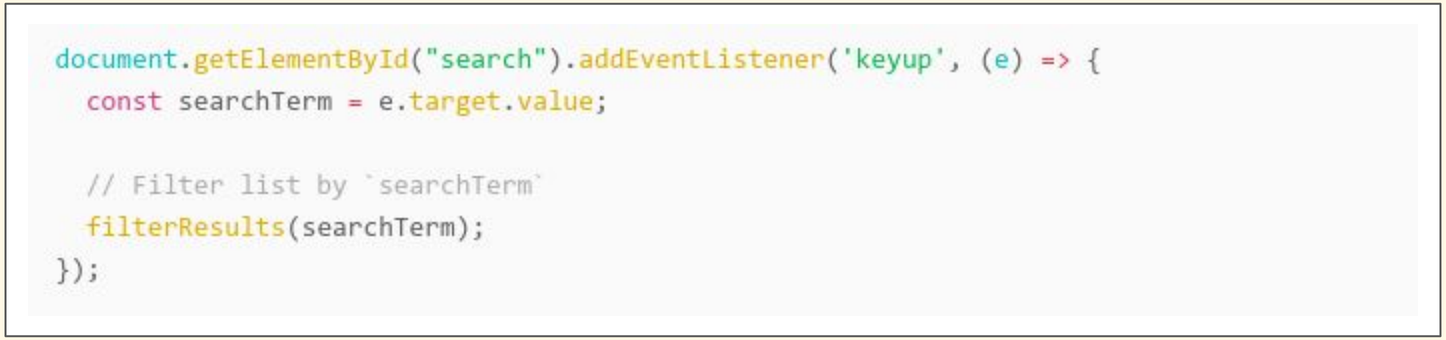
The event loop



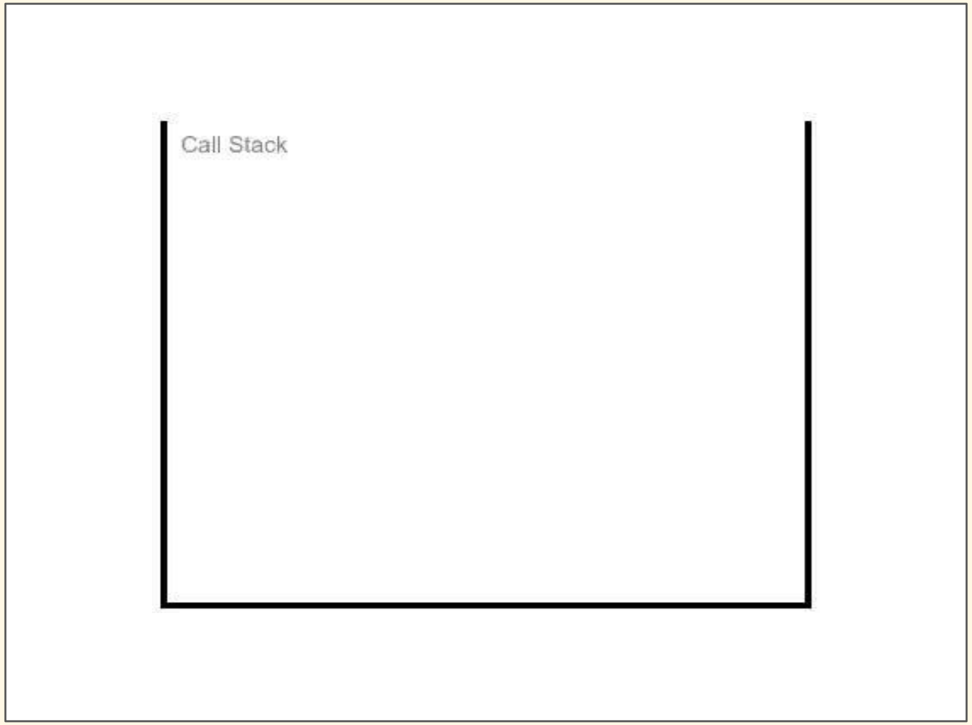
Source: https://www.youtube.com/watch?v=cCOL7MC4Pl0

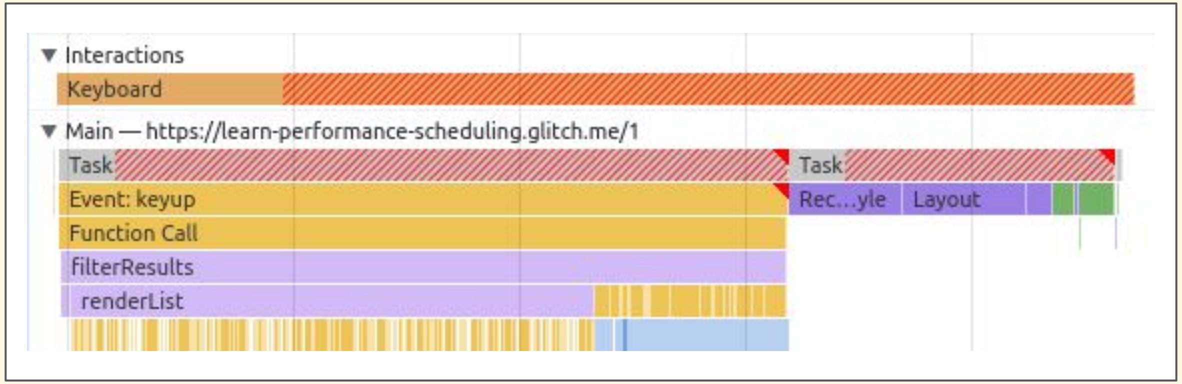


```javascript
// Function definition:
function doWork() {
  const container = document.getElementById('container');
  container.style.backgroundColor = 'palegreen';
  container.style.width = '200px';
  container.style.height = '200px';
}

// Function call:
doWork();
```



```
document.getElementById("search").addEventListener('keyup', (e) => {
  const searchTerm = e.target.value;

  // Filter list by `searchTerm`
  filterResults(searchTerm);
});
```

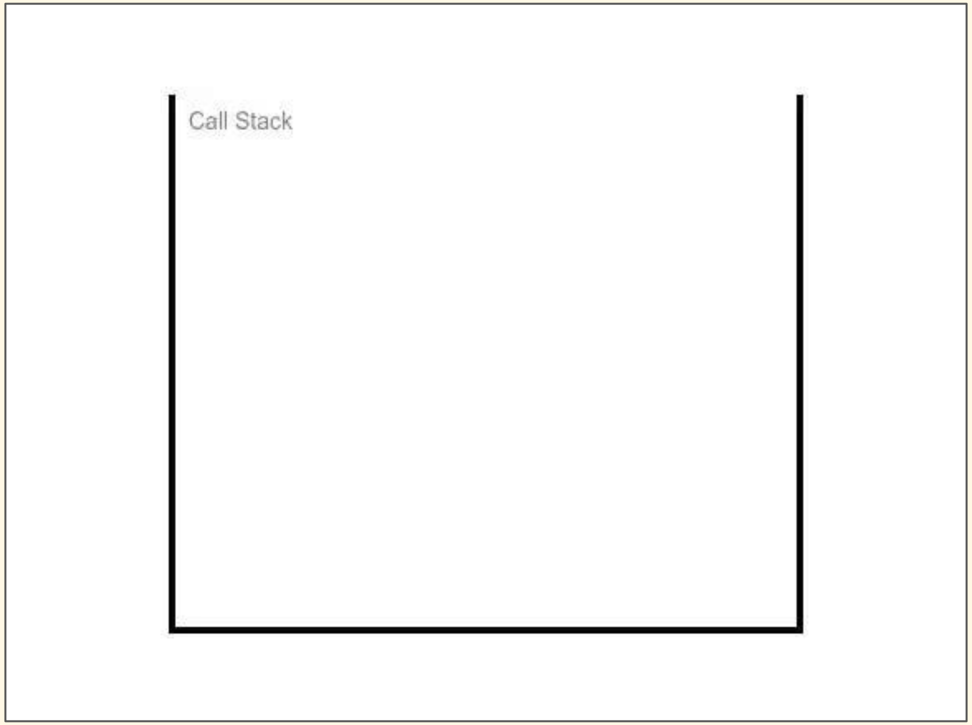

Call Stack

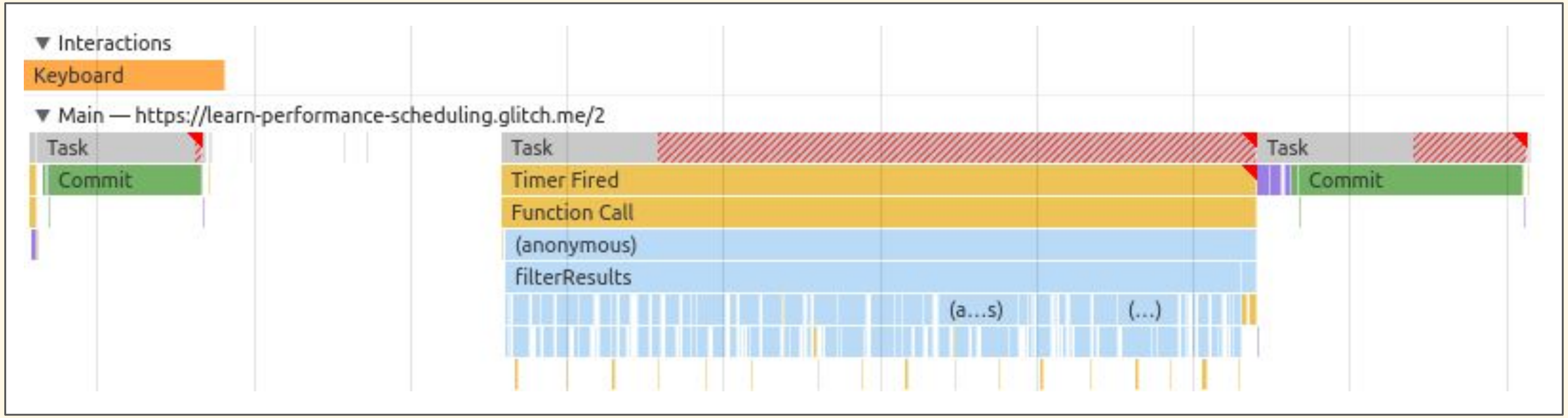

Interactions
Keyboard
Main — https://learn-performance-scheduling.glitch.me/1
Task
Task
Event: keyup
Rec…yle
Layout
Function Call
filterResults
renderList

`setTimeout`

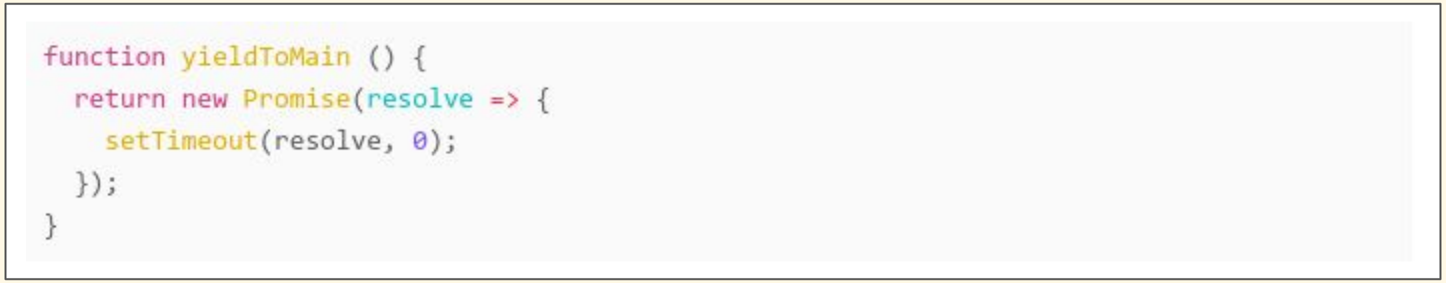
```
setTimeout(() => {
  doWork();
}, 1000);
```

```
document.getElementById('search').addEventListener('keyup', (e) => {
  const searchTerm = e.target.value;

  // Yield to the main thread
  setTimeout(() => {
    // Filter list by `searchTerm`
    filterResults(searchTerm);
  }, 0);
});
```

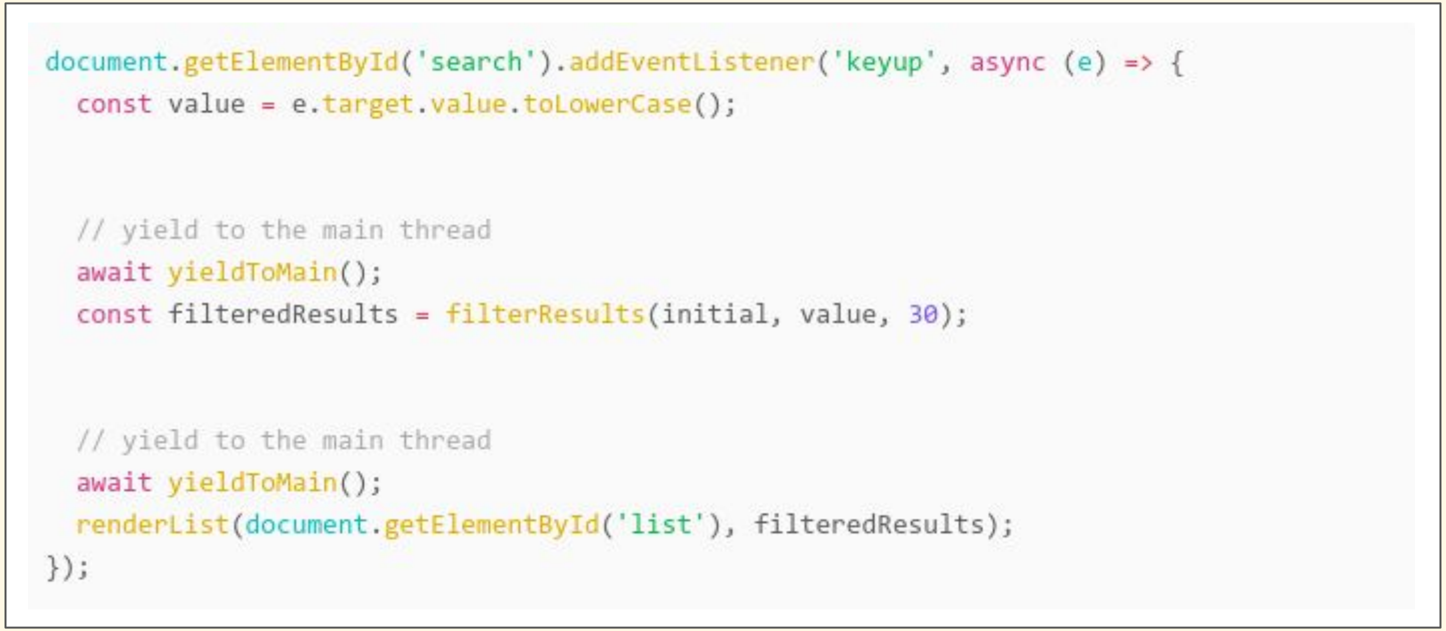

Call Stack


▼ Interactions
Keyboard
▼ Main — https://learn-performance-scheduling.glitch.me/2
Task
Commit
Task
Timer Fired
Function Call
(anonymous)
filterResults
(a…s)
(…)
Task
Commit

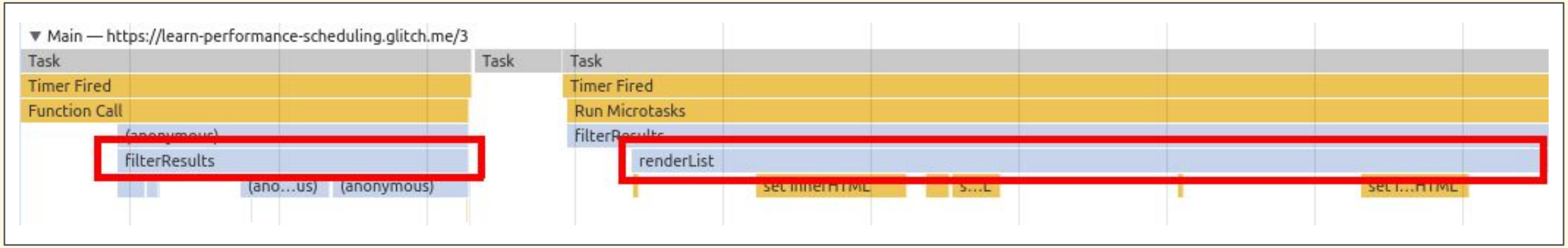
async / await



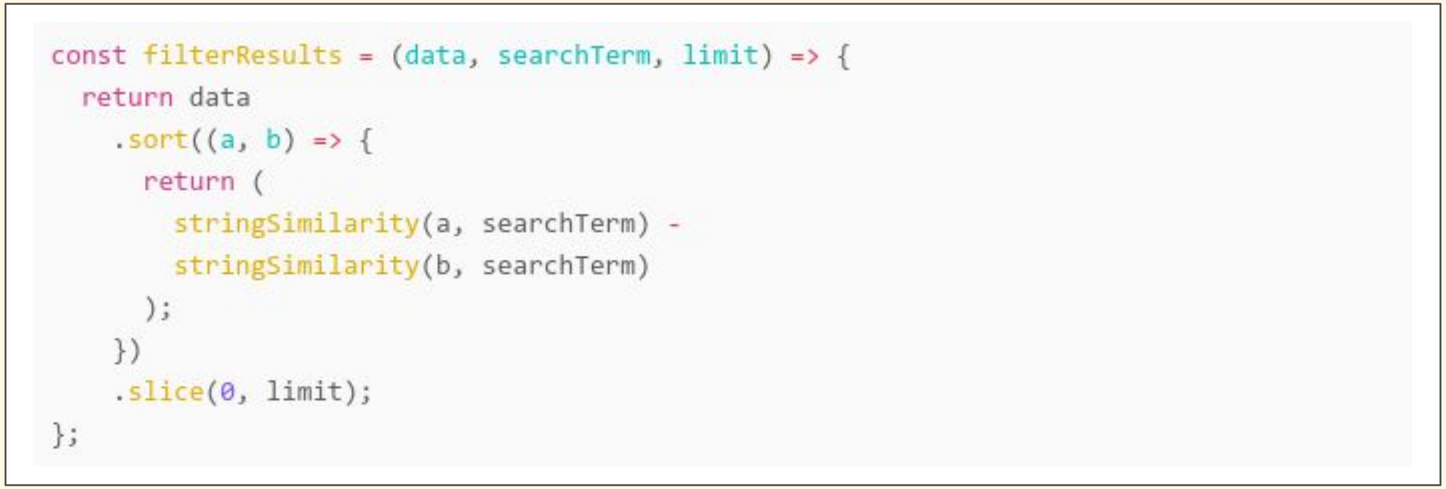
```javascript
function yieldToMain () {
  return new Promise(resolve => {
    setTimeout(resolve, 0);
  });
}
```



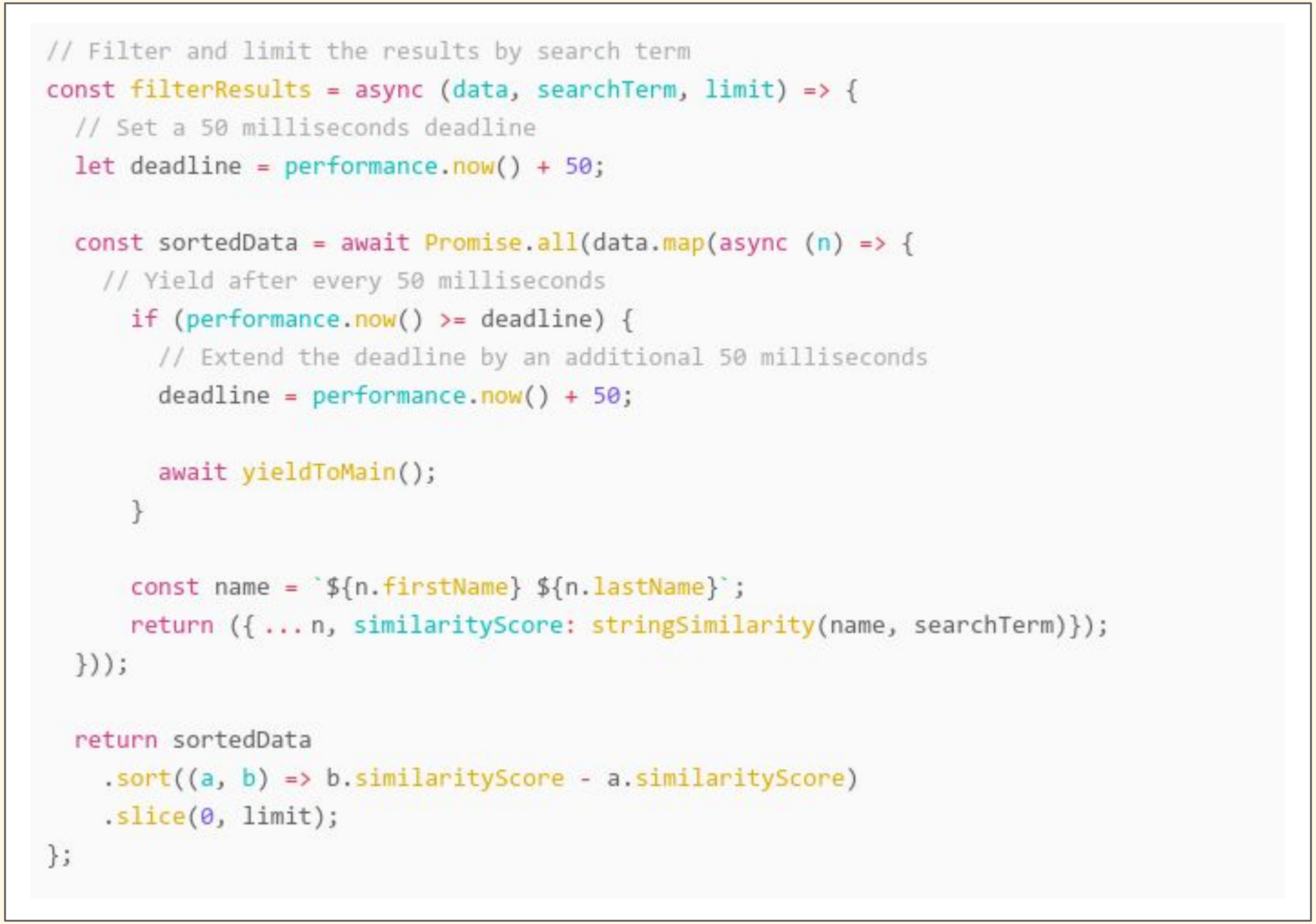
```
document.getElementById('search').addEventListener('keyup', async (e) => {
  const value = e.target.value.toLowerCase();


  // yield to the main thread
  await yieldToMain();
  const filteredResults = filterResults(initial, value, 30);


  // yield to the main thread
  await yieldToMain();
  renderList(document.getElementById('list'), filteredResults);
});
```


▼ Main — https://learn-performance-scheduling.glitch.me/3
Task
Task
Task
Timer Fired
Timer Fired
Function Call
Run Microtasks
(anonymous)
filterResults
filterResults
renderList
(ano…us)
(anonymous)
set innerHTML
S…L
Set I…HTML

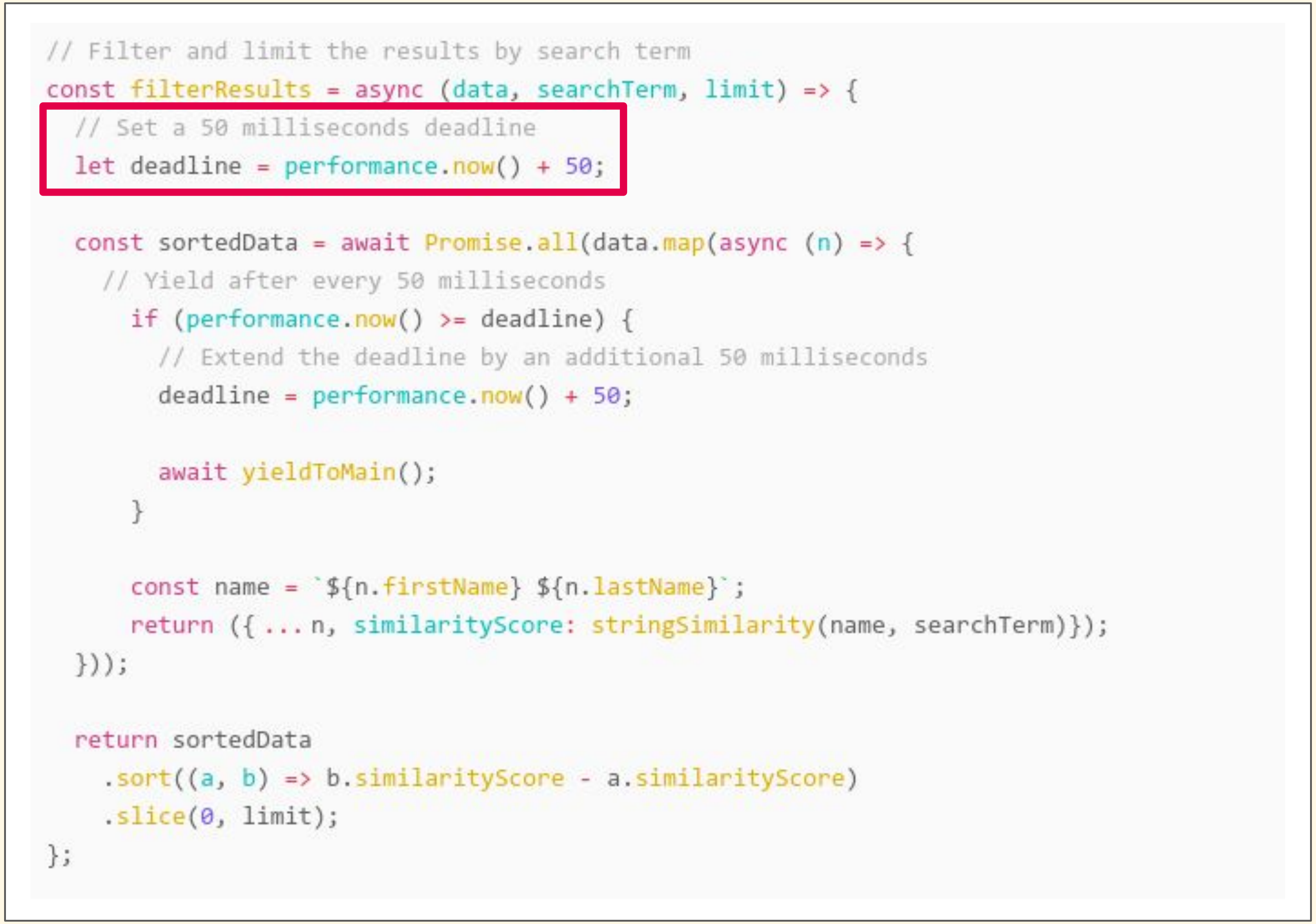
When to yield?



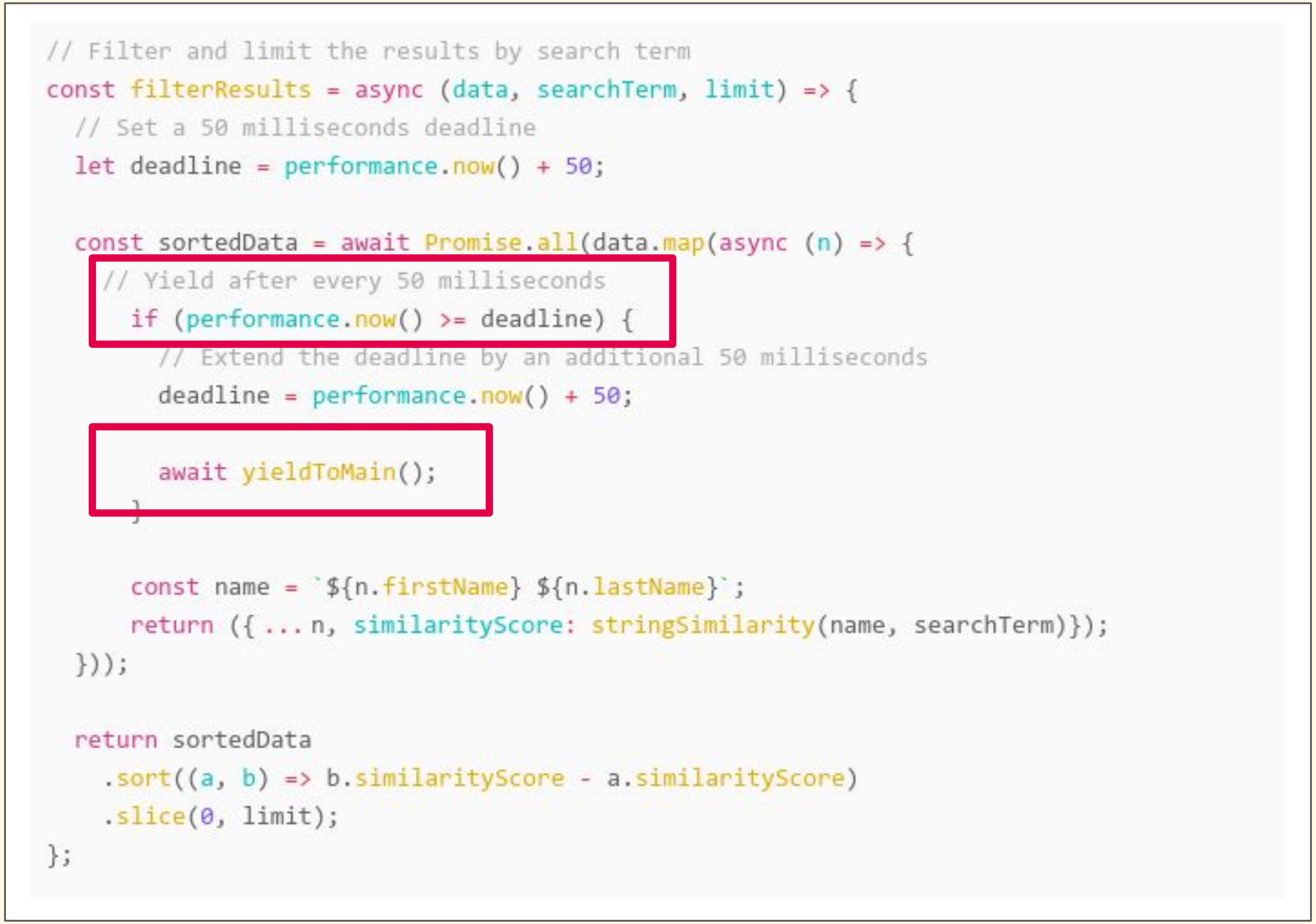
```
const filterResults = (data, searchTerm, limit) => {
  return data
    .sort((a, b) => {
      return (
        stringSimilarity(a, searchTerm) -
        stringSimilarity(b, searchTerm)
      );
    })
    .slice(0, limit);
};
```



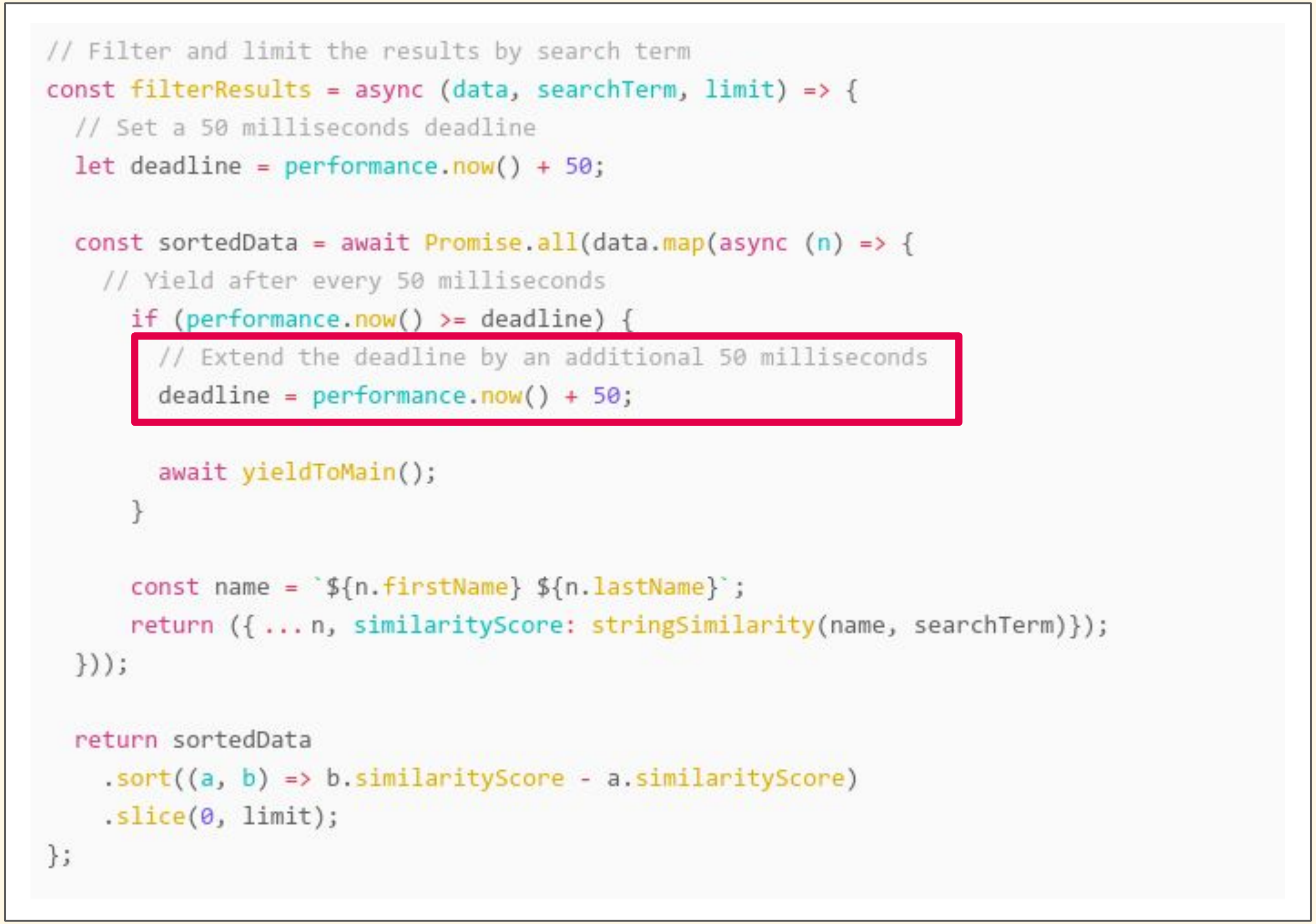
```javascript
// Filter and limit the results by search term
const filterResults = async (data, searchTerm, limit) => {
  // Set a 50 milliseconds deadline
  let deadline = performance.now() + 50;

  const sortedData = await Promise.all(data.map(async (n) => {
    // Yield after every 50 milliseconds
      if (performance.now() >= deadline) {
        // Extend the deadline by an additional 50 milliseconds
        deadline = performance.now() + 50;

        await yieldToMain();
      }

      const name = `${n.firstName} ${n.lastName}`;
      return ({ ...n, similarityScore: stringSimilarity(name, searchTerm)});
  }));

  return sortedData
    .sort((a, b) => b.similarityScore - a.similarityScore)
    .slice(0, limit);
};
```



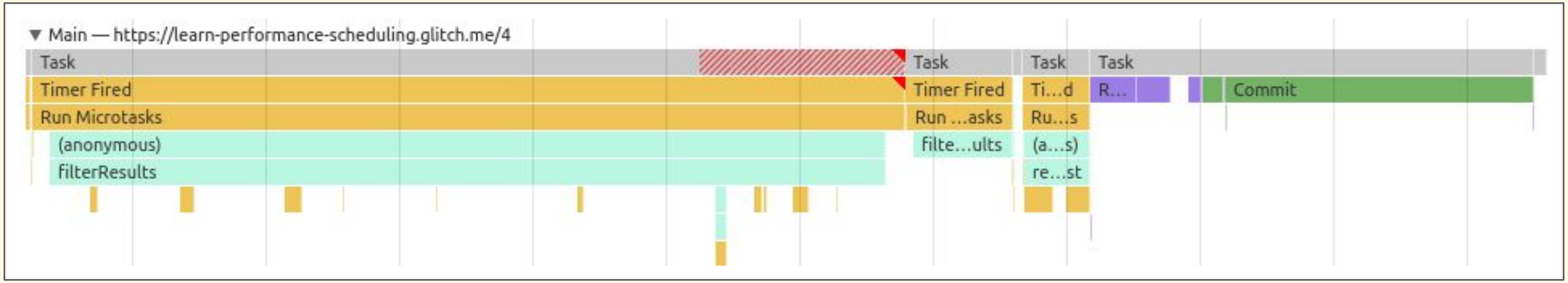
```
// Filter and limit the results by search term
const filterResults = async (data, searchTerm, limit) => {
  // Set a 50 milliseconds deadline
  let deadline = performance.now() + 50;

  const sortedData = await Promise.all(data.map(async (n) => {
    // Yield after every 50 milliseconds
      if (performance.now() >= deadline) {
        // Extend the deadline by an additional 50 milliseconds
        deadline = performance.now() + 50;

        await yieldToMain();
      }

      const name = `${n.firstName} ${n.lastName}`;
      return ({ ...n, similarityScore: stringSimilarity(name, searchTerm)});
  }));

  return sortedData
    .sort((a, b) => b.similarityScore - a.similarityScore)
    .slice(0, limit);
};
```



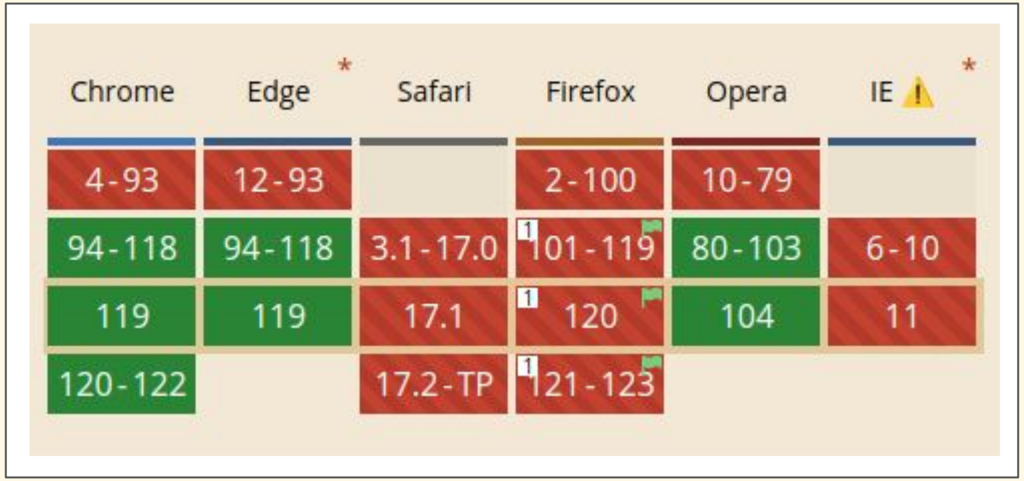
```
// Filter and limit the results by search term
const filterResults = async (data, searchTerm, limit) => {
  // Set a 50 milliseconds deadline
  let deadline = performance.now() + 50;

  const sortedData = await Promise.all(data.map(async (n) => {
    // Yield after every 50 milliseconds
      if (performance.now() >= deadline) {
        // Extend the deadline by an additional 50 milliseconds
        deadline = performance.now() + 50;

        await yieldToMain();
      }

      const name = `${n.firstName} ${n.lastName}`;
      return ({ ...n, similarityScore: stringSimilarity(name, searchTerm)});
  }));

  return sortedData
    .sort((a, b) => b.similarityScore - a.similarityScore)
    .slice(0, limit);
};
```



```javascript
// Filter and limit the results by search term
const filterResults = async (data, searchTerm, limit) => {
  // Set a 50 milliseconds deadline
  let deadline = performance.now() + 50;

  const sortedData = await Promise.all(data.map(async (n) => {
    // Yield after every 50 milliseconds
      if (performance.now() >= deadline) {
        // Extend the deadline by an additional 50 milliseconds
        deadline = performance.now() + 50;

        await yieldToMain();
      }

      const name = `${n.firstName} ${n.lastName}`;
      return ({ ...n, similarityScore: stringSimilarity(name, searchTerm)});
  }));

  return sortedData
    .sort((a, b) => b.similarityScore - a.similarityScore)
    .slice(0, limit);
};
```


▼ Main — https://learn-performance-scheduling.glitch.me/4
Task
Timer Fired
Run Microtasks
(anonymous)
filterResults
Task
Timer Fired
Run …asks
filte…ults
Task
Ti…d
Ru…s
(a…s)
re…st
Task
R…
Commit

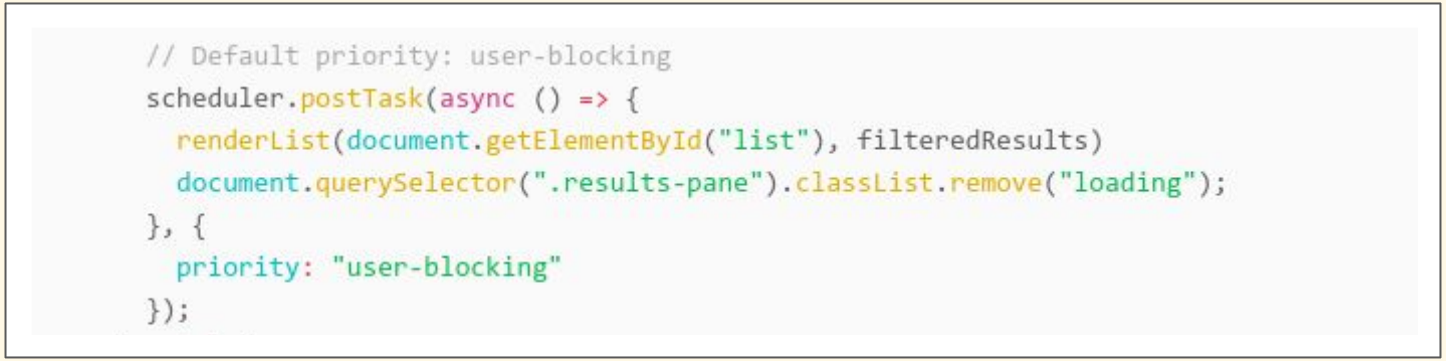
```
scheduler.postTask
```



Chrome	Edge *	Safari	Firefox	Opera	IE ⚠ *
4-93	12-93		2-100	10-79	
94-118	94-118	3.1-17.0	[1] 101-119	80-103	6-10
119	119	17.1	[1] 120	104	11
120-122		17.2-TP	[1] 121-123		

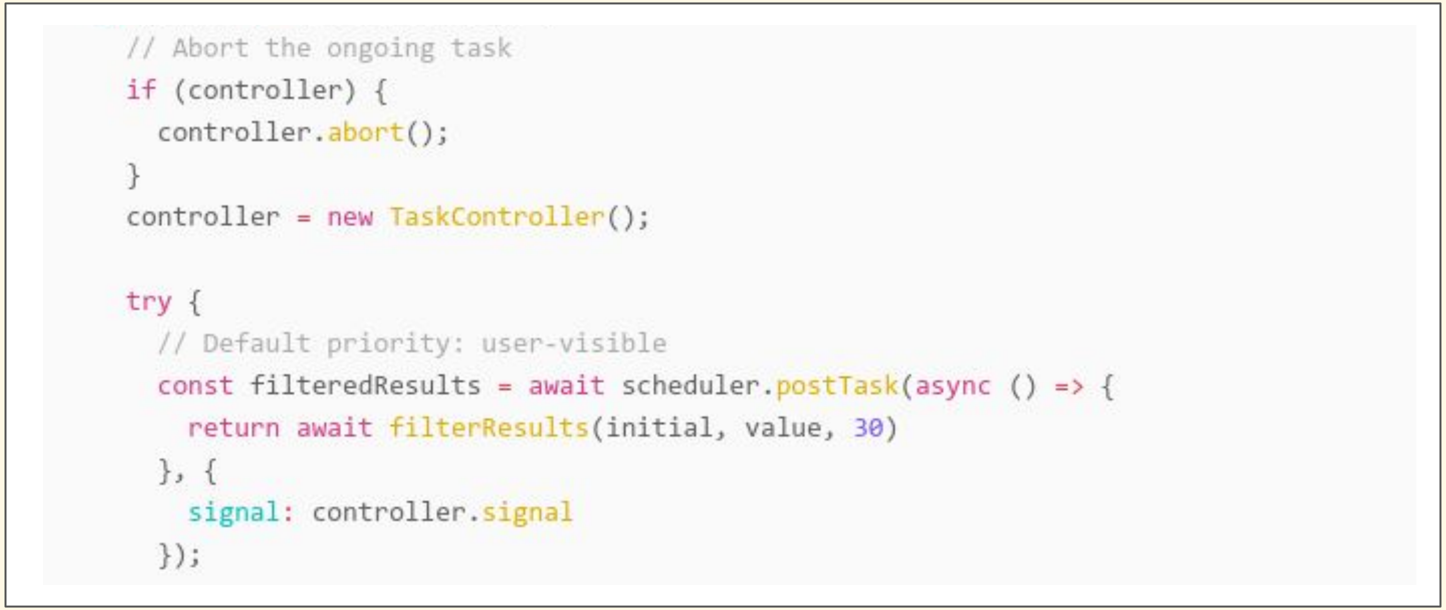
Source: https://caniuse.com/mdn-api_scheduler_posttask

```
user-blocking
user-visible
background
```

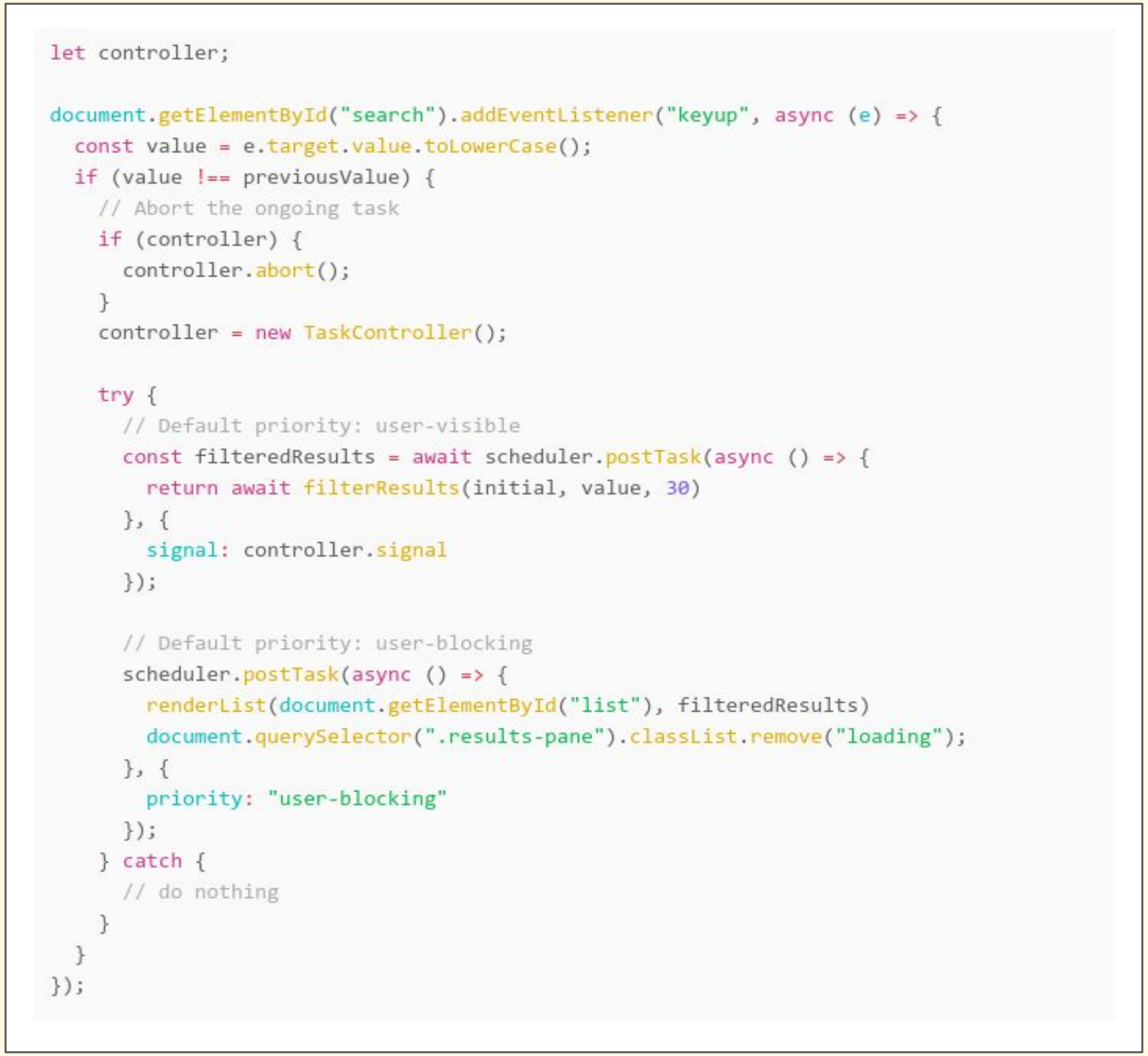


```
// Default priority: user-blocking
scheduler.postTask(async () => {
  renderList(document.getElementById("list"), filteredResults)
  document.querySelector(".results-pane").classList.remove("loading");
}, {
  priority: "user-blocking"
});
```

TaskController

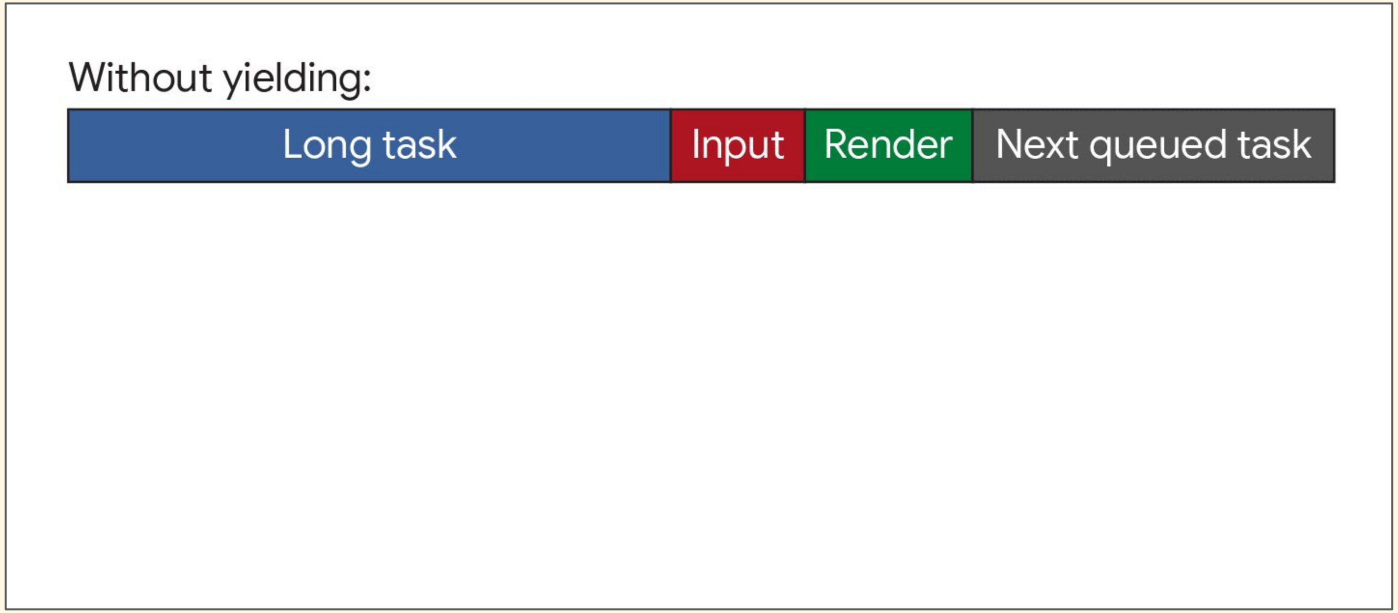


```
// Abort the ongoing task
if (controller) {
  controller.abort();
}
controller = new TaskController();

try {
  // Default priority: user-visible
  const filteredResults = await scheduler.postTask(async () => {
    return await filterResults(initial, value, 30)
  }, {
    signal: controller.signal
  });
```

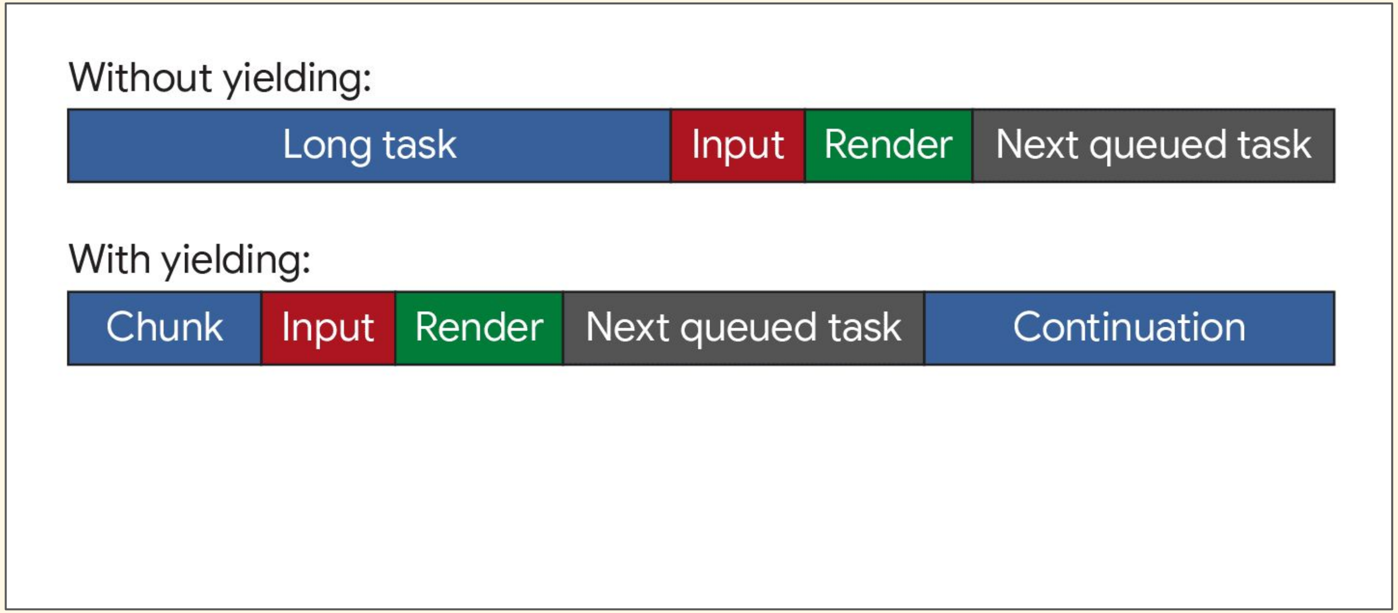


```javascript
let controller;

document.getElementById("search").addEventListener("keyup", async (e) => {
  const value = e.target.value.toLowerCase();
  if (value !== previousValue) {
    // Abort the ongoing task
    if (controller) {
      controller.abort();
    }
    controller = new TaskController();

    try {
      // Default priority: user-visible
      const filteredResults = await scheduler.postTask(async () => {
        return await filterResults(initial, value, 30)
      }, {
        signal: controller.signal
      });

      // Default priority: user-blocking
      scheduler.postTask(async () => {
        renderList(document.getElementById("list"), filteredResults)
        document.querySelector(".results-pane").classList.remove("loading");
      }, {
        priority: "user-blocking"
      });
    } catch {
      // do nothing
    }
  }
});
```

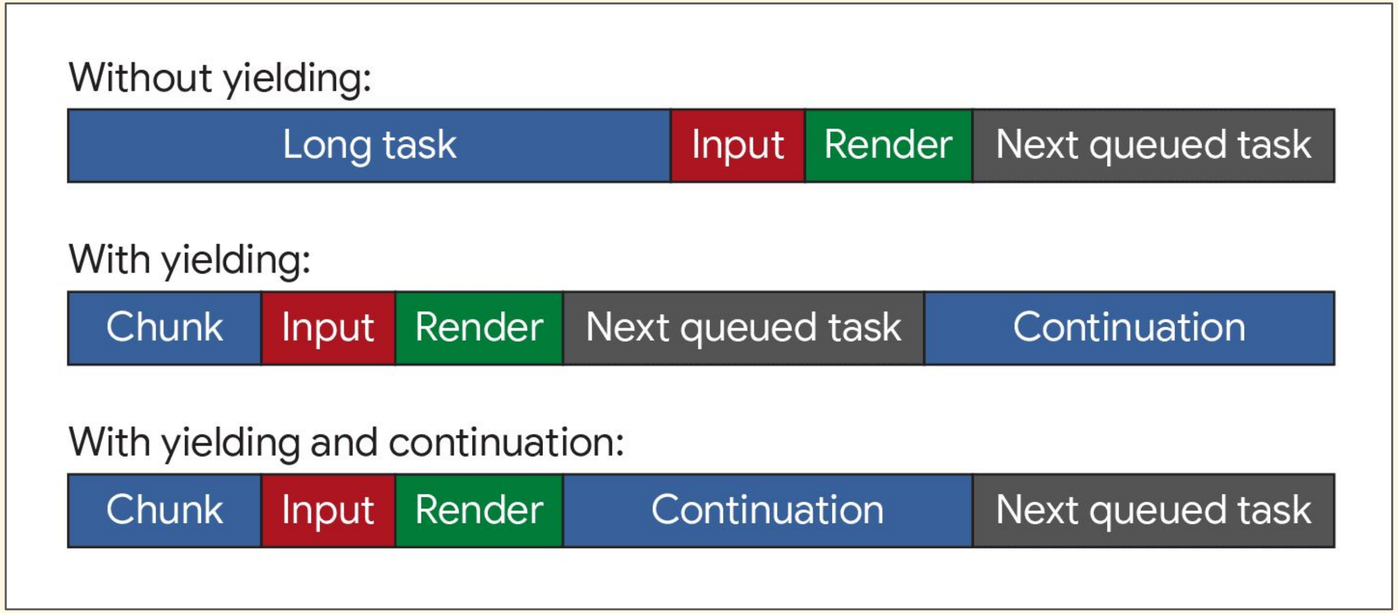
`scheduler.yield`



Source: https://web.dev/articles/optimize-long-tasks

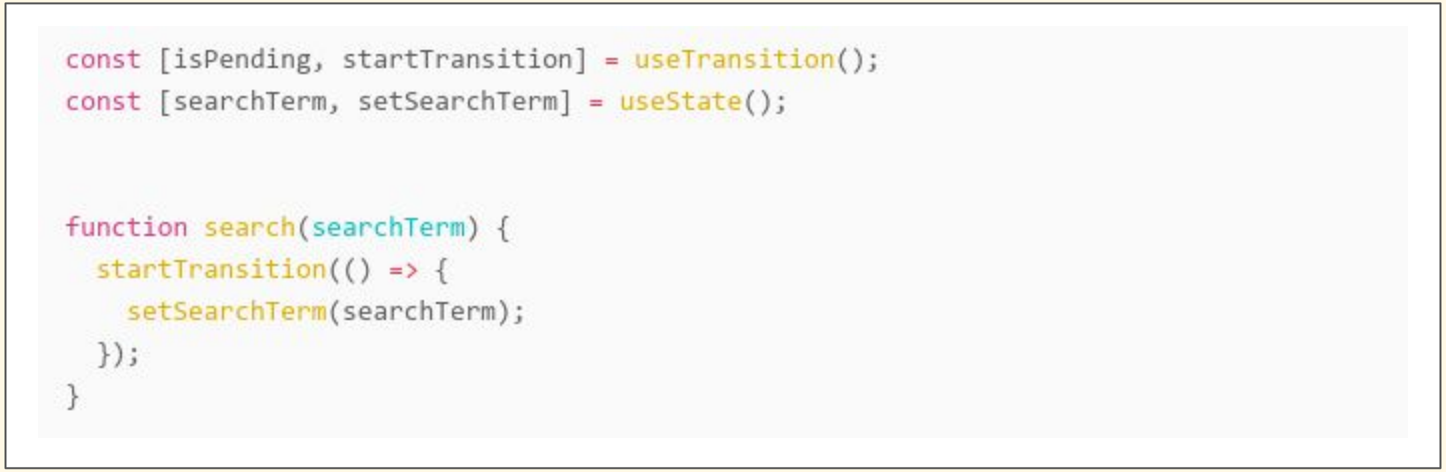


Source: https://web.dev/articles/optimize-long-tasks



Source: https://web.dev/articles/optimize-long-tasks

React’s Concurrency Mode



```
const [isPending, startTransition] = useTransition();
const [searchTerm, setSearchTerm] = useState();


function search(searchTerm) {
  startTransition(() => {
    setSearchTerm(searchTerm);
  });
}
```

Thank you!

References & links

- Slides: https://imkev.dev/devfest-2023
- Learn Performance: https://web.dev/learn/performance
- Demos: https://learn-performance-scheduling.glitch.me/
- React `useTransition` demo: https://learn-performance-react-use-transition.glitch.me/
- Geekbench 5 Multi-Core Scores: https://toot.cafe/@slightlyoff/111527384063250418
- Interaction to Next Paint video: https://imkev.dev/inp
- Interaction to Next Paint (INP): https://web.dev/articles/inp
- Anatomy of a Pixel: https://aerotwist.com/blog/the-anatomy-of-a-frame/
- In the loop: https://www.youtube.com/watch?v=cCOL7MC4Pl0
- Compatibility table for `scheduler.postTask`: https://caniuse.com/mdn-api_scheduler_posttask
- Optimize long tasks: https://web.dev/articles/optimize-long-tasks